i.MX50 Multimedia Applications Processor Reference Manual

Document Number: IMX50RM
Rev. 1
10/2011

freescale™

Contents

Section Number **Title** **Page**

Chapter 1
Introduction

1.1 About This Document....125

1.1.1 Audience....125

1.1.2 Organization....125

1.1.3 Conventions....126

1.1.4 Register Access....127

1.1.4.1 Register Diagram Field Access Type Legend....127

1.1.4.2 Register Macro Usage....128

1.1.5 Acronyms and Abbreviations....128

1.2 Target Applications....130

1.3 Features....130

1.4 Architectural Overview....134

1.4.1 Block Diagram....134

1.4.2 Major Subsystems....135

1.4.3 Architectural Partitioning....136

1.5 i.MX50 Modules List....137

Chapter 2
Memory Map

2.1 Introduction....145

2.2 ARM Platform Memory Map....145

2.3 DMA Memory Map....150

Chapter 3
Interrupts and DMA Events

3.1 Overview....153

3.2 AP Interrupts....153

3.3 SDMA Event Mapping....157

Chapter 4
External Signals and Pin Multiplexing

4.1 Introduction....159

4.2 External Signals....159

Chapter 5
Clock Controller Module (CCM)

5.1 Introduction....279

5.1.1 Overview....280

5.1.2 Features....281

5.1.3 CCM Block Diagram....281

5.2 CCM External Signals Description....283

5.3 Functional Description....284

5.3.1 Clock Generation....284

5.3.1.1 External Low Frequency Clock—CKIL....285

5.3.1.2 External High Frequency Clock—CKIH and Internal Oscillator....285

5.3.1.3 DPLL Reference Clock....285

5.3.1.4 Analog PLL Reference Clock....285

5.3.1.5 CCM Internal Clock Generation....286

5.3.1.5.1 CCM_CLK_SWITCHER....286

5.3.1.5.2 PLL Bypass Procedure....288

5.3.1.5.3 DPLLC Reference Clock Connectivity....288

5.3.1.5.4 PLL Clock Change....288

5.3.1.5.5 CCM_CLK_IGNITION....289

5.3.1.5.6 CCM_CLK_PUSM Reset Sequence....291

5.3.1.5.7 Reset Values for DPLLC....292

5.3.1.5.8 Analog 480 PLL and PFD....292

5.3.1.5.9 CCM_CLK_ROOT_GEN....292

5.3.1.5.10 Initial Reset Values Controlled by SJC....296

5.3.1.5.11 Divider Change Handshake....296

5.3.1.5.12 CKIL Synchronizing to ipg_clk....297
5.3.1.6 DPLL's Disabling/Enabling....298
5.3.1.7 Observability Output Signals....298
5.3.1.8 Low Power Clock Gating Module (LPCG)....298
5.3.2 DVFS Support....301
5.3.2.1 ARM Clock Domain Frequency Change....301
5.3.2.2 Peripheral Clock Domain Frequency Change....303
5.3.2.3 Peripherals Restrictions in DVFS Scenario....304
5.3.3 Auto-Slow Mode....305
5.3.4 Power Modes....305
5.3.4.1 Run Mode....305
5.3.4.2 Wait Mode....306
5.3.4.2.1 Wait Mode Procedure....306
5.3.4.2.2 Exiting Wait Mode....308
5.3.4.3 Stop Mode....308
5.3.4.3.1 Entering Stop Mode....309
5.3.4.3.2 Exiting Stop Mode....311
5.3.4.3.3 PMIC Signal Description....315
5.3.4.4 Low Power Audio Playback Mode (LP-APM)....316
5.3.4.4.1 Low Power APM Mode Definition....316
5.3.4.4.2 LP-APM Mode Restrictions....317
5.3.4.4.3 LP-APM Entry and Exit....317
5.3.4.5 Recomendations for Using Low Power Consumption from CCM....320
5.4 CCM Memory Map/Register Definition....321
5.4.1 CCM Control Register (CCM_CCR)....323
5.4.2 CCM Control Divider Register (CCM_CCDR)....324
5.4.3 CCM Status Register (CCM_CSR)....325
5.4.4 CCM Clock Switcher Register (CCM_CCSR)....326
5.4.5 CCM ARM Clock Root Register (CCM_CACRR)....328

5.4.6 CCM Bus Clock Divider Register (CCM_CBCDR).....329
5.4.7 CCM Bus Clock Multiplexer Register (CCM_CBCMR).....332
5.4.8 CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1).....333
5.4.9 CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2).....336
5.4.10 CCM Serial Clock Divider Register 1 (CCM_CSCDR1).....336
5.4.11 CCM SSI1 Clock Divider Register (CCM_CS1CDR).....339
5.4.12 CCM SSI2 Clock Divider Register (CCM_CS2CDR).....340
5.4.13 CCM DI Clock Divider Register (CCM_CDCDR).....342
5.4.14 CCM HSC Clock Divider Register (CCM_CHSCCDR).....342
5.4.15 CCM Serial Clock Divider Register 2 (CCM_CSCDR2).....343
5.4.16 CCM Serial Clock Divider Register 3 (CCM_CSCDR3).....344
5.4.17 CCM Serial Clock Divider Register 4 (CCM_CSCDR4).....344
5.4.18 CCM Wakeup Detector Register (CCM_CWDR).....345
5.4.19 CCM Divider Handshake In-Process Register (CCM_CDHIPR).....346
5.4.20 CCM DVFS Control Register (CCM_CDCR).....348
5.4.21 CCM Testing Observability Register (CCM_CTOR).....349
5.4.22 CCM Low Power Control Register (CCM_CLPCR).....351
5.4.23 CCM Interrupt Status Register (CCM_CISR).....354
5.4.24 CCM Interrupt Mask Register (CCM_CIMR).....356
5.4.25 CCM Clock Output Source Register (CCM_CCOSR).....358
5.4.26 CCM General Purpose Register (CCM_CGPR).....361
5.4.27 CCM Clock Gating Register 0 (CCM_CCGR0).....361
5.4.28 CCM Clock Gating Register 1 (CCM_CCGR1).....364
5.4.29 CCM Clock Gating Register 2 (CCM_CCGR2).....366
5.4.30 CCM Clock Gating Register 3 (CCM_CCGR3).....369
5.4.31 CCM Clock Gating Register 4 (CCM_CCGR4).....371
5.4.32 CCM Clock Gating Register 5 (CCM_CCGR5).....374
5.4.33 CCM Clock Gating Register 6 (CCM_CCGR6).....376
5.4.34 CCM Clock Gating Register 7 (CCM_CCGR7).....379

5.4.35 CCM Module Enable Override Register (CCM_CMEOR)....382
5.4.36 CCM Control Status Register 2 (CCM_CSR2)....383
5.4.37 CCM Clock Sequence Bypass (CCM_CLKSEQ_BYPASS)....385
5.4.38 CCM System Clock Register (CCM_CLK_SYS)....387
5.4.39 CCM DDR Clock Register (CCM_CLK_DDR)....388
5.4.40 CCM ELCDIF PIX Clock Serial Divide Register (CCM_ELCDIFPIX)....389
5.4.41 CCM EPDC PIX Clock Serial Divide Register (CCM_EPDCPIX)....390
5.4.42 CCM DISPLAY_AXI Clock Divide Register (CCM_DISPLAY_AXI)....392
5.4.43 CCM EPDC_AXI Clock Divide Register (CCM_EPDC_AXI)....394
5.4.44 CCM GPMI Clock Divide Register (CCM_GPMI)....395
5.4.45 CCM BCH Clock Divide Register (CCM_BCH)....396
5.4.46 CCM MSHC_XMSCKI Clock Divide Register (CCM_MSHC_XMSCKI)....397
5.4.47 Clock Gating bits (CG(i))....398
5.5 CCM Analog Memory Map/Register Definition....404
5.5.1 Fractional Clock Control Register 0 (CCM_ANALOG_FRAC0*n*)....406
5.5.2 Fractional Clock Control Register 1 (CCM_ANALOG_FRAC1*n*)....408
5.5.3 Miscellaneous Register Description (CCM_ANALOG_MISC*n*)....410
5.5.4 PLL Control Register (CCM_ANALOG_PLLCTRL*n*)....411

Chapter 6
System Boot

6.1 Introduction....413
6.2 Boot Modes....414
6.2.1 Boot Mode Pin Settings....414
6.2.2 High Level Boot Sequence....415
6.2.3 Overview of USB Low Power Boot....416
6.2.4 Internal Boot (BOOT_MODE [1:0] = 00)....417
6.2.5 Internal Boot-Fuses Only (BOOT_MODE [1:0] = 10)....418
6.2.6 USB Downloader (BOOT_MODE [1:0] = 11)....418
6.2.7 Boot Security Settings....419

6.3 Device Configuration....419
6.3.1 Boot OCOTP bit Descriptions....420
6.3.2 GPIO Boot Overrides....423
6.3.3 Device Configuration Data....424
6.4 Device Initialization....424
6.4.1 Memory Map....424
6.4.2 Boot Module Activation....425
6.4.3 Clocks at Boot Time....426
6.4.4 Clock Initialization....426
6.4.4.1 NORMAL Mode....426
6.4.4.2 USB Low-Power Boot Mode (LPB)....428
6.4.5 Enabling MMU/Caches....428
6.4.6 Error Logging....429
6.4.7 Exception Handling....429
6.4.8 Interrupt Handling During Boot....430
6.5 Boot Devices (Internal Boot)....430
6.5.1 NOR/OneNAND Boot from EIM Interface....430
6.5.1.1 IOMUX Configuration for EIM....431
6.5.1.2 NOR Flash Boot Operation....432
6.5.1.3 OneNAND Flash Boot Operation....432
6.5.2 Raw NAND Boot from GPMI-2 Interface....433
6.5.2.1 NAND eFUSE/GPIO Configuration....433
6.5.2.2 NAND Flash Boot Flow and Boot Control Blocks(BCB)....434
6.5.2.3 Firmware Configuration Block....437
6.5.2.4 Discovered Bad Block Table....440
6.5.2.5 Bad Block Handling in the ROM....441
6.5.2.6 Toggle Mode DDR NAND Boot....442
6.5.2.6.1 Configures GPMI and BCH Clocks to Run at 33 MHz....442
6.5.2.6.2 Setup DMA for DDR Transfers....442

6.5.2.6.3 Reconfigure Timing and Speed Using Values in FCB....442
6.5.2.7 Typical NAND Page Organization....443
6.5.2.7.1 BCH ECC Page Organization....443
6.5.2.7.2 Metadata....444
6.5.2.8 IOMUX Configuration for NAND Flash....444
6.5.3 Expansion Device Support....445
6.5.3.1 IOMUX Configuration of eSDHC....447
6.5.3.2 MMC and eMMC Boot....447
6.5.4 SD and eSD....453
6.5.5 Serial ROM Support via SPI and I2C....454
6.5.5.1 I2C EEPROM Boot....455
6.5.5.2 SPI EEPROM/Flash Boot....456
6.6 Program Image....458
6.6.1 Image Vector Table and Boot Data....459
6.6.1.1 Image Vector Table Structure....460
6.6.1.2 Boot Data Structure....461
6.6.2 Device Configuration Data (DCD)....461
6.6.2.1 Write Data Command....464
6.6.2.2 Check Data Command....465
6.6.2.3 NOP Command....467
6.7 Plugin Image....467
6.8 USB Low-Power Boot....468
6.8.1 LPB Mode Flow....469
6.9 USB Downloader (BOOT_MODE[1:0] = 11)....471
6.9.1 USB....472
6.9.1.1 USB Configuration....473
6.9.1.2 Serial Download protocol....474
6.9.1.3 SDP Command....474
6.9.1.3.1 READ REGISTER....475

6.9.1.3.2 WRITE REGISTER....475
6.9.1.3.3 WRITE_FILE....476
6.9.1.3.4 ERROR_STATUS....477
6.9.1.3.5 DCD WRITE....478
6.9.1.3.6 JUMP ADDRESS....479
6.9.2 External Entry....480
6.9.2.1 pu_irom_boot_decision....480
6.10 High Assurance Boot (HAB)....481
6.10.1 ROM Vector Table Addresses....482
6.10.2 SRTC Initialization....482

Chapter 7
System Debug

7.1 Overview....483
7.1.1 Debug Strategy....483
7.2 System JTAG Controller-SJC....484
7.2.1 JTAG Topology....484
7.2.2 System JTAG Controller Main Features....485
7.2.3 SJC TAP Port....486
7.2.4 SJC Main Blocks....486
7.2.5 i.MX50 Specific SJC Features....486
7.2.5.1 JTAG Disable Mode....486
7.2.5.2 SJC i.MX50 Specific Registers Summary....487
7.2.5.3 JTAG ID....487
7.3 CoreSight Design Kit....487
7.3.1 Memory Map and Register Definition....487
7.3.2 CoreSight Clock Enable....488
7.3.3 CoreSight Debug Access Port (DAP) and DAP_SYS....488
7.3.4 Embedded Cross Trigger (ECT)....489
7.3.4.1 CoreSight CTM....491

7.3.4.2 CoreSight CTI....491
7.3.4.3 Extended CTI (CTI Wrapper)....494
7.3.5 CoreSight Trace Port Interface (TPIU)....495
7.4 Cortex-A8 Core and Platform....495
7.4.1 Cortex-A8 Core Debug Support Features....495
7.4.2 Embedded Cross Trigger Interface....496
7.4.3 Additional Platform Debug Functionality....496
7.5 Smart DMA (SDMA) Core....496
7.5.1 SDMA On Chip Emulation Module (OnCE) Feature Summary....497
7.5.2 Other SDMA Debug Functionality....497
7.6 Debug Visibility-IOMUX....498
7.7 Supported Tools....499
7.8 Interrupt Visibility....499
7.9 Miscellaneous....499
7.9.1 SOC-Level Bus Trace....499
7.9.2 Clock/Reset/Power....500

Chapter 8
ARM Cortex A8 Platform (ARM Platform)

8.1 Introduction....501
8.2 Overview....501
8.2.1 Core Platform Sub-Blocks....504
8.2.1.1 ARM platform....504
8.2.1.2 Instruction Fetch....505
8.2.1.3 Instruction Decode....505
8.2.1.4 Instruction Execute....506
8.2.1.5 Load/Store....506
8.2.1.6 L2 Cache....506
8.2.1.7 NEON....507
8.2.1.8 Processor Debug Unit....507

8.2.1.9 Embedded Trace Marcocell (ETM).....507
8.2.1.10 Cross Trigger Interface 0 (CTI0).....508
8.2.2 Summary of Remaining Platform Components.....508
8.2.2.1 Platform Controller 508
8.2.2.2 Debug Sub-Blocks.....509
8.2.2.2.1 Embedded Trace Buffer (ETB).....509
8.2.2.2.2 AMBA Trace Bus (ATB) Replicator.....509
8.2.2.2.3 Cross Trigger Interface 1 (CTI1).....510
8.2.2.2.4 Cross Trigger Matrix - CTM.....510
8.2.2.2.5 Advanced Peripheral Bus - APB Debug Bus.....510
8.2.2.3 Asynchronous Wrapper.....510
8.2.3 Configuration.....511
8.2.4 Endian Modes.....511
8.2.5 Bus Interfaces.....511
8.2.5.1 AMBA AXI Interface.....511
8.2.5.2 APB CoreSight Interface.....511
8.2.5.3 ATB CoreSight Interface.....512
8.2.5.4 Peripheral Interface (IP Bus).....512
8.3 Memory Map and Register Definition.....512
8.3.1 Platform Version ID (ARM_PVID).....513
8.3.2 General Purpose Control (ARM_GPC).....514
8.3.3 Low Power Control (ARM_LPC).....515
8.3.4 NEON Low Power Control (ARM_NLPC).....516
8.3.5 Internal Clock Generation Control (ARM_ICGC).....517
8.3.6 ARM Memory Configuration (ARM_AMC).....519
8.3.7 NEON Monitor Control (ARM_NMC).....520
8.3.8 NEON Monitor Status (ARM_NMS).....521
8.4 Platform Clocks.....521

8.5 Platform Power Management....522
8.5.1 Voltage and Frequency Scaling....522
8.5.1.1 Asynchronous Interface Logic....522
8.5.1.2 Level Shifting between SoC-logic and ARM-logic....523
8.5.2 Power Gating in the ARM platform....523
8.5.2.1 Isolation Circuitry at the ARM platform Interface....523
8.5.2.2 Power Gating the Memory Peripheries....523
8.5.2.3 Retaining the State of the ARM platform Registers....523
8.5.2.4 Power Gating the ARM platform High Performance Logic Gates....524
8.5.2.5 Power Gating the L2 Bit Arrays....524
8.5.2.6 Controlling Power Gating using the General Power Controller (GPC) 524
8.5.2.7 Power Gating the NEON Block....525
8.5.3 Modes of Operation....529

Chapter 9
ARM Platform Debug

9.1 Introduction 531
9.1.1 Overview....531
9.1.2 ARM Debug Blocks....532
9.1.2.1 Processor Debug Unit....533
9.1.2.1.1 Halting Debug-Mode Debugging....533
9.1.2.1.2 Monitor Debug-Mode Debugging....533
9.1.2.1.3 Programming the Debug Unit....533
9.1.2.2 ARM embedded trace macrocell (ARM ETM)....534
9.1.2.3 CoreSight Embedded Trace Buffer (CSETB)....535
9.1.2.4 CoreSight Replicator (CSREPLICATOR)....536
9.1.2.5 CoreSight trace port interface unit (CSTPIU)....537
9.1.2.6 CoreSight cross trigger interface (CSCTI)....537
9.1.2.7 CoreSight Cross Trigger Matrix (CSCTM)....539

9.1.2.8 Debug Access Port (DAP)....540
9.1.2.8.1 DAP_SYS....541
9.1.3 Modes of Operation....542
9.1.3.1 ARM Invasive Debug Mode....542
9.1.3.2 ARM Non-Invasive Debug Mode (Real-time Trace)....542
9.1.3.3 Normal Operating Modes....543
9.1.3.4 Low Power Modes....543
9.2 Memory Map and Register Definition....543
9.2.1 Register Summary....544
9.2.1.1 DAP JTAG DP Registers....545
9.2.1.2 DAP ROM Register Summary....546
9.2.1.3 Processor Debug Unit Register Summary....546
9.2.1.3.1 Coprocessor Registers Summary....546
9.2.1.3.2 Memory Mapped Registers Summary....547
9.2.1.4 ETB Register Summary....548
9.2.1.5 ETM Register Summary....550
9.2.1.6 CSCTI Register Summary....552
9.2.1.7 TPIU Register Summary....554
9.2.2 Clocks....557
9.2.3 Reset....557

Chapter 10
Multi-Layer AHB Crossbar Switch (AHBMAX)

10.1 Overview....559
10.2 Features....561
10.2.1 Limitations....561
10.2.2 General Operation....561
10.3 AHBMAX Interface Signals....563
10.3.1 AHBMAX Signal Descriptions....563
10.3.1.1 max_halt_request....563

10.3.1.2 max_halted....563
10.4 Coherency....563
10.5 Detailed Functional Description....564
10.5.1 Arbitration....564
10.5.1.1 Arbitration During Undefined Length Bursts....564
10.5.1.2 Fixed Priority Operation....565
10.5.1.3 Round-Robin Priority Operation....565
10.5.2 Priority Assignment....566
10.5.3 Master Port Functionality....566
10.5.3.1 Master Port General Information....566
10.5.3.2 Master Port Decoders....569
10.5.3.3 Master Port Capture Unit....569
10.5.3.4 Master Port Registers....569
10.5.3.5 Master Port State Machine....569
10.5.3.5.1 Master Port State Machine States....569
10.5.3.5.2 Master Port State Machine Slave Swapping....570
10.5.4 Slave Port Functionality....571
10.5.4.1 Slave Port General Information....571
10.5.4.2 Slave Port Muxes....572
10.5.4.3 Slave Port Registers....573
10.5.4.4 Slave Port State Machine....573
10.5.4.4.1 Slave Port State Machine States....573
10.5.4.4.2 Slave Port State Machine Arbitration....574
10.5.4.4.3 Slave Port State Machine Master Handoff....574
10.5.4.4.4 Slave Port State Machine Parking....577
10.5.4.4.5 Slave Port State Machine Halt Mode....580
10.6 Initialization/Application Information....581
10.7 AHBMAX Interface....581
10.7.1 AHBMAX Interface Overview....581

10.7.2 Master Ports-AHBMAX Interface....581
10.7.2.1 Terminated Accesses....581
10.7.2.2 Taken Accesses....581
10.7.2.3 Stalled Accesses....582
10.7.2.4 Error Response Terminated Accesses....582
10.7.3 Slave Ports-AHBMAX Interface....582
10.8 Programmable Registers....583
10.8.1 Master Priority Register for Slave port n (AHBMAX_MPR*n*)....584
10.8.2 General Purpose Control Register for Slave port n (AHBMAX_SGPCR*n*)....586
10.8.3 General Purpose Control Register for Master port n (AHBMAX_MGPCR*n*)....588

Chapter 11
AHB-to-APBH Bridge with DMA (APBH-Bridge-DMA)

11.1 Overview....591
11.2 APBH DMA....593
11.3 NAND Read Status Polling Example....598
11.4 Programmable Registers....600
11.4.1 AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0*n*)....607
11.4.2 AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1*n*)....608
11.4.3 AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2*n*)....612
11.4.4 AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL*n*)....617
11.4.5 AHB to APBH DMA Device Assignment Register (APBH_DEVSEL)....618
11.4.6 AHB to APBH DMA burst size (APBH_DMA_BURST_SIZE)....619
11.4.7 AHB to APBH DMA Debug Register (APBH_DEBUG)....620
11.4.8 APBH DMA Channel n Current Command Address Register (APBH_CH*n*_CURCMDAR)....621
11.4.9 APBH DMA Channel n Next Command Address Register (APBH_CH*n*_NXTCMDAR)....622
11.4.10 APBH DMA Channel n Command Register (APBH_CH*n*_CMD)....623
11.4.11 APBH DMA Channel n Buffer Address Register (APBH_CH*n*_BAR)....625
11.4.12 APBH DMA Channel n Semaphore Register (APBH_CH*n*_SEMA)....626
11.4.13 AHB to APBH DMA Channel n Debug Information (APBH_CH*n*_DEBUG1)....627

11.4.14 AHB to APBH DMA Channel n Debug Information (APBH_CH*n*_DEBUG2).....630
11.4.15 APBH Bridge Version Register (APBH_VERSION).....630

Chapter 12
Digital Audio Multiplexer (AUDMUX)

12.1 Overview.....633
12.1.1 Features.....635
12.1.2 Modes and Operations.....635
12.2 External Signal Description.....635
12.3 Default Register Configuration.....636
12.3.1 Default Port Configuration.....636
12.4 Functional Description.....636
12.4.1 Operating Modes.....636
12.4.1.1 Port Receive Data Modes.....637
12.4.1.1.1 Normal Mode.....638
12.4.1.1.2 Internal Network Mode.....639
12.4.1.1.3 Transmit Data Output Enable Assertion.....646
12.4.1.2 Tx/Rx Switch and External Network Mode.....647
12.4.1.3 Timing Modes.....648
12.4.1.3.1 Synchronous Mode (4-Wire Interface).....648
12.4.1.3.2 Asynchronous Mode (6-Wire Interface).....650
12.4.2 Connectivity Between Ports.....653
12.4.2.1 Internal Port to External Port Connectivity.....654
12.4.2.2 External Port to External Port Connectivity.....655
12.4.2.3 Internal Port to Internal Port Connectivity.....655
12.4.2.4 Loopback Connectivity.....656
12.4.3 AUDMUX Clocking.....656
12.4.3.1 AUDMUX Clock Inputs.....656
12.4.3.2 AUDMUX Clock Diagram.....656
12.4.3.3 Clocking Restrictions.....657

12.5 Programmable Registers....657
12.5.1 Port Timing Control Register 1 (AUDMUX_PTCR1)....658
12.5.2 Port Data Control Register 1 (AUDMUX_PDCR1)....660
12.5.3 Port Timing Control Register 2 (AUDMUX_PTCR2)....661
12.5.4 Port Data Control Register 2 (AUDMUX_PDCR2)....663
12.5.5 Port Timing Control Register 3 (AUDMUX_PTCR3)....664
12.5.6 Port Data Control Register 3 (AUDMUX_PDCR3)....666
12.5.7 Port Timing Control Register n (AUDMUX_PTCR*n*)....667
12.5.8 Port Data Control Register 4 (AUDMUX_PDCR4)....670
12.5.9 Port Data Control Register 5 (AUDMUX_PDCR5)....671
12.5.10 Port Data Control Register 6 (AUDMUX_PDCR6)....672

Chapter 13
AHB to IP Bridge (AIPSTZ)

13.1 Introduction....675
13.1.1 Features....675
13.2 General Operation....676
13.2.1 AIPSTZ Registers....676
13.2.2 Overview....676
13.2.3 Control Registers....677
13.3 Register Descriptions....678
13.3.1 Master Privilege Registers....678
13.3.2 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACRs)....679
13.4 Functional Description....680
13.5 Access Protections....680
13.6 Access Support....680
13.7 Initialization Information....681

Chapter 14
32-Bit Correcting ECC Accelerator (BCH)

14.1 Introduction....683

14.2 Overview....683
14.3 Operation....686
14.3.1 BCH Limitations and Assumptions....687
14.3.2 Flash Page Layout....687
14.3.3 Determining the ECC layout for a Device....689
14.3.3.1 4K + 218 Flash, 10 bytes Metadata, 512 byte Data Blocks, Separate Metadata, Assuming GF(2^13)....689
14.3.3.2 4K + 128 flash, 10 bytes Metadata, 1024 byte Data Blocks, Separate Metadata, Assuming GF(2^14) for Data and GF(2^13) for Metadata....690
14.3.4 Data buffers in System Memory....690
14.4 Memory to Memory (Loopback) Operation....693
14.5 Programming the BCH/GPMI Interfaces....694
14.5.1 BCH Encoding for NAND Writes....694
14.5.1.1 DMA Structure Code Example 1....698
14.5.1.2 Using the BCH Encoder....702
14.5.2 BCH Decoding for NAND Reads....703
14.5.2.1 DMA Structure Code Example 2....707
14.5.2.2 Using the Decoder....710
14.5.3 Interrupts....712
14.6 Behavior During Reset....713
14.7 Programmable Registers....713
14.7.1 Hardware BCH ECC Accelerator Control Register (BCH_CTRL*n*)....715
14.7.2 Hardware ECC Accelerator Status Register 0 (BCH_STATUS0)....717
14.7.3 Hardware ECC Accelerator Mode Register (BCH_MODE)....719
14.7.4 Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR)....719
14.7.5 Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR)....720
14.7.6 Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR)....720
14.7.7 Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT)....721
14.7.8 Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0)....722

14.7.9 Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1).....724
14.7.10 Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0).....725
14.7.11 Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1).....727
14.7.12 Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0).....728
14.7.13 Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1).....730
14.7.14 Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0).....731
14.7.15 Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1).....733
14.7.16 Hardware BCH ECC Debug Register0 (BCH_DEBUG0*n*).....735
14.7.17 KES Debug Read Register (BCH_DBGKESREAD).....736
14.7.18 Chien Search Debug Read Register (BCH_DBGCSFEREAD).....737
14.7.19 Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD).....737
14.7.20 Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD).....738
14.7.21 Block Name Register (BCH_BLOCKNAME).....738
14.7.22 BCH Version Register (BCH_VERSION).....739

Chapter 15
Clock Amplifier (CAMP)

15.1 Introduction.....741
15.1.1 Overview.....742
15.1.2 Features.....742
15.1.3 Modes of Operation.....742
15.1.3.1 Normal Mode.....742
15.1.3.2 Power Down Mode.....742
15.1.3.3 Test Mode (Fault Bypass or Scan).....742
15.2 External Signal Description.....742
15.2.1 Signals Overview.....743
15.2.2 Detailed Signal Description.....743
15.2.2.1 CKIH - External Clock Input.....743
15.2.2.2 VDD - Power supply.....743
15.2.2.3 VSS - Ground.....743

15.2.2.4 IPT_SCAN_MODE - Scan Signal....743
15.2.2.5 PWD - Power Down Signal....744
15.2.2.6 FAULT_BYP - Control Signal for Test Mode....744
15.2.2.7 CAMP_OUT - Clock Output from CAMP....744
15.3 Memory Map/Register Definition....744
15.4 Functional Description....744
15.4.1 CAMP Sub-Blocks....744
15.4.1.1 Main Clock Amplifier....745
15.4.1.2 Output Buffer....745
15.4.1.3 Level Shifter....745
15.4.2 CAMP Modes of Operation....745
15.4.2.1 Normal Mode....745
15.4.2.2 Power Down Mode....745
15.4.2.3 Test Mode....746
15.5 Initialization/Application Information....746

Chapter 16
Configurable Serial Peripheral Interface (CSPI)

16.1 Overview....747
16.1.1 Features....748
16.1.2 Modes and Operations....748
16.2 External Signals....749
16.3 Functional Description....750
16.3.1 Operating Modes....751
16.3.1.1 Master Mode....751
16.3.1.2 Slave Mode....752
16.3.2 Low Power Modes....752
16.3.3 Operations....752
16.3.3.1 Typical Master Mode....752
16.3.3.1.1 Master Mode with SPI_RDY....753

16.3.3.1.2 Master Mode with Wait States....755
16.3.3.1.3 Master Mode with SSCTL Control....755
16.3.3.1.4 Master Mode with Phase Control....757
16.3.3.2 Typical Slave Mode....757
16.3.4 Clocks....758
16.3.5 Reset....759
16.3.6 Interrupts....759
16.3.7 DMA....760
16.3.8 Byte Order....761
16.4 Initialization....761
16.5 Applications....762
16.6 Programmable Registers....764
16.6.1 Receive Data Register (CSPI_RXDATA)....764
16.6.2 Transmit Data Register (CSPI_TXDATA)....765
16.6.3 Control Register (CSPI_CONREG)....766
16.6.4 Interrupt Control Register (CSPI_CSPI_INTREG)....769
16.6.5 DMA Control Register (CSPI_CSPI_DMAREG)....770
16.6.6 Status Register (CSPI_CSPI_STATREG)....771
16.6.7 Sample Period Control Register (CSPI_PERIODREG)....772
16.6.8 Test Control Register (CSPI_TESTREG)....773

Chapter 17
Data Co-Processor (DCP)

17.1 Overview....775
17.1.1 DCP Limitations for Software....777
17.2 Operation....778
17.2.1 Memory Copy, Blit, and Fill Functionality....778
17.2.2 Advanced Encryption Standard (AES)....779
17.2.2.1 Key Storage....780
17.2.2.2 AES OTP Key....780

17.2.2.3 Encryption Modes....780
17.2.3 Hashing....782
17.2.4 One Time Programmable (OTP) Key....782
17.2.5 Managing DCP Channel Arbitration and Performance....783
17.2.5.1 DCP Arbitration....783
17.2.5.2 Channel Recovery Timers....784
17.2.6 Programming Channel Operations....784
17.2.6.1 Virtual Channels....785
17.2.6.2 Context Switching....786
17.2.6.3 Working with Semaphores....787
17.2.6.4 Work Packet Structure....788
17.2.6.4.1 Next Command Address Field....788
17.2.6.4.2 Control0 Field....789
17.2.6.4.3 Control1 Field....791
17.2.6.4.4 Source Buffer....791
17.2.6.4.5 Destination Buffer....792
17.2.6.4.6 Buffer Size Field....792
17.2.6.4.7 Payload Pointer....792
17.2.6.4.8 Status....793
17.2.6.4.9 Payload....793
17.2.7 Programming DCP Functions....794
17.2.7.1 Basic Memory Copy Programming Example....795
17.2.7.2 Basic Hash Operation Programming Example....796
17.2.7.3 Basic Cipher Operation Programming Example....797
17.2.7.4 Multi-Buffer Scatter/Gather Cipher and Hash Operation Programming Example....799
17.3 Programmable Registers....802
17.3.1 DCP Control Register 0 (DCP_CTRL)....804
17.3.2 DCP Status Register (DCP_STAT)....806
17.3.3 DCP Channel Control Register (DCP_CHANNELCTRL)....808

17.3.4 DCP Capability 0 Register (DCP_CAPABILITY0)....809
17.3.5 DCP Capability 1 Register (DCP_CAPABILITY1)....810
17.3.6 DCP Context Buffer Pointer (DCP_CONTEXT)....811
17.3.7 DCP Key Index (DCP_KEY)....811
17.3.8 DCP Key Data (DCP_KEYDATA)....812
17.3.9 DCP Work Packet 0 Status Register (DCP_PACKET0)....813
17.3.10 DCP Work Packet 1 Status Register (DCP_PACKET1)....813
17.3.11 DCP Work Packet 2 Status Register (DCP_PACKET2)....816
17.3.12 DCP Work Packet 3 Status Register (DCP_PACKET3)....817
17.3.13 DCP Work Packet 4 Status Register (DCP_PACKET4)....817
17.3.14 DCP Work Packet 5 Status Register (DCP_PACKET5)....818
17.3.15 DCP Work Packet 6 Status Register (DCP_PACKET6)....818
17.3.16 DCP Channel 0 Command Pointer Address Register (DCP_CH0CMDPTR)....819
17.3.17 DCP Channel 0 Semaphore Register (DCP_CH0SEMA)....820
17.3.18 DCP Channel 0 Status Register (DCP_CH0STAT)....821
17.3.19 DCP Channel 0 Options Register (DCP_CH0OPTS)....823
17.3.20 DCP Channel 1 Command Pointer Address Register (DCP_CH1CMDPTR)....824
17.3.21 DCP Channel 1 Semaphore Register (DCP_CH1SEMA)....825
17.3.22 DCP Channel 1 Status Register (DCP_CH1STAT)....826
17.3.23 DCP Channel 1 Options Register (DCP_CH1OPTS)....827
17.3.24 DCP Channel 2 Command Pointer Address Register (DCP_CH2CMDPTR)....828
17.3.25 DCP Channel 2 Semaphore Register (DCP_CH2SEMA)....829
17.3.26 DCP Channel 2 Status Register (DCP_CH2STAT)....830
17.3.27 DCP Channel 2 Options Register (DCP_CH2OPTS)....832
17.3.28 DCP Channel 3 Command Pointer Address Register (DCP_CH3CMDPTR)....833
17.3.29 DCP Channel 3 Semaphore Register (DCP_CH3SEMA)....834
17.3.30 DCP Channel 3 Status Register (DCP_CH3STAT)....835
17.3.31 DCP Channel 3 Options Register (DCP_CH3OPTS)....836
17.3.32 DCP Debug Select Register (DCP_DBGSELECT)....837

17.3.33 DCP Debug Data Register (DCP_DBGDATA).....838
17.3.34 DCP Page Table Register (DCP_PAGETABLE).....838
17.3.35 DCP Version Register (DCP_VERSION).....839

Chapter 18
Digital Control (DIGCTL) and On-Chip RAM

18.1 Overview.....841
18.2 eSDHC Version control.....842
18.3 OCRAM Pipeline Controls.....842
18.4 Speed Sensor Controls.....842
18.5 Programmable Registers.....843
18.5.1 DIGCTL Control Register (DIGCTL_CTRL*n*).....844
18.5.2 DIGCTL OCRAM Register (DIGCTL_OCRAM*n*).....845
18.5.3 Transistor Speed Control Register (DIGCTL_SPEEDCTL*n*).....847
18.5.4 Transistor Speed Status Register (DIGCTL_DIGCTL_SPEEDSTAT).....847

Chapter 19
DPLL Controller (DPLLC)

19.1 Overview.....849
19.1.1 Feature Description.....850
19.1.2 Modes and Operation.....850
19.2 Functional Description.....850
19.2.1 Operating Modes.....851
19.2.1.1 Normal Mode.....851
19.2.1.2 DPLL Desense Mode.....851
19.2.1.3 DVFS Support: HFS Mode.....853
19.2.2 Operations.....853
19.2.2.1 DPLLC_DP_CTL, DPLLC_DP_OP, DPLLC_DP_MFD Register Update.....853
19.2.2.2 DP_MFN Register Update.....854
19.2.2.3 Multiple Options for DPLL Control.....855
19.2.2.4 Calculating the Output Frequency.....856

19.3 Programmable Registers....856
19.3.1 DPLLC Memory Map/Register Definition....856
19.31.1 DPLLC Control Register (DPLLC*x*_CTL)....859
19.31.2 DPLLC Configuration Register (DPLLC*x*_CONFIG)....861
19.31.3 DPLLC Operation Register (DPLLC*x*_OP)....862
19.31.4 DPLLC Multiplication Factor Denominator Register (DPLLC*x*_MFD)....863
19.31.5 DPLLC Multiplication Factor Numerator Register (DPLLC*x*_MFN)....863
19.31.6 DPLLC Multiplication Factor Numerator PLUS/MINUS Registers (DPLLC*x*_MFN*n*)....864
19.31.7 DPLLC High Frequency Support, Operation Register (DPLLC*x*_HFS_OP)....865
19.31.8 DPLLC High Frequency Support Multiplication Factor Denominator Register (DPLLC*x*_HFS_MFD)....866
19.31.9 DPLLC High Frequency Support Multiplication Factor Numerator Register (DPLLC*x*_HFS_MFN)....867
19.31.10 DPLLC Multiplication Factor Numerator Toggle Control Register (DPLLC*x*_MFN_TOGC)....867
19.31.11 DPLLC Desense Status Register (DPLLC*x*_DESTAT)....869

Chapter 20
DRAM Memory Controller (DRAM MC)

20.1 Overview....871
20.1.1 Block Diagram....871
20.1.2 DRAM MC Special Signal Considerations....872
20.2 DRAM MC Address Mapping....873
20.2.1 DDR SDRAM Address Mapping Options....873
20.2.2 Maximum Address Space....874
20.2.3 Memory Mapping to Address Space....874
20.2.4 Out-of-Range Address Checking....875
20.3 DRAM MC Clocking....875
20.3.1 Changing the Input Clock Frequency....875
20.4 DRAM MC AHB and AXI Interface....876
20.4.1 AHB Register Port....876

20.4.2 AXI Interface....877
20.4.2.1 Internal Command Handling....877
20.4.2.2 Configured Options....878
20.4.3 Port Clocking....880
20.4.4 AXI Interface FIFOs....881
20.4.4.1 In-Port Arbitration....882
20.4.5 AXI Write Response Channel....884
20.4.5.1 Bufferable, Coherent Bufferable and Non-Bufferable Response Types....884
20.4.6 AXI Transactions....885
20.4.7 Exclusive Access....885
20.4.7.1 Initiating an Exclusive Access Request....885
20.4.7.2 Verifying the Request....885
20.4.7.3 Processing Exclusive Accesses....886
20.4.7.4 Validity of an Exclusive Request....887
20.4.7.5 Read Commands to Exclusive Access Regions....887
20.4.7.6 Non-Exclusive Write Commands to Exclusive Access Regions....888
20.4.7.7 Exclusive Writes....888
20.4.7.8 Passing Exclusive Access Check....888
20.4.7.9 Failing Exclusive Access Check....888
20.4.8 Error Responses....889
20.4.8.1 AXI Error Response....889
20.4.8.2 AXI Error Reporting....889
20.4.9 Arbiter....889
20.4.10 Write Data Queue....890
20.4.11 DRAM Command Processing....890
20.4.12 Latency....890
20.5 DRAM MC Core-Logic....891
20.5.1 Command Queue with Placement Logic....892

20.6 Core Command Queue with Placement Logic....893
20.6.1 Rules of the Placement Algorithm....893
20.6.1.1 Address Collision/Data Coherency Violation....893
20.6.1.2 Source ID Collision....894
20.6.1.3 Write Buffer Collision....894
20.6.1.4 Priority....894
20.6.1.5 Bank Splitting....895
20.6.1.6 Read/Write Grouping....895
20.6.2 Command Execution Order After Placement....896
20.6.2.1 Command Aging....896
20.6.2.2 High-Priority Command Swapping....896
20.7 Notes for Programming of the Timing Parameters....898
20.7.1 Command to Command Timing....898
20.7.2 Refresh Per Chip-select....899
20.7.3 Programming of the tref Parameter....899
20.7.4 Programming of the tref_interval Parameter....900
20.8 DRAM Low Power Operation....900
20.8.1 Low Power Modes....900
20.8.2 Low Power Mode Control....902
20.8.3 Automatic Entry....902
20.8.4 Automatic Exit....903
20.8.5 Manual "On-Demand" Entry....903
20.8.6 Manual "On-Demand" Exit....904
20.8.7 Register Programming....904
20.9 Interface for LPDDR2 Devices....906
20.9.1 Restrictions....906
20.9.2 ZQ Calibration for off-chip LPDDR2 Devices....906
20.9.3 Programmable Control....907
20.9.4 ZQ Short Calibration....908

20.9.5 ZQ Long Calibration....909
20.9.6 ZQ Initialization....909
20.9.7 ZQ Reset....909
20.10 DDR PHY....910
20.10.1 High Level Block Diagram....910
20.10.2 DFI....911
20.10.3 The I/O Timing of Address & Command....911
20.10.4 The Address Timing of LP-DDR2....912
20.10.5 Data Slice Overview....913
20.10.6 Read Data Capture....914
20.10.7 Synchronize Read Data From delayed_dqs To PHY_CLK domain....914
20.10.8 Write Data Path....916
20.10.8.1 The Low-Latency Option of Write Data Path....917
20.10.9 Digital DLL and the Delay-Line....918
20.10.10 Configure the "output enable" of I/O Control....920
20.11 Programmable Registers....921
20.11.1 DRAM CTL Register 00 (DRAM_CTL00)....928
20.11.2 DRAM CTL Register 01 (DRAM_CTL01)....929
20.11.3 DRAM CTL Register 02 (DRAM_CTL02)....929
20.11.4 DRAM CTL Register 03 (DRAM_CTL03)....930
20.11.5 DRAM CTL Register 04 (DRAM_CTL04)....930
20.11.6 DRAM CTL Register 05 (DRAM_CTL05)....931
20.11.7 DRAM CTL Register 06 (DRAM_CTL06)....932
20.11.8 DRAM CTL Register 07 (DRAM_CTL07)....933
20.11.9 DRAM CTL Register 08 (DRAM_CTL08)....934
20.11.10 DRAM CTL Register 09 (DRAM_CTL09)....935
20.11.11 DRAM CTL Register 10 (DRAM_CTL10)....937
20.11.12 DRAM CTL Register 11 (DRAM_CTL11)....938
20.11.13 DRAM CTL Register 12 (DRAM_CTL12)....939

20.11.14 DRAM CTL Register 13 (DRAM_CTL13).....940
20.11.15 DRAM CTL Register 14 (DRAM_CTL14).....941
20.11.16 DRAM CTL Register 15 (DRAM_CTL15).....942
20.11.17 DRAM CTL Register 16 (DRAM_CTL16).....943
20.11.18 DRAM CTL Register 17 (DRAM_CTL17).....944
20.11.19 DRAM CTL Register 18 (DRAM_CTL18).....945
20.11.20 DRAM CTL Register 19 (DRAM_CTL19).....946
20.11.21 DRAM CTL Register 20 (DRAM_CTL20).....947
20.11.22 DRAM CTL Register 21 (DRAM_CTL21).....948
20.11.23 DRAM CTL Register 22 (DRAM_CTL22).....949
20.11.24 DRAM CTL Register 23 (DRAM_CTL23).....950
20.11.25 DRAM CTL Register 24 (DRAM_CTL24).....951
20.11.26 DRAM CTL Register 25 (DRAM_CTL25).....952
20.11.27 DRAM CTL Register 26 (DRAM_CTL26).....952
20.11.28 DRAM CTL Register 27 (DRAM_CTL27).....953
20.11.29 DRAM CTL Register 28 (DRAM_CTL28).....954
20.11.30 DRAM CTL Register 29 (DRAM_CTL29).....955
20.11.31 DRAM CTL Register 30 (DRAM_CTL30).....955
20.11.32 DRAM CTL Register 31 (DRAM_CTL31).....956
20.11.33 DRAM CTL Register 32 (DRAM_CTL32).....956
20.11.34 DRAM CTL Register 33 (DRAM_CTL33).....957
20.11.35 DRAM CTL Register 34 (DRAM_CTL34).....958
20.11.36 DRAM CTL Register 35 (DRAM_CTL35).....959
20.11.37 DRAM CTL Register 36 (DRAM_CTL36).....960
20.11.38 DRAM CTL Register 37 (DRAM_CTL37).....961
20.11.39 DRAM CTL Register 38 (DRAM_CTL38).....963
20.11.40 DRAM CTL Register 39 (DRAM_CTL39).....964
20.11.41 DRAM CTL Register 40 (DRAM_CTL40).....965
20.11.42 DRAM CTL Register 41 (DRAM_CTL41).....967

20.11.43 DRAM CTL Register 42 (DRAM_CTL42)....968
20.11.44 DRAM CTL Register 43 (DRAM_CTL43)....969
20.11.45 DRAM CTL Register 44 (DRAM_CTL44)....969
20.11.46 DRAM CTL Register 45 (DRAM_CTL45)....970
20.11.47 DRAM CTL Register 46 (DRAM_CTL46)....970
20.11.48 DRAM CTL Register 47 (DRAM_CTL47)....971
20.11.49 DRAM CTL Register 48 (DRAM_CTL48)....971
20.11.50 DRAM CTL Register 49 (DRAM_CTL49)....972
20.11.51 DRAM CTL Register 50 (DRAM_CTL50)....973
20.11.52 DRAM CTL Register 51 (DRAM_CTL51)....974
20.11.53 DRAM CTL Register 52 (DRAM_CTL52)....975
20.11.54 DRAM CTL Register 53 (DRAM_CTL53)....976
20.11.55 DRAM CTL Register 54 (DRAM_CTL54)....977
20.11.56 DRAM CTL Register 55 (DRAM_CTL55)....977
20.11.57 DRAM CTL Register 56 (DRAM_CTL56)....978
20.11.58 DRAM CTL Register 57 (DRAM_CTL57)....979
20.11.59 DRAM CTL Register 58 (DRAM_CTL58)....980
20.11.60 DRAM CTL Register 59 (DRAM_CTL59)....981
20.11.61 DRAM CTL Register 60 (DRAM_CTL60)....982
20.11.62 DRAM CTL Register 61 (DRAM_CTL61)....983
20.11.63 DRAM CTL Register 62 (DRAM_CTL62)....984
20.11.64 DRAM CTL Register 63 (DRAM_CTL63)....985
20.11.65 DRAM CTL Register 64 (DRAM_CTL64)....986
20.11.66 DRAM CTL Register 65 (DRAM_CTL65)....987
20.11.67 DRAM CTL Register 66 (DRAM_CTL66)....988
20.11.68 DRAM CTL Register 67 (DRAM_CTL67)....988
20.11.69 DRAM CTL Register 68 (DRAM_CTL68)....989
20.11.70 DRAM CTL Register 69 (DRAM_CTL69)....989
20.11.71 DRAM CTL Register 70 (DRAM_CTL70)....990

20.11.72 DRAM CTL Register 71 (DRAM_CTL71)....991
20.11.73 DRAM CTL Register 72 (DRAM_CTL72)....992
20.11.74 DRAM CTL Register 73 (DRAM_CTL73)....993
20.11.75 DRAM CTL Register 74 (DRAM_CTL74)....994
20.11.76 DRAM CTL Register 75 (DRAM_CTL75)....995
20.11.77 DRAM CTL Register 76 (DRAM_CTL76)....995
20.11.78 DRAM CTL Register 77 (DRAM_CTL77)....996
20.11.79 DRAM CTL Register 78 (DRAM_CTL78)....996
20.11.80 DRAM CTL Register 79 (DRAM_CTL79)....997
20.11.81 DRAM CTL Register 80 (DRAM_CTL80)....998
20.11.82 DRAM CTL Register 81 (DRAM_CTL81)....998
20.11.83 DRAM CTL Register 82 (DRAM_CTL82)....998
20.11.84 DRAM CTL Register 83 (DRAM_CTL83)....999
20.11.85 DRAM CTL Register 84 (DRAM_CTL84)....999
20.11.86 DRAM CTL Register 85 (DRAM_CTL85)....1000
20.11.87 DRAM CTL Register 86 (DRAM_CTL86)....1000
20.11.88 DRAM PHY Register 00 (DRAM_PHY00)....1001
20.11.89 DRAM PHY Register 01 (DRAM_PHY01)....1001
20.11.90 DRAM PHY Register 02 (DRAM_PHY02)....1002
20.11.91 DRAM PHY Register 03 (DRAM_PHY03)....1004
20.11.92 DRAM PHY Register 04 (DRAM_PHY04)....1006
20.11.93 DRAM PHY Register 05 (DRAM_PHY05)....1006
20.11.94 DRAM PHY Register 06 (DRAM_PHY06)....1006
20.11.95 DRAM PHY Register 07 (DRAM_PHY07)....1007
20.11.96 DRAM PHY Register 08 (DRAM_PHY08)....1007
20.11.97 DRAM PHY Register 09 (DRAM_PHY09)....1008
20.11.98 DRAM PHY Register 10 (DRAM_PHY10)....1008
20.11.99 DRAM PHY Register 11 (DRAM_PHY11)....1008
20.11.100 DRAM PHY Register 12 (DRAM_PHY12)....1009

20.11.101 DRAM PHY Register 13 (DRAM_PHY13)....1009
20.11.102 DRAM PHY Register 14 (DRAM_PHY14)....1010
20.11.103 DRAM PHY Register 15 (DRAM_PHY15)....1011
20.11.104 DRAM PHY Register 16 (DRAM_PHY16)....1012
20.11.105 DRAM PHY Register 17 (DRAM_PHY17)....1012
20.11.106 DRAM PHY Register 18 (DRAM_PHY18)....1013
20.11.107 DRAM PHY Register 19 (DRAM_PHY19)....1013
20.11.108 DRAM PHY Register 20 (DRAM_PHY20)....1014
20.11.109 DRAM PHY Register 21 (DRAM_PHY21)....1014
20.11.110 DRAM PHY Register 22 (DRAM_PHY22)....1014
20.11.111 DRAM PHY Register 23 (DRAM_PHY23)....1015
20.11.112 DRAM PHY Register 24 (DRAM_PHY24)....1015
20.11.113 DRAM PHY Register 25 (DRAM_PHY25)....1016
20.11.114 DRAM PHY Register 26 (DRAM_PHY26)....1017
20.11.115 DRAM PHY Register 27 (DRAM_PHY27)....1017
20.11.116 DRAM PHY Register 28 (DRAM_PHY28)....1018
20.11.117 DRAM PHY Register 29 (DRAM_PHY29)....1018
20.11.118 DRAM PHY Register 30 (DRAM_PHY30)....1018
20.11.119 DRAM PHY Register 31 (DRAM_PHY31)....1019
20.11.120 DRAM PHY Register 32 (DRAM_PHY32)....1019
20.11.121 DRAM PHY Register 33 (DRAM_PHY33)....1020
20.11.122 DRAM PHY Register 34 (DRAM_PHY34)....1020
20.11.123 DRAM PHY Register 35 (DRAM_PHY35)....1020
20.11.124 DRAM PHY Register 36 (DRAM_PHY36)....1021
20.11.125 DRAM PHY Register 37 (DRAM_PHY37)....1021
20.11.126 DRAM PHY Register 38 (DRAM_PHY38)....1022

Chapter 21
Dynamic Voltage and Frequency Scaling Core (DVFSC)

21.1 Introduction 1023
21.1.1 Overview....1023
21.1.2 Features....1024
21.2 Functional Description of DVFS Load Tracking....1025
21.3 Component Blocks Description....1025
21.3.1 dvfs_stdb_smpl Block....1025
21.3.2 dvfs_sig_wt Block....1025
21.3.3 dvfs_pre_avg Block....1026
21.3.4 dvfs_ld_add Block....1029
21.3.5 dvfs_ema_avg Block....1029
21.3.6 dvfs_thres_cmp Block....1033
21.3.7 dvfs_thresh_count Block....1034
21.3.8 Load Tracking Buffer Register....1035
21.3.9 Frequency Pattern Generator....1036
21.4 DVFS Output Event/interrupt Configuration....1037
21.4.1 Interrupts....1037
21.5 Initialization Information....1037
21.6 Programmable Registers....1038
21.6.1 DVFSC Thresholds (DVFSC_THRS)....1039
21.6.2 DVFSC Counters thresholds (DVFSC_COUN)....1039
21.6.3 DVFSC general purpose bits weight (DVFSC_SIG1)....1040
21.6.4 DVFSC general purpose bits weight (DVFSC_SIG0)....1041
21.6.5 DVFSC general purpose bit 0 weight counter (DVFSC_GPC0)....1042
21.6.6 DVFSC general purpose bit 1 weight counter (DVFSC_GPC1)....1043
21.6.7 DVFSC general purpose bits enables (DVFSC_GPBT)....1044
21.6.8 DVFSC EMAC settings (DVFSC_EMAC)....1045
21.6.9 DVFSC Control (DVFSC_CNTR)....1045

21.6.10 DVFSC Load Tracking Register 0, portion 0 (DVFSC_LTR0_0)....1048
21.6.11 DVFSC Load Tracking Register 0, portion 1 (DVFSC_LTR0_1)....1049
21.6.12 DVFSC Load Tracking Register 1, portion 0 (DVFSC_LTR1_0)....1050
21.6.13 DVFS Load Tracking Register 3, portion 1 (DVFSC_LTR1_1)....1051
21.6.14 DVFSC pattern 0 length (DVFSC_PT0)....1051
21.6.15 DVFSC pattern 1 length (DVFSC_PT1)....1052
21.6.16 DVFSC pattern 2 length (DVFSC_PT2)....1053
21.6.17 DVFSC pattern 3 length (DVFSC_PT3)....1053

Chapter 22
Enhanced Configurable SPI (ECSPI)

22.1 Overview....1055
22.1.1 Features....1056
22.1.2 Modes and Operations....1056
22.2 External Signals....1057
22.3 Functional Description....1058
22.3.1 Master Mode....1059
22.3.2 Slave Mode....1059
22.3.3 Hardware Trigger (HT) Mode....1060
22.3.4 Low Power Modes....1060
22.3.5 Operations....1060
22.3.5.1 Typical Master Mode....1060
22.3.5.1.1 Master Mode with SPI_RDY....1061
22.3.5.1.2 Master Mode with Wait States....1063
22.3.5.1.3 Master Mode with SS_CTL[3:0] Control....1063
22.3.5.1.4 Master Mode with Phase Control....1064
22.3.5.2 Typical Slave Mode....1065
22.3.6 Clocks....1066
22.3.7 Reset....1066
22.3.8 Interrupts....1067

22.3.9 DMA 1068
22.3.10 Byte Order....1069
22.4 Initialization....1069
22.5 Applications....1070
22.6 Programmable Registers....1071
22.6.1 Receive Data Register (ECSPI*x*_RXDATA)....1072
22.6.2 Transmit Data Register (ECSPI*x*_TXDATA)....1073
22.6.3 Control Register (ECSPI*x*_CONREG)....1074
22.6.4 Config Register (ECSPI*x*_CONFIGREG)....1076
22.6.5 Interrupt Control Register (ECSPI*x*_INTREG)....1078
22.6.6 DMA Control Register (ECSPI*x*_DMAREG)....1079
22.6.7 Status Register (ECSPI*x*_STATREG)....1081
22.6.8 Sample Period Control Register (ECSPI*x*_PERIODREG)....1082
22.6.9 Test Control Register (ECSPI*x*_TESTREG)....1083
22.6.10 Message Data Register (ECSPI*x*_MSGDATA)....1084

Chapter 23
External Interface Module (EIM)

23.1 Overview....1086
23.2 Features....1087
23.3 Modes of Operation....1087
23.3.1 Asynchronous Mode....1088
23.3.2 Asynchronous Page Read Mode....1088
23.3.3 Multiplexed Address/Data Mode....1088
23.3.4 Burst Clock Mode....1089
23.3.5 Low Power Modes....1090
23.3.6 Boot Mode....1090
23.4 External Signal Description....1090
23.4.1 Signals Overview....1090
23.4.2 Detailed Signal Descriptions....1091

23.4.3 Other Important Block I/O Signals Internal to the SoC....1092
23.5 Chip Select Memory Map....1093
23.6 Functional Description....1093
23.6.1 Clocks....1093
23.6.2 Bus Sizing Configuration....1094
23.6.2.1 8 BIT PORT SUPPORT....1094
23.6.2.1.1 MOTOROLA 68000....1094
23.6.2.1.2 INTEL 386....1095
23.6.3 EIM Operational Modes....1095
23.6.4 Burst Mode (Synchronous) Memory Operation....1095
23.6.5 Burst Clock Divisor (BCD)....1096
23.6.6 Burst Clock Start (BCS)....1097
23.6.7 Multiplexed Address/Data Mode Support....1097
23.6.8 Mixed Master/Memory Burst Modes Support....1097
23.6.9 AXI (Master) Bus Cycles Support....1098
23.6.10 WAIT_B Signal, RWSC and WWSC bit fields Usage....1100
23.6.11 IPS Register Interface....1100
23.6.12 MRS Set for PSRAM....1101
23.6.13 EIM Access Termination....1101
23.6.14 Error Conditions....1101
23.6.15 DTACK Mode....1102
23.6.16 RDY_INT Signal as Interrupt....1102
23.6.17 RDY_INT Signal as Ready After Reset Indication....1103
23.6.18 EIM_GRANT / EIM_BUSY Handshake Description....1103
23.6.19 LPMD / LPACK Handshake Description....1103
23.6.20 Endianness....1104
23.6.21 Strobe Signal Use....1105
23.7 Initialization Information....1105
23.7.1 Active Chip Selects and Address Space Configuration....1105

23.7.2 Booting from EIM....1106
23.8 Application Note....1107
23.8.1 Access to AMD Flash....1107
23.8.1.1 AMD Flash Asynchronous Mode Configuration....1107
23.8.1.2 AMD Flash Utility....1108
23.8.2 Access to Intel Sibley Flash....1108
23.8.2.1 Intel Sibley Flash Asynchronous Mode Configuration....1109
23.8.2.2 Intel Sibley Flash Synchronous Mode Configuration....1109
23.8.2.3 Intel Sibley Flash Utility....1109
23.8.3 Access to MDOC Device....1110
23.8.3.1 MDOC Device Boot....1110
23.8.3.2 MDOC Device Asynchronous Mode Configuration....1110
23.8.3.3 MDOC Device Utility....1110
23.8.4 Access to Micron PSRAM....1110
23.8.4.1 Micron PSRAM Asynchronous Mode Configuration....1110
23.8.4.2 Micron PSRAM Synchronous Mode Configuration....1111
23.8.5 Access to Samsung OneNAND....1111
23.8.5.1 Samsung OneNAND Boot....1111
23.8.5.2 Samsung OneNAND Asynchronous Mode Configuration....1112
23.8.5.3 Samsung OneNAND Synchronous Mode Configuration....1112
23.8.5.4 Samsung OneNAND Utility....1112
23.8.6 Access to Samsung UtRAM....1113
23.8.6.1 Samsung UtRAM Asynchronous Mode Configuration....1113
23.8.6.2 Samsung UtRAM Synchronous Mode Configuration....1113
23.8.7 Access to Spansion Flash....1113
23.8.7.1 Spansion Flash Asynchronous Mode Configuration....1113
23.8.7.2 Spansion Flash Synchronous Mode Configuration....1114
23.8.7.3 Spansion Flash Utility....1114
23.8.8 8 bit support....1115

23.9 Booting from OneNAND and NOR Flash devices....1116
23.9.1 Asynchronous Read Memory Accesses Timing Diagram....1116
23.9.2 Asynchronous Write Memory Accesses Timing Diagram....1117
23.9.3 Asynchronous Read/Write Memory Accesses Timing Diagram....1118
23.9.4 Asynchronous Read/Write Using RAL, WAL and CSREC....1120
23.9.5 Consecutive Asynchronous Write Memory Accesses Timing Diagram....1121
23.9.6 Consecutive Asynchronous Read Memory Accesses Timing Diagram....1124
23.9.7 Async. Page Mode Access....1126
23.9.8 DTACK Mode - AXI Single Access....1127
23.9.9 DTACK Mode - AXI Single Write Access....1130
23.9.10 DTACK Mode - AXI Burst Access....1131
23.10 Programmable Registers....1133
23.10.1 EIM Memory Map/Register Definition....1133
23.101.1 Chip Select n General Configuration Register 1 (EIM_CS*n*GCR1)....1135
23.101.2 Chip Select n General Configuration Register 2 (EIM_CS*n*GCR2)....1139
23.101.3 Chip Select n Read Configuration Register 1 (EIM_CS*n*RCR1)....1141
23.101.4 Chip Select n Read Configuration Register 2 (EIM_CS*n*RCR2)....1144
23.101.5 Chip Select n Write Configuration Register 1 (EIM_CS*n*WCR1)....1145
23.101.6 Chip Select n Write Configuration Register 2 (EIM_CS*n*WCR2)....1148
23.101.7 EIM Configuration Register (EIM_WCR)....1149
23.101.8 EIM IP Access Register (EIM_WIAR)....1150
23.101.9 Error Address Register (EIM_EAR)....1151

Chapter 24
Electrophoretic Display Controller (EPDC)

24.1 Overview....1153
24.2 Block Diagram and Overview....1154
24.3 Programming Model....1155
24.3.1 Assumptions....1155
24.3.2 Register Space (Write/Set/Clear/Toggle)....1156

24.3.3 Interrupts....1156
24.3.3.1 Interrupt Sources....1156
24.3.3.2 Enabling/Masking Interrupts....1157
24.3.3.3 Handling/Clearing Interrupts....1157
24.3.4 Controller Setup....1157
24.3.4.1 Memory Requirements....1158
24.3.4.2 Panel Architecture Configuration....1158
24.3.4.3 TFT Panel Timing Configuration....1161
24.3.4.4 Source-Driver and Pixel-Clock Configuration....1167
24.3.4.5 Initializing the Display....1169
24.3.4.5.1 Reset/Clocks and Buffer Preparation....1169
24.3.4.5.2 Performing an Initialization Display Update....1170
24.3.5 Dual-Scan Configuration....1170
24.3.6 Display Update Programming....1173
24.3.6.1 Initiating a Display Update....1173
24.3.6.2 Update Processing and Collisions....1174
24.3.6.3 Multiple Update Flow....1178
24.3.7 Architectural Clock Gating (Low Power Mode)....1181
24.3.8 Performance Tuning and Considerations....1183
24.3.8.1 Memory and Bus Bandwidth Requirements....1183
24.3.8.2 Pixel Latency FIFOs....1184
24.3.8.3 Basic Watermarking Control....1184
24.3.8.4 Update/Refresh Tuning (VSCAN_HOLDOFF)....1185
24.3.8.5 System-Level Arbitration Control....1185
24.4 Programmable Registers....1187
24.4.1 EPDC Control Register (EPDC_CTRL*n*)....1193
24.4.2 EPDC Waveform Address Pointer (EPDC_WVADDR)....1194
24.4.3 EPDC Working Buffer Address (EPDC_WB_ADDR)....1194
24.4.4 EPDC Screen Resolution (EPDC_RES)....1195

24.4.5 EPDC Format Control Register (EPDC_FORMAT*n*)....1196
24.4.6 EPDC FIFO Control Register (EPDC_FIFOCTRL*n*)....1197
24.4.7 EPDC Update Region Address (EPDC_UPD_ADDR)....1198
24.4.8 EPDC Update Command Co-ordinate (EPDC_UPD_CORD)....1198
24.4.9 EPDC Update Command Size (EPDC_UPD_SIZE)....1199
24.4.10 EPDC Update Command Control (EPDC_UPD_CTRL*n*)....1200
24.4.11 EPDC Update Fixed Pixel Control (EPDC_UPD_FIXED*n*)....1201
24.4.12 EPDC Temperature Register (EPDC_EPDC_TEMP)....1201
24.4.13 EPDC Timing Control Engine Control Register (EPDC_EPDC_TCE_CTRL*n*)....1202
24.4.14 EPDC Timing Control Engine Source-Driver Config Register (EPDC_TCE_SDCFG*n*)....1204
24.4.15 EPDC Timing Control Engine Gate-Driver Config Register (EPDC_TCE_GDCFG*n*)....1205
24.4.16 EPDC Timing Control Engine Horizontal Timing Register 1 (EPDC_TCE_HSCAN1*n*)....1206
24.4.17 EPDC Timing Control Engine Horizontal Timing Register 2 (EPDC_TCE_HSCAN2*n*)....1207
24.4.18 EPDC Timing Control Engine Vertical Timing Register (EPDC_TCE_VSCAN*n*)....1207
24.4.19 EPDC Timing Control Engine OE timing control Register (EPDC_TCE_OE*n*)....1208
24.4.20 EPDC Timing Control Engine Driver Polarity Register (EPDC_TCE_POLARITY*n*)....1209
24.4.21 EPDC Timing Control Engine Timing Register 1 (EPDC_TCE_TIMING1*n*)....1210
24.4.22 EPDC Timing Control Engine Timing Register 2 (EPDC_TCE_TIMING2*n*)....1211
24.4.23 EPDC Timing Control Engine Timing Register 3 (EPDC_TCE_TIMING3*n*)....1212
24.4.24 EPDC IRQ Mask Register (EPDC_IRQ_MASK*n*)....1213
24.4.25 EPDC Interrupt Register (EPDC_IRQ*n*)....1215
24.4.26 EPDC Status Register - LUTs (EPDC_EPDC_STATUS_LUTS*n*)....1217
24.4.27 EPDC Status Register - Next Available LUT (EPDC_STATUS_NEXTLUT)....1219
24.4.28 EPDC LUT Collision Status (EPDC_STATUS_COL*n*)....1220
24.4.29 EPDC General Status Register (EPDC_STATUS*n*)....1222
24.4.30 EPDC Debug register (EPDC_DEBUG*n*)....1223
24.4.31 EPDC LUTn Debug Information register (EPDC_DEBUG_LUT*n*)....1224
24.4.32 EPDC General Purpose I/O Debug register (EPDC_GPIO*n*)....1225
24.4.33 EPDC Version Register (EPDC_VERSION)....1225

Chapter 25
Enhanced Periodic Interrupt Timer (EPIT)

25.1 Overview....1228
25.1.1 Features....1228
25.1.2 Modes and Operations....1229
25.2 External Signals....1229
25.3 Functional Description....1229
25.3.1 Operating Modes....1229
25.3.1.1 Set-and-Forget Mode....1229
25.3.1.2 Free-Running Mode....1230
25.3.2 Operations....1230
25.3.3 Clocks....1231
25.3.4 Compare Event....1232
25.3.4.1 Counter Value Overwrite....1232
25.3.4.2 Low-Power Mode Behavior....1233
25.3.4.3 Debug Mode Behavior....1233
25.4 Initialization/ Application Information....1233
25.4.1 Change of Clock Source....1233
25.5 Programmable Registers....1234
25.5.1 Control register (EPIT_EPITCR)....1234
25.5.2 Status register (EPIT_EPITSR)....1237
25.5.3 Load register (EPIT_EPITLR)....1238
25.5.4 Compare register (EPIT_EPITCMPR)....1238
25.5.5 Counter register (EPIT_EPITCNR)....1239

Chapter 26
Enhanced Secured Digital Host Controller (eSDHCv2)

26.1 Overview....1241
26.1.1 Features....1243

26.1.2 Modes and Operations....1244
26.1.2.1 Data transfer Modes....1244
26.2 External Signals....1244
26.2.1 Signals Overview....1245
26.2.2 Ports Table....1245
26.3 Functional Description....1246
26.3.1 Data Buffer....1246
26.3.1.1 Write Operation Sequence....1249
26.3.1.2 Read Operation Sequence....1249
26.3.1.3 Data Buffer and Block Size....1250
26.3.1.4 Dividing Large Data Transfer....1251
26.3.1.5 External DMA Request....1252
26.3.2 DMA AHB Interface....1253
26.3.2.1 Internal DMA Request....1254
26.3.2.2 DMA Burst Length....1254
26.3.2.3 AHB Master Interface....1255
26.3.2.4 ADMA Engine....1255
26.3.2.4.1 ADMA Concept and Descriptor Format....1256
26.3.2.4.2 ADMA Interrupt....1259
26.3.2.4.3 ADMA Error-DMA....1259
26.3.3 Register Bank with IP Bus Interface....1259
26.3.4 SD Protocol Unit....1260
26.3.4.1 SD Transceiver....1261
26.3.4.2 SD Clock & Monitor....1261
26.3.4.3 Command Agent....1261
26.3.4.4 Data Agent....1262
26.3.5 Clock & Reset Manager....1262
26.3.6 Clock Generator....1263

26.3.7 SDIO Card Interrupt....1263
26.3.7.1 Interrupts in 1-bit Mode....1263
26.3.7.2 Interrupt in 4-bit Mode....1263
26.3.7.3 Card Interrupt Handling....1264
26.3.8 Card Insertion and Removal Detection....1265
26.3.9 Power Management and Wake Up Events....1266
26.3.9.1 Setting Wake Up Events....1267
26.3.10 MMC Fast Boot....1267
26.3.10.1 Boot Operation....1267
26.3.10.2 Alternative Boot Operation....1268
26.4 Initialization/Application of ESDHC....1269
26.4.1 Command Send & Response Receive Basic Operation....1269
26.4.2 Card Identification Mode....1270
26.4.2.1 Card Detect....1270
26.4.2.2 Reset....1272
26.4.2.3 Voltage Validation....1273
26.4.2.4 Card Registry....1274
26.4.3 Card Access....1275
26.4.3.1 Block Write....1276
26.4.3.1.1 Normal Write....1276
26.4.3.1.2 Write with Pause....1277
26.4.3.2 Block Read....1278
26.4.3.2.1 Normal Read....1278
26.4.3.2.2 Read with Pause....1279
26.4.3.3 Suspend Resume....1280
26.4.3.3.1 Resume....1281
26.4.3.4 ADMA1 Usage....1281
26.4.3.5 Transfer Error....1282
26.4.3.5.1 CRC Error....1282

26.4.3.5.2 Internal DMA Error....1282
26.4.3.5.3 ADMA Error-Card Access....1283
26.4.3.5.4 Auto CMD12 Error....1283
26.4.3.6 Card Interrupt....1284
26.4.4 Switch Function....1284
26.4.4.1 Query, Enable and Disable SDIO High Speed Mode....1285
26.4.4.2 Query, Enable and Disable SD High Speed Mode....1285
26.4.4.3 Query, Enable and Disable MMC High Speed Mode....1286
26.4.4.4 Set MMC Bus Width....1286
26.4.5 ADMA Operation....1286
26.4.5.1 ADMA1 Operation....1286
26.4.5.2 ADMA2 Operation....1287
26.4.6 Fast Boot Operation....1287
26.4.6.1 Normal fast boot flow....1287
26.4.6.2 Alternative fast boot flow....1288
26.4.6.3 Fast boot application case (in DMA mode)....1289
26.5 Commands for MMC/SD/SDIO....1291
26.6 Software Restrictions....1296
26.6.1 Initialization Active....1296
26.6.2 Softeware Polling Procedure....1297
26.6.3 Suspend Operation....1297
26.6.4 Data Length Setting....1297
26.6.5 (A)DMA Address Setting....1297
26.6.6 Data Port Access....1297
26.6.7 Change Clock Frequency....1298
26.6.8 Multi-block Read....1298
26.7 Programmable Registers....1298
26.7.1 DMA System Address (ESDHCV2*x*_DSADDR)....1302
26.7.2 Block Attributes (ESDHCV2*x*_BLKATTR)....1303

26.7.3 Command Argument (ESDHCV2x_CMDARG)....1304
26.7.4 Command Transfer Type (ESDHCV2x_XFERTYP)....1305
26.7.5 Command Response 0 (ESDHCV2x_CMDRSP0)....1309
26.7.6 Command Response 1 (ESDHCV2x_CMDRSP1)....1310
26.7.7 Command Response 2 (ESDHCV2x_CMDRSP2)....1310
26.7.8 Command Response 3 (ESDHCV2x_CMDRSP3)....1310
26.7.9 Data Buffer Access Port (ESDHCV2x_DATPORT)....1312
26.7.10 Present State (ESDHCV2x_PRSSTAT)....1313
26.7.11 Protocol Control (ESDHCV2x_PROCTL)....1318
26.7.12 System Control (ESDHCV2x_SYSCTL)....1322
26.7.13 Interrupt Status (ESDHCV2x_IRQSTAT)....1326
26.7.14 Interrupt Status Enable (ESDHCV2x_IRQSTATEN)....1331
26.7.15 Interrupt Signal Enable (ESDHCV2x_IRQSIGEN)....1334
26.7.16 Auto CMD12 Status (ESDHCV2x_AUTOC12ERR)....1336
26.7.17 Host Controller Capabilities (ESDHCV2x_HOSTCAPBLT)....1339
26.7.18 Watermark Level (ESDHCV2x_WML)....1341
26.7.19 Force Event (ESDHCV2x_FEVT)....1342
26.7.20 ADMA Error Status Register (ESDHCV2x_ADMAES)....1344
26.7.21 ADMA System Address (ESDHCV2x_ADSADDR)....1346
26.7.22 Vendor Specific Register (ESDHCV2x_VENDOR)....1347
26.7.23 MMC Boot Register (ESDHCV2x_MMCBOOT)....1348
26.7.24 Host Controller Version (ESDHCV2x_HOSTVER)....1349

Chapter 27
Enhanced Secured Digital Host Controller (eSDHCv3)

27.1 Overview....1351
27.1.1 Features....1353
27.1.2 Modes and Operations....1354
27.1.2.1 Data transfer Modes....1354

27.2 External Signals....1355
27.2.1 Signals Overview....1355
27.2.2 Ports Table....1355
27.3 Functional Description....1356
27.3.1 Data Buffer....1356
27.3.1.1 Write Operation Sequence....1359
27.3.1.2 Read Operation Sequence....1360
27.3.1.3 Data Buffer and Block Size....1360
27.3.1.4 Dividing Large Data Transfer....1361
27.3.1.5 External DMA Request....1362
27.3.2 DMA AHB Interface....1363
27.3.2.1 Internal DMA Request....1364
27.3.2.2 DMA Burst Length....1364
27.3.2.3 AHB Master Interface....1365
27.3.2.4 ADMA Engine....1365
27.3.2.4.1 ADMA Concept and Descriptor Format....1366
27.3.2.4.2 ADMA Interrupt....1369
27.3.2.4.3 ADMA Error-DMA....1369
27.3.3 Register Bank with IP Bus Interface....1369
27.3.4 SD Protocol Unit....1370
27.3.4.1 SD Transceiver....1371
27.3.4.2 SD Clock & Monitor....1371
27.3.4.3 Command Agent....1371
27.3.4.4 Data Agent....1372
27.3.5 Clock & Reset Manager....1372
27.3.6 Clock Generator....1373
27.3.7 SDIO Card Interrupt....1373
27.3.7.1 Interrupts in 1-bit Mode....1373
27.3.7.2 Interrupt in 4-bit Mode....1373

27.3.7.3 Card Interrupt Handling....1374
27.3.8 Card Insertion and Removal Detection....1375
27.3.9 Power Management and Wake Up Events....1376
27.3.9.1 Setting Wake Up Events....1377
27.3.10 MMC Fast Boot....1377
27.3.10.1 Boot Operation....1377
27.3.10.2 Alternative Boot Operation....1378
27.4 Initialization/Application of ESDHC....1379
27.4.1 Command Send & Response Receive Basic Operation....1379
27.4.2 Card Identification Mode....1380
27.4.2.1 Card Detect....1380
27.4.2.2 Reset....1382
27.4.2.3 Voltage Validation....1383
27.4.2.4 Card Registry....1384
27.4.3 Card Access....1385
27.4.3.1 Block Write....1386
27.4.3.1.1 Normal Write....1386
27.4.3.1.2 DDR Write....1387
27.4.3.1.3 Write with Pause....1387
27.4.3.2 Block Read....1389
27.4.3.2.1 Normal Read....1389
27.4.3.2.2 DDR Read....1390
27.4.3.2.3 Read with Pause....1390
27.4.3.2.4 DLL (Delay Line) in Read Path....1391
27.4.3.3 Suspend Resume....1392
27.4.3.3.1 Resume....1393
27.4.3.4 ADMA1 Usage....1393
27.4.3.5 Transfer Error....1394
27.4.3.5.1 CRC Error....1394

27.4.3.5.2 Internal DMA Error....1394
27.4.3.5.3 ADMA Error-Card Access....1395
27.4.3.5.4 Auto CMD12 Error....1395
27.4.3.6 Card Interrupt....1396
27.4.4 Switch Function....1396
27.4.4.1 Query, Enable and Disable SDIO High Speed Mode....1397
27.4.4.2 Query, Enable and Disable SD High Speed Mode....1397
27.4.4.3 Query, Enable and Disable MMC High Speed Mode....1398
27.4.4.4 Set MMC Bus Width....1398
27.4.5 ADMA Operation....1398
27.4.5.1 ADMA1 Operation....1398
27.4.5.2 ADMA2 Operation....1399
27.4.6 Fast Boot Operation....1399
27.4.6.1 Normal fast boot flow....1399
27.4.6.2 Alternative fast boot flow....1400
27.4.6.3 Fast boot application case (in DMA mode)....1401
27.5 Commands for MMC/SD/SDIO....1403
27.6 Software Restrictions....1408
27.6.1 Initialization Active....1409
27.6.2 Softeware Polling Procedure....1409
27.6.3 Suspend Operation....1409
27.6.4 Data Length Setting....1409
27.6.5 (A)DMA Address Setting....1409
27.6.6 Data Port Access....1410
27.6.7 Change Clock Frequency....1410
27.6.8 Multi-block Read....1410
27.7 Programmable Registers....1411
27.7.1 DMA System Address (ESDHCV3*x*_DSADDR)....1412
27.7.2 Block Attributes (ESDHCV3*x*_BLKATTR)....1413

27.7.3 Command Argument (ESDHCV3x_CMDARG)....1414
27.7.4 Command Transfer Type (ESDHCV3x_XFERTYP)....1415
27.7.5 Command Response n (ESDHCV3x_CMDRSPn)....1419
27.7.6 Data Buffer Access Port (ESDHCV3x_DATPORT)....1421
27.7.7 Present State (ESDHCV3x_PRSSTAT)....1421
27.7.8 Protocol Control (ESDHCV3x_PROCTL)....1426
27.7.9 System Control (ESDHCV3x_SYSCTL)....1430
27.7.10 Interrupt Status (ESDHCV3x_IRQSTAT)....1434
27.7.11 Interrupt Status Enable (ESDHCV3x_IRQSTATEN)....1439
27.7.12 Interrupt Signal Enable (ESDHCV3x_IRQSIGEN)....1442
27.7.13 Auto CMD12 Status (ESDHCV3x_AUTOC12ERR)....1444
27.7.14 Host Controller Capabilities (ESDHCV3x_HOSTCAPBLT)....1447
27.7.15 Watermark Level (ESDHCV3x_WML)....1449
27.7.16 Force Event (ESDHCV3x_FEVT)....1450
27.7.17 ADMA Error Status Register (ESDHCV3x_ADMAES)....1452
27.7.18 ADMA System Address (ESDHCV3x_ADSADDR)....1454
27.7.19 DLL (Delay Line) Control (ESDHCV3x_DLLCTRL)....1455
27.7.20 DLL Status (ESDHCV3x_DLLSTS)....1456
27.7.21 Vendor Specific Register (ESDHCV3x_VENDOR)....1457
27.7.22 MMC Boot Register (ESDHCV3x_MMCBOOT)....1458
27.7.23 Host Controller Version (ESDHCV3x_HOSTVER)....1459

Chapter 28
Fast Ethernet Controller (FEC)

28.1 Overview....1461
28.1.1 Features....1463
28.2 Modes of Operation....1464
28.2.1 Full- and Half-Duplex Operation....1464
28.2.2 Interface Options....1464
28.2.2.1 10-Mbps and 100-Mbps Media Independent Interface (MII)....1464

28.2.2.2 10 Mbps and 100 Mbps RMII Interface....1465
28.2.2.3 10-Mbps 7-Wire Interface Operation....1465
28.2.3 Address Recognition Options....1465
28.2.4 Internal Loopback....1465
28.3 Functional Description....1466
28.3.1 Network Interface Options....1466
28.3.2 FEC Frame Transmission....1467
28.3.2.1 Transmit Inter-Packet Gap (IPG) Time....1468
28.3.2.2 Collision Handling....1468
28.3.2.3 Transmission Error Handling....1469
28.3.2.3.1 Transmitter Underrun....1469
28.3.2.3.2 Retransmission Attempts Limit Expired....1469
28.3.2.3.3 Late Collision....1469
28.3.2.3.4 Heartbeat....1470
28.3.3 FEC Frame Reception....1470
28.3.3.1 Receive Inter-Packet Gap (IPG) Time....1471
28.3.3.2 Ethernet Address Recognition....1471
28.3.3.2.1 Hash Algorithm....1474
28.3.3.3 Reception Error Handling....1477
28.3.3.3.1 Overrun....1477
28.3.3.3.2 Non-Octet (Dribbling Bits)....1477
28.3.3.3.3 CRC....1478
28.3.3.3.4 Frame Length Violation....1478
28.3.3.3.5 Truncation....1478
28.3.4 Full-Duplex Flow Control....1478
28.3.5 Internal and External Loopback....1480
28.4 Initialization/Application Information....1480
28.4.1 Initialization Sequence....1480
28.4.1.1 Hardware Controlled Initialization....1480

28.4.1.2 User Initialization (Prior to Asserting FEC_ECR[ETHER_EN])....1481
28.4.1.3 Microcontroller Initialization....1482
28.4.1.4 User Initialization (after asserting FEC_ECR[ETHER_EN])....1482
28.4.2 Buffer Descriptors....1483
28.4.2.1 Driver/DMA Operation with Buffer Descriptors....1483
28.4.2.2 Ethernet Transmit Buffer Descriptor (TxBD)....1484
28.4.2.2.1 Driver/DMA Operation with Transmit Buffer Descriptors....1485
28.4.2.2.1.1 Transmit Frame in Multiple Buffers....1485
28.4.2.3 Ethernet Receive Buffer Descriptor (RxBD)....1486
28.4.2.3.1 Driver/DMA Operation with Receive Buffer Descriptors....1488
28.5 Programmable Registers....1489
28.5.1 Top Level Block Memory Map....1489
28.5.2 Message Information Block (MIB) Counters Memory Map....1489
28.5.3 MIIGSK Registers Memory Map....1492
28.5.4 Ethernet interrupt event register (FEC_EIR)....1494
28.5.5 Ethernet interrupt mask register (FEC_EIMR)....1496
28.5.6 Receive descriptor active register (FEC_RDAR)....1498
28.5.7 Transmit descriptor active register (FEC_TDAR)....1499
28.5.8 Ethernet control register (FEC_ECR)....1500
28.5.9 MII management frame register (FEC_MMFR)....1501
28.5.10 MII speed control register (FEC_MSCR)....1503
28.5.11 MIB control register (FEC_MIBC)....1505
28.5.12 Receive control register (FEC_RCR)....1506
28.5.13 Transmit control register (FEC_TCR)....1507
28.5.14 Physical address low register (FEC_PALR)....1508
28.5.15 Physical address upper register (FEC_PAUR)....1509
28.5.16 Opcode and pause duration register (FEC_OPDR)....1510
28.5.17 Descriptor individual address upper register (FEC_IAUR)....1510
28.5.18 Descriptor individual address lower register (FEC_IALR)....1511

28.5.19 Descriptor group address upper register (FEC_GAUR)....1511
28.5.20 Descriptor group address lower register (FEC_GALR)....1512
28.5.21 Transmit FIFO watermark register (FEC_TFWR)....1512
28.5.22 FIFO receive bound register (FEC_FRBR)....1513
28.5.23 FIFO receive FIFO start registers (FEC_FRSR)....1513
28.5.24 Receive buffer descriptor ring start register (FEC_ERDSR)....1514
28.5.25 Transmit buffer descriptor ring start register (FEC_ETDSR)....1515
28.5.26 Maximum receive buffer size register (FEC_EMRBR)....1515

Chapter 29
General Power Controller (GPC)

29.1 Introduction....1517
29.1.1 Overview....1517
29.1.2 Features....1518
29.2 Functional Description....1518
29.2.1 DVFS - Dynamic Voltage & Frequency Scaling....1519
29.2.2 DVFS Change Request Sequence Diagrams....1520
29.2.3 Frequency / Voltage Change Controller Description....1522
29.2.3.1 GPC Controller Description....1522
29.2.3.2 Clock Control Module Frequency Update Controller Description....1524
29.2.4 State Retention Power Gating (SRPG)....1526
29.2.5 PMIC Interface Requirements for APM Support....1526
29.3 Programmable Registers....1530
29.3.1 Interface Control Register (GPC_CNTR)....1530
29.3.2 Voltage Counter Register (GPC_VCR)....1532
29.3.3 NEON register (GPC_NEON)....1533

Chapter 30
General Purpose Input/Output (GPIO)

30.1 Overview....1535
30.1.1 Features....1537

30.2 GPIO Functional Description....1537
30.2.1 GPIO Function....1537
30.2.2 GPIO Programming....1538
30.2.2.1 GPIO Read Mode....1538
30.2.2.2 GPIO Write Mode....1538
30.2.3 Interrupt Control Unit....1539
30.3 Programmable Registers....1539
30.3.1 GPIO data register (GPIO*x*_GPIO_DR)....1542
30.3.2 GPIO direction register (GPIO*x*_GPIO_GDIR)....1543
30.3.3 GPIO pad status register (GPIO*x*_GPIO_PSR)....1544
30.3.4 GPIO interrupt configuration register1 (GPIO*x*_GPIO_ICR1)....1544
30.3.5 GPIO interrupt configuration register2 (GPIO*x*_GPIO_ICR2)....1545
30.3.6 GPIO interrupt mask register (GPIO*x*_GPIO_IMR)....1546
30.3.7 GPIO interrupt status register (GPIO*x*_GPIO_ISR)....1547
30.3.8 GPIO edge select register (GPIO*x*_GPIO_EDGE_SEL)....1548

Chapter 31
General-Purpose Media Interface (GPMI)

31.1 Introduction....1549
31.2 Overview....1549
31.3 GPMI NAND Mode....1550
31.3.1 Multiple NAND Support....1551
31.3.2 GPMI NAND Timing and Clocking....1552
31.3.3 Basic NAND Timing....1552
31.3.3.1 NAND Asynchronous Timing....1552
31.3.3.2 NAND Asynchronous EDO Mode Timing....1554
31.3.3.3 NAND ONFI Source Synchronous Mode Timing....1557
31.3.3.4 NAND Toggle Mode Timing....1561
31.3.4 Hardware BCH Interface....1564
31.4 Behavior During Reset....1565

31.5 Programmable Registers....1565
31.5.1 GPMI Control Register 0 (GPMI_CTRL0*n*)....1567
31.5.2 GPMI Compare Register (GPMI_COMPARE)....1569
31.5.3 GPMI Integrated ECC Control Register (GPMI_ECCCTRL*n*)....1570
31.5.4 GPMI Integrated ECC Transfer Count Register (GPMI_ECCCOUNT)....1571
31.5.5 GPMI Payload Address Register (GPMI_PAYLOAD)....1572
31.5.6 GPMI Auxiliary Address Register (GPMI_AUXILIARY)....1572
31.5.7 GPMI Control Register 1 (GPMI_CTRL1*n*)....1573
31.5.8 GPMI Timing Register 0 (GPMI_TIMING0)....1576
31.5.9 GPMI Timing Register 1 (GPMI_TIMING1)....1577
31.5.10 GPMI Timing Register 2 (GPMI_TIMING2)....1577
31.5.11 GPMI DMA Data Transfer Register (GPMI_DATA)....1578
31.5.12 GPMI Status Register (GPMI_STAT)....1579
31.5.13 GPMI Debug Information Register (GPMI_DEBUG)....1581
31.5.14 GPMI Version Register (GPMI_VERSION)....1581
31.5.15 GPMI Debug2 Information Register (GPMI_DEBUG2)....1582
31.5.16 GPMI Debug3 Information Register (GPMI_DEBUG3)....1584
31.5.17 GPMI Double Rate Read DLL Control Register (GPMI_READ_DDR_DLL_CTRL)....1585
31.5.18 GPMI Double Rate Write DLL Control Register (GPMI_WRITE_DDR_DLL_CTRL)....1586
31.5.19 GPMI Double Rate Read DLL Status Register (GPMI_READ_DDR_DLL_STS)....1587
31.5.20 GPMI Double Rate Write DLL Status Register (GPMI_WRITE_DDR_DLL_STS)....1588

Chapter 32
General Purpose Timer (GPT)

32.1 Overview....1591
32.1.1 Features....1593
32.1.2 Modes and Operation....1593
32.2 External Signals....1594
32.2.1 External Clock Input: CLKIN....1594
32.2.2 Input Capture Trigger Signals: CAPIN1, CAPIN2....1594

32.2.3 Output Compare Signals: DO_CMPOUT1, DO_CMPOUT2, DO_CMPOUT3....1595
32.3 Functional Description....1595
32.3.1 Operating Modes....1595
32.3.1.1 Restart Mode....1595
32.3.1.2 Free-Run Mode....1595
32.3.2 Operation....1596
32.3.2.1 Clocks....1596
32.3.2.2 Input Capture....1598
32.3.2.3 Output Compare....1599
32.3.2.4 Interrupts....1600
32.3.2.5 Low Power Mode Behavior....1601
32.3.2.6 Debug Mode Behavior....1601
32.4 Programmable Registers....1601
32.4.1 GPT Control Register (GPT_CR)....1603
32.4.2 GPT Prescaler Register (GPT_PR)....1606
32.4.3 GPT Status Register (GPT_SR)....1607
32.4.4 GPT Interrupt Register (GPT_IR)....1608
32.4.5 GPT Output Compare Register 1 (GPT_OCR1)....1609
32.4.6 GPT Output Compare Register 2 (GPT_OCR2)....1610
32.4.7 GPT Output Compare Register 3 (GPT_OCR3)....1610
32.4.8 GPT Input Capture Register 1 (GPT_ICR1)....1611
32.4.9 GPT Input Capture Register 2 (GPT_ICR2)....1611
32.4.10 GPT Counter Register (GPT_CNT)....1612

Chapter 33
2D Graphics Processing Unit (GPU2D)

33.1 Overview....1613
33.2 GPU2D Feature List....1613
33.2.1 Frame Buffer....1613
33.2.2 2D Bitmap Graphics (Separate 2D Unit)....1614

33.2.3 Vector Graphics....1615
33.3 GPU2D Block Diagram....1616
33.4 GPU SoC Interface....1616
33.4.1 GPU2D Top Level Diagram....1616
33.4.2 SoC Bus Connection....1617
33.5 Other Signals Connection....1618
33.6 Clocking Architecture....1618
33.7 Power Management....1618
33.8 Modes of Operation....1619
33.9 Reset....1619
33.10 Interrupts....1620
33.11 DMA....1620
33.12 Memory Map....1620
33.12.1 AHB Slave Interface....1620
33.12.2 AXI Master Memory Interface (EXTMC Port)....1621

Chapter 34
I2C Controller (I2C)

34.1 Overview....1623
34.1.1 Features....1625
34.1.2 Modes and Operations....1626
34.1.2.1 Standard Mode....1626
34.1.2.2 Fast Mode....1626
34.2 External Signals....1626
34.3 Functional Description....1627
34.3.1 I2C System Configuration....1627
34.3.2 I2C Protocol....1627
34.3.2.1 START Signal....1628
34.3.2.2 Slave Address Transmission....1628
34.3.2.3 Data Transfer....1628

34.3.2.4 STOP Signal 1629
34.3.2.5 Repeat Start....1629
34.3.3 Arbitration Procedure....1630
34.3.4 Clock Synchronization....1630
34.3.5 Handshaking....1631
34.3.6 Clock Stretching....1631
34.3.7 Peripheral Bus Accesses....1632
34.3.8 Generation of Transfer Error on IP Bus....1632
34.3.9 Clocks....1632
34.3.10 Reset....1632
34.3.11 Interrupts....1632
34.3.12 Byte Order....1633
34.4 Initialization....1633
34.4.1 Initialization Sequence....1633
34.4.2 Generation of START....1633
34.4.3 Post-Transfer Software Response....1634
34.4.4 Generation of STOP....1634
34.4.5 Generation of Repeated START....1635
34.4.6 Slave Mode....1635
34.4.7 Arbitration Lost....1635
34.5 Software Restriction....1644
34.6 Programmable Registers....1644
34.6.1 I2C Memory Map/Register Definition....1644
34.61.1 I2C Address Register (I2C*x*_IADR)....1645
34.61.2 I2C Frequency Divider Register (I2C*x*_IFDR)....1646
34.61.3 I2C Control Register (I2C*x*_I2CR)....1647
34.61.4 I2C Status Register (I2C*x*_I2SR)....1649
34.61.5 I2C Data I/O Register (I2C*x*_I2DR)....1650

Chapter 35
IOMUXC

35.1 Overview....1653
35.1.1 Features....1654
35.1.2 Modes of Operation....1655
35.2 External Signal Description....1655
35.3 Functional Description....1655
35.3.1 ALT6 and ALT7 Extended Muxing Modes....1658
35.3.2 SW Loopback through SION Bit....1658
35.3.3 "Daisy Chain"-Multi Pads Driving Same Module Input Pin....1658
35.3.4 Interrupts....1660
35.4 Programmable Registers....1660
35.4.1 GPR0 (IOMUXC_GPR0)....1686
35.4.2 GPR1 (IOMUXC_GPR1)....1687
35.4.3 GPR2 (IOMUXC_GPR2)....1688
35.4.4 OBSERVE_MUX_0 (IOMUXC_OBSMUX0)....1689
35.4.5 OBSERVE_MUX_1 (IOMUXC_OBSMUX1)....1690
35.4.6 OBSERVE_MUX_2 (IOMUXC_OBSMUX2)....1691
35.4.7 OBSERVE_MUX_3 (IOMUXC_OBSMUX3)....1692
35.4.8 OBSERVE_MUX_4 (IOMUXC_OBSMUX4)....1693
35.4.9 SW_MUX_CTL_PAD_KEY_COL0 (IOMUXC_SMUXC_PKC0)....1695
35.4.10 SW_MUX_CTL_PAD_KEY_ROW0 (IOMUXC_SMUXC_PKR0)....1696
35.4.11 SW_MUX_CTL_PAD_KEY_COL1 (IOMUXC_SMUXC_PKC1)....1697
35.4.12 SW_MUX_CTL_PAD_KEY_ROW1 (IOMUXC_SMUXC_PKR1)....1698
35.4.13 SW_MUX_CTL_PAD_KEY_COL2 (IOMUXC_SMUXC_PKC2)....1699
35.4.14 SW_MUX_CTL_PAD_KEY_ROW2 (IOMUXC_SMUXC_PKR2)....1700
35.4.15 SW_MUX_CTL_PAD_KEY_COL3 (IOMUXC_SMUXC_PKC3)....1701
35.4.16 SW_MUX_CTL_PAD_KEY_ROW3 (IOMUXC_SMUXC_PKR3)....1702
35.4.17 SW_MUX_CTL_PAD_I2C1_SCL (IOMUXC_SMUXC_PI2C1_SCL)....1703

35.4.18 SW_MUX_CTL_PAD_I2C1_SDA (IOMUXC_SMUXC_PI2C1_SDA)....1704
35.4.19 SW_MUX_CTL_PAD_I2C2_SCL (IOMUXC_SMUXC_PI2C2_SCL)....1705
35.4.20 SW_MUX_CTL_PAD_I2C2_SDA (IOMUXC_SMUXC_PI2C2_SDA)....1706
35.4.21 SW_MUX_CTL_PAD_I2C3_SCL (IOMUXC_SMUXC_PI2C3_SCL)....1707
35.4.22 SW_MUX_CTL_PAD_I2C3_SDA (IOMUXC_SMUXC_PI2C3_SDA)....1708
35.4.23 SW_MUX_CTL_PAD_PWM1 (IOMUXC_SMUXC_PPWM1)....1709
35.4.24 SW_MUX_CTL_PAD_PWM2 (IOMUXC_SMUXC_PPWM2)....1710
35.4.25 SW_MUX_CTL_PAD_OWIRE (IOMUXC_SMUXC_POWIRE)....1711
35.4.26 SW_MUX_CTL_PAD_EPITO (IOMUXC_SMUXC_PEPITO)....1712
35.4.27 SW_MUX_CTL_PAD_WDOG (IOMUXC_SMUXC_PWDOG)....1713
35.4.28 SW_MUX_CTL_PAD_SSI_TXFS (IOMUXC_SMUXC_PSSI_TXFS)....1714
35.4.29 SW_MUX_CTL_PAD_SSI_TXC (IOMUXC_SMUXC_PSSI_TXC)....1715
35.4.30 SW_MUX_CTL_PAD_SSI_TXD (IOMUXC_SMUXC_PSSI_TXD)....1716
35.4.31 SW_MUX_CTL_PAD_SSI_RXD (IOMUXC_SMUXC_PSSI_RXD)....1717
35.4.32 SW_MUX_CTL_PAD_SSI_RXF (IOMUXC_SMUXC_PSSI_RXF)....1718
35.4.33 SW_MUX_CTL_PAD_SSI_RXC (IOMUXC_SMUXC_PSSI_RXC)....1719
35.4.34 SW_MUX_CTL_PAD_UART1_TXD (IOMUXC_SMUXC_PUART1_TXD)....1720
35.4.35 SW_MUX_CTL_PAD_UART1_RXD (IOMUXC_SMUXC_PUART1_RXD)....1721
35.4.36 SW_MUX_CTL_PAD_UART1_CTS (IOMUXC_SMUXC_PUART1_CTS)....1722
35.4.37 SW_MUX_CTL_PAD_UART1_RTS (IOMUXC_SMUXC_PUART1_RTS)....1723
35.4.38 SW_MUX_CTL_PAD_UART2_TXD (IOMUXC_SMUXC_PUART2_TXD)....1724
35.4.39 SW_MUX_CTL_PAD_UART2_RXD (IOMUXC_SMUXC_PUART2_RXD)....1725
35.4.40 SW_MUX_CTL_PAD_UART2_CTS (IOMUXC_SMUXC_PUART2_CTS)....1726
35.4.41 SW_MUX_CTL_PAD_UART2_RTS (IOMUXC_SMUXC_PUART2_RTS)....1727
35.4.42 SW_MUX_CTL_PAD_UART3_TXD (IOMUXC_SMUXC_PUART3_TXD)....1728
35.4.43 SW_MUX_CTL_PAD_UART3_RXD (IOMUXC_SMUXC_PUART3_RXD)....1730
35.4.44 SW_MUX_CTL_PAD_UART4_TXD (IOMUXC_SMUXC_PUART4_TXD)....1731
35.4.45 SW_MUX_CTL_PAD_UART4_RXD (IOMUXC_SMUXC_PUART4_RXD)....1732
35.4.46 SW_MUX_CTL_PAD_CSPI_SCLK (IOMUXC_SMUXC_PCSPI_SCLK)....1733

35.4.47 SW_MUX_CTL_PAD_CSPI_MOSI (IOMUXC_SMUXC_PCSPI_MOSI)....1734
35.4.48 SW_MUX_CTL_PAD_CSPI_MISO (IOMUXC_SMUXC_PCSPI_MISO)....1735
35.4.49 SW_MUX_CTL_PAD_CSPI_SS0 (IOMUXC_SMUXC_PCSPI_SS0)....1736
35.4.50 SW_MUX_CTL_PAD_ECSPI1_SCLK (IOMUXC_SMUXC_PECSPI1_SCLK)....1737
35.4.51 SW_MUX_CTL_PAD_ECSPI1_MOSI (IOMUXC_SMUXC_PECSPI1_MOSI)....1738
35.4.52 SW_MUX_CTL_PAD_ECSPI1_MISO (IOMUXC_SMUXC_PECSPI1_MISO)....1739
35.4.53 SW_MUX_CTL_PAD_ECSPI1_SS0 (IOMUXC_SMUXC_PECSPI1_SS0)....1740
35.4.54 SW_MUX_CTL_PAD_ECSPI2_SCLK (IOMUXC_SMUXC_PECSPI2_SCLK)....1741
35.4.55 SW_MUX_CTL_PAD_ECSPI2_MOSI (IOMUXC_SMUXC_PECSPI2_MOSI)....1742
35.4.56 SW_MUX_CTL_PAD_ECSPI2_MISO (IOMUXC_SMUXC_PECSPI2_MISO)....1743
35.4.57 SW_MUX_CTL_PAD_ECSPI2_SS0 (IOMUXC_SMUXC_PECSPI2_SS0)....1745
35.4.58 SW_MUX_CTL_PAD_SD1_CLK (IOMUXC_SMUXC_PSD1_CLK)....1746
35.4.59 SW_MUX_CTL_PAD_SD1_CMD (IOMUXC_SMUXC_PSD1_CMD)....1747
35.4.60 SW_MUX_CTL_PAD_SD1_D0 (IOMUXC_SMUXC_PSD1_D0)....1748
35.4.61 SW_MUX_CTL_PAD_SD1_D1 (IOMUXC_SMUXC_PSD1_D1)....1749
35.4.62 SW_MUX_CTL_PAD_SD1_D2 (IOMUXC_SMUXC_PSD1_D2)....1750
35.4.63 SW_MUX_CTL_PAD_SD1_D3 (IOMUXC_SMUXC_PSD1_D3)....1751
35.4.64 SW_MUX_CTL_PAD_SD2_CLK (IOMUXC_SMUXC_PSD2_CLK)....1752
35.4.65 SW_MUX_CTL_PAD_SD2_CMD (IOMUXC_SMUXC_PSD2_CMD)....1753
35.4.66 SW_MUX_CTL_PAD_SD2_D0 (IOMUXC_SMUXC_PSD2_D0)....1754
35.4.67 SW_MUX_CTL_PAD_SD2_D1 (IOMUXC_SMUXC_PSD2_D1)....1755
35.4.68 SW_MUX_CTL_PAD_SD2_D2 (IOMUXC_SMUXC_PSD2_D2)....1756
35.4.69 SW_MUX_CTL_PAD_SD2_D3 (IOMUXC_SMUXC_PSD2_D3)....1757
35.4.70 SW_MUX_CTL_PAD_SD2_D4 (IOMUXC_SMUXC_PSD2_D4)....1758
35.4.71 SW_MUX_CTL_PAD_SD2_D5 (IOMUXC_SMUXC_PSD2_D5)....1759
35.4.72 SW_MUX_CTL_PAD_SD2_D6 (IOMUXC_SMUXC_PSD2_D6)....1760
35.4.73 SW_MUX_CTL_PAD_SD2_D7 (IOMUXC_SMUXC_PSD2_D7)....1761
35.4.74 SW_MUX_CTL_PAD_SD2_WP (IOMUXC_SMUXC_PSD2_WP)....1762
35.4.75 SW_MUX_CTL_PAD_SD2_CD (IOMUXC_SMUXC_PSD2_CD)....1763

35.4.76 SW_MUX_CTL_PAD_DISP_D0 (IOMUXC_SMUXC_PDISP_D0)....1764
35.4.77 SW_MUX_CTL_PAD_DISP_D1 (IOMUXC_SMUXC_PDISP_D1)....1765
35.4.78 SW_MUX_CTL_PAD_DISP_D2 (IOMUXC_SMUXC_PDISP_D2)....1766
35.4.79 SW_MUX_CTL_PAD_DISP_D3 (IOMUXC_SMUXC_PDISP_D3)....1767
35.4.80 SW_MUX_CTL_PAD_DISP_D4 (IOMUXC_SMUXC_PDISP_D4)....1768
35.4.81 SW_MUX_CTL_PAD_DISP_D5 (IOMUXC_SMUXC_PDISP_D5)....1769
35.4.82 SW_MUX_CTL_PAD_DISP_D6 (IOMUXC_SMUXC_PDISP_D6)....1770
35.4.83 SW_MUX_CTL_PAD_DISP_D7 (IOMUXC_SMUXC_PDISP_D7)....1771
35.4.84 SW_MUX_CTL_PAD_DISP_WR (IOMUXC_SMUXC_PDISP_WR)....1772
35.4.85 SW_MUX_CTL_PAD_DISP_RD (IOMUXC_SMUXC_PDISP_RD)....1773
35.4.86 SW_MUX_CTL_PAD_DISP_RS (IOMUXC_SMUXC_PDISP_RS)....1774
35.4.87 SW_MUX_CTL_PAD_DISP_CS (IOMUXC_SMUXC_PDISP_CS)....1775
35.4.88 SW_MUX_CTL_PAD_DISP_BUSY (IOMUXC_SMUXC_PDISP_BUSY)....1776
35.4.89 SW_MUX_CTL_PAD_DISP_RESET (IOMUXC_SMUXC_PDISP_RESET)....1777
35.4.90 SW_MUX_CTL_PAD_SD3_CMD (IOMUXC_SMUXC_PSD3_CMD)....1778
35.4.91 SW_MUX_CTL_PAD_SD3_CLK (IOMUXC_SMUXC_PSD3_CLK)....1779
35.4.92 SW_MUX_CTL_PAD_SD3_D0 (IOMUXC_SMUXC_PSD3_D0)....1780
35.4.93 SW_MUX_CTL_PAD_SD3_D1 (IOMUXC_SMUXC_PSD3_D1)....1781
35.4.94 SW_MUX_CTL_PAD_SD3_D2 (IOMUXC_SMUXC_PSD3_D2)....1782
35.4.95 SW_MUX_CTL_PAD_SD3_D3 (IOMUXC_SMUXC_PSD3_D3)....1783
35.4.96 SW_MUX_CTL_PAD_SD3_D4 (IOMUXC_SMUXC_PSD3_D4)....1784
35.4.97 SW_MUX_CTL_PAD_SD3_D5 (IOMUXC_SMUXC_PSD3_D5)....1785
35.4.98 SW_MUX_CTL_PAD_SD3_D6 (IOMUXC_SMUXC_PSD3_D6)....1786
35.4.99 SW_MUX_CTL_PAD_SD3_D7 (IOMUXC_SMUXC_PSD3_D7)....1787
35.4.100 SW_MUX_CTL_PAD_SD3_WP (IOMUXC_SMUXC_PSD3_WP)....1788
35.4.101 SW_MUX_CTL_PAD_DISP_D8 (IOMUXC_SMUXC_PDISP_D8)....1789
35.4.102 SW_MUX_CTL_PAD_DISP_D9 (IOMUXC_SMUXC_PDISP_D9)....1790
35.4.103 SW_MUX_CTL_PAD_DISP_D10 (IOMUXC_SMUXC_PDISP_D10)....1791
35.4.104 SW_MUX_CTL_PAD_DISP_D11 (IOMUXC_SMUXC_PDISP_D11)....1792

35.4.105 SW_MUX_CTL_PAD_DISP_D12 (IOMUXC_SMUXC_PDISP_D12).....1794
35.4.106 SW_MUX_CTL_PAD_DISP_D13 (IOMUXC_SMUXC_PDISP_D13).....1795
35.4.107 SW_MUX_CTL_PAD_DISP_D14 (IOMUXC_SMUXC_PDISP_D14).....1796
35.4.108 SW_MUX_CTL_PAD_DISP_D15 (IOMUXC_SMUXC_PDISP_D15).....1797
35.4.109 SW_MUX_CTL_PAD_EPDC_D0 (IOMUXC_SMUXC_PEPDC_D0).....1799
35.4.110 SW_MUX_CTL_PAD_EPDC_D1 (IOMUXC_SMUXC_PEPDC_D1).....1800
35.4.111 SW_MUX_CTL_PAD_EPDC_D2 (IOMUXC_SMUXC_PEPDC_D2).....1801
35.4.112 SW_MUX_CTL_PAD_EPDC_D3 (IOMUXC_SMUXC_PEPDC_D3).....1802
35.4.113 SW_MUX_CTL_PAD_EPDC_D4 (IOMUXC_SMUXC_PEPDC_D4).....1803
35.4.114 SW_MUX_CTL_PAD_EPDC_D5 (IOMUXC_SMUXC_PEPDC_D5).....1804
35.4.115 SW_MUX_CTL_PAD_EPDC_D6 (IOMUXC_SMUXC_PEPDC_D6).....1805
35.4.116 SW_MUX_CTL_PAD_EPDC_D7 (IOMUXC_SMUXC_PEPDC_D7).....1806
35.4.117 SW_MUX_CTL_PAD_EPDC_D8 (IOMUXC_SMUXC_PEPDC_D8).....1807
35.4.118 SW_MUX_CTL_PAD_EPDC_D9 (IOMUXC_SMUXC_PEPDC_D9).....1808
35.4.119 SW_MUX_CTL_PAD_EPDC_D10 (IOMUXC_SMUXC_PEPDC_D10).....1809
35.4.120 SW_MUX_CTL_PAD_EPDC_D11 (IOMUXC_SMUXC_PEPDC_D11).....1810
35.4.121 SW_MUX_CTL_PAD_EPDC_D12 (IOMUXC_SMUXC_PEPDC_D12).....1811
35.4.122 SW_MUX_CTL_PAD_EPDC_D13 (IOMUXC_SMUXC_PEPDC_D13).....1812
35.4.123 SW_MUX_CTL_PAD_EPDC_D14 (IOMUXC_SMUXC_PEPDC_D14).....1813
35.4.124 SW_MUX_CTL_PAD_EPDC_D15 (IOMUXC_SMUXC_PEPDC_D15).....1814
35.4.125 SW_MUX_CTL_PAD_EPDC_GDCLK (IOMUXC_SMUXC_PEPDC_GDCLK).....1815
35.4.126 SW_MUX_CTL_PAD_EPDC_GDSP (IOMUXC_SMUXC_PEPDC_GDSP).....1816
35.4.127 SW_MUX_CTL_PAD_EPDC_GDOE (IOMUXC_SMUXC_PEPDC_GDOE).....1817
35.4.128 SW_MUX_CTL_PAD_EPDC_GDRL (IOMUXC_SMUXC_PEPDC_GDRL).....1818
35.4.129 SW_MUX_CTL_PAD_EPDC_SDCLK (IOMUXC_SMUXC_PEPDC_SDCLK).....1819
35.4.130 SW_MUX_CTL_PAD_EPDC_SDOEZ (IOMUXC_SMUXC_PEPDC_SDOEZ).....1820
35.4.131 SW_MUX_CTL_PAD_EPDC_SDOED (IOMUXC_SMUXC_PEPDC_SDOED).....1821
35.4.132 OMUXC_SW_MUX_CTL_PAD_EPDC_SDOE (IOMUXC_OMUXC_SMUXC_PEPDC_SDOE).....1822
35.4.133 SW_MUX_CTL_PAD_EPDC_SDLE (IOMUXC_SMUXC_PEPDC_SDLE).....1823

35.4.134 SW_MUX_CTL_PAD_EPDC_SDCLKN (IOMUXC_SMUXC_PEPDC_SDCLKN)....1824
35.4.135 SW_MUX_CTL_PAD_EPDC_SDSHR (IOMUXC_SMUXC_PEPDC_SDSHR)....1825
35.4.136 SW_MUX_CTL_PAD_EPDC_PWRCOM (IOMUXC_SMUXC_PEPDC_PWRCOM)....1826
35.4.137 SW_MUX_CTL_PAD_EPDC_PWRSTAT (IOMUXC_SMUXC_PEPDC_PWRSTAT)....1827
35.4.138 SW_MUX_CTL_PAD_EPDC_PWRCTRL0 (IOMUXC_SMUXC_PEPDC_PWRCTRL0)....1828
35.4.139 OMUXC_SW_MUX_CTL_PAD_EPDC_PWRCTRL1 (IOMUXC_OMUXC_SMUXC_PEPDC_PWRCTRL1)....1829
35.4.140 SW_MUX_CTL_PAD_EPDC_PWRCTRL2 (IOMUXC_SMUXC_PEPDC_PWRCTRL2)....1830
35.4.141 SW_MUX_CTL_PAD_EPDC_PWRCTRL3 (IOMUXC_SMUXC_PEPDC_PWRCTRL3)....1831
35.4.142 SW_MUX_CTL_PAD_EPDC_VCOM0 (IOMUXC_SMUXC_PEPDC_VCOM0)....1832
35.4.143 SW_MUX_CTL_PAD_EPDC_VCOM1 (IOMUXC_SMUXC_PEPDC_VCOM1)....1833
35.4.144 SW_MUX_CTL_PAD_EPDC_BDR0 (IOMUXC_SMUXC_PEPDC_BDR0)....1834
35.4.145 SW_MUX_CTL_PAD_EPDC_BDR1 (IOMUXC_SMUXC_PEPDC_BDR1)....1835
35.4.146 SW_MUX_CTL_PAD_EPDC_SDCE0 (IOMUXC_SMUXC_PEPDC_SDCE0)....1836
35.4.147 SW_MUX_CTL_PAD_EPDC_SDCE1 (IOMUXC_SMUXC_PEPDC_SDCE1)....1837
35.4.148 SW_MUX_CTL_PAD_EPDC_SDCE2 (IOMUXC_SMUXC_PEPDC_SDCE2)....1838
35.4.149 SW_MUX_CTL_PAD_EPDC_SDCE3 (IOMUXC_SMUXC_PEPDC_SDCE3)....1839
35.4.150 SW_MUX_CTL_PAD_EPDC_SDCE4 (IOMUXC_SMUXC_PEPDC_SDCE4)....1840
35.4.151 SW_MUX_CTL_PAD_EPDC_SDCE5 (IOMUXC_SMUXC_PEPDC_SDCE5)....1841
35.4.152 SW_MUX_CTL_PAD_EIM_DA0 (IOMUXC_SMUXC_PEIM_DA0)....1842
35.4.153 SW_MUX_CTL_PAD_EIM_DA1 (IOMUXC_SMUXC_PEIM_DA1)....1843
35.4.154 SW_MUX_CTL_PAD_EIM_DA2 (IOMUXC_SMUXC_PEIM_DA2)....1844
35.4.155 SW_MUX_CTL_PAD_EIM_DA3 (IOMUXC_SMUXC_PEIM_DA3)....1845
35.4.156 SW_MUX_CTL_PAD_EIM_DA4 (IOMUXC_SMUXC_PEIM_DA4)....1846
35.4.157 SW_MUX_CTL_PAD_EIM_DA5 (IOMUXC_SMUXC_PEIM_DA5)....1847
35.4.158 SW_MUX_CTL_PAD_EIM_DA6 (IOMUXC_SMUXC_PEIM_DA6)....1848
35.4.159 SW_MUX_CTL_PAD_EIM_DA7 (IOMUXC_SMUXC_PEIM_DA7)....1849
35.4.160 SW_MUX_CTL_PAD_EIM_DA8 (IOMUXC_SMUXC_PEIM_DA8)....1850
35.4.161 SW_MUX_CTL_PAD_EIM_DA9 (IOMUXC_SMUXC_PEIM_DA9)....1851

35.4.162 SW_MUX_CTL_PAD_EIM_DA10 (IOMUXC_SMUXC_PEIM_DA10)....1852
35.4.163 SW_MUX_CTL_PAD_EIM_DA11 (IOMUXC_SMUXC_PEIM_DA11)....1853
35.4.164 SW_MUX_CTL_PAD_EIM_DA12 (IOMUXC_SMUXC_PEIM_DA12)....1854
35.4.165 SW_MUX_CTL_PAD_EIM_DA13 (IOMUXC_SMUXC_PEIM_DA13)....1855
35.4.166 SW_MUX_CTL_PAD_EIM_DA14 (IOMUXC_SMUXC_PEIM_DA14)....1856
35.4.167 SW_MUX_CTL_PAD_EIM_DA15 (IOMUXC_SMUXC_PEIM_DA15)....1857
35.4.168 SW_MUX_CTL_PAD_EIM_CS2 (IOMUXC_SMUXC_PEIM_CS2)....1858
35.4.169 SW_MUX_CTL_PAD_EIM_CS1 (IOMUXC_SMUXC_PEIM_CS1)....1859
35.4.170 SW_MUX_CTL_PAD_EIM_CS0 (IOMUXC_SMUXC_PEIM_CS0)....1860
35.4.171 SW_MUX_CTL_PAD_EIM_EB0 (IOMUXC_SMUXC_PEIM_EB0)....1861
35.4.172 SW_MUX_CTL_PAD_EIM_EB1 (IOMUXC_SMUXC_PEIM_EB1)....1862
35.4.173 SW_MUX_CTL_PAD_EIM_WAIT (IOMUXC_SMUXC_PEIM_WAIT)....1863
35.4.174 SW_MUX_CTL_PAD_EIM_BCLK (IOMUXC_SMUXC_PEIM_BCLK)....1864
35.4.175 SW_MUX_CTL_PAD_EIM_RDY (IOMUXC_SMUXC_PEIM_RDY)....1865
35.4.176 SW_MUX_CTL_PAD_EIM_OE (IOMUXC_SMUXC_PEIM_OE)....1866
35.4.177 SW_MUX_CTL_PAD_EIM_RW (IOMUXC_SMUXC_PEIM_RW)....1867
35.4.178 SW_MUX_CTL_PAD_EIM_LBA (IOMUXC_SMUXC_PEIM_LBA)....1868
35.4.179 SW_MUX_CTL_PAD_EIM_CRE (IOMUXC_SMUXC_PEIM_CRE)....1869
35.4.180 SW_PAD_CTL_PAD_KEY_COL0 (IOMUXC_SPADC_PKC0)....1870
35.4.181 SW_PAD_CTL_PAD_KEY_ROW0 (IOMUXC_SPADC_PKR0)....1871
35.4.182 SW_PAD_CTL_PAD_KEY_COL1 (IOMUXC_SPADC_PKC1)....1873
35.4.183 SW_PAD_CTL_PAD_KEY_ROW1 (IOMUXC_SPADC_PKR1)....1874
35.4.184 SW_PAD_CTL_PAD_KEY_COL2 (IOMUXC_SPADC_PKC2)....1876
35.4.185 SW_PAD_CTL_PAD_KEY_ROW2 (IOMUXC_SPADC_PKR2)....1877
35.4.186 SW_PAD_CTL_PAD_KEY_COL3 (IOMUXC_SPADC_PKC3)....1879
35.4.187 SW_PAD_CTL_PAD_KEY_ROW3 (IOMUXC_SPADC_PKR3)....1880
35.4.188 SW_PAD_CTL_PAD_I2C1_SCL (IOMUXC_SPADC_PI2C1_SCL)....1882
35.4.189 SW_PAD_CTL_PAD_I2C1_SDA (IOMUXC_SPADC_PI2C1_SDA)....1883
35.4.190 SW_PAD_CTL_PAD_I2C2_SCL (IOMUXC_SPADC_PI2C2_SCL)....1885

35.4.191 SW_PAD_CTL_PAD_I2C2_SDA (IOMUXC_SPADC_PI2C2_SDA)....1886
35.4.192 SW_PAD_CTL_PAD_I2C3_SCL (IOMUXC_SPADC_PI2C3_SCL)....1888
35.4.193 SW_PAD_CTL_PAD_I2C3_SDA (IOMUXC_SPADC_PI2C3_SDA)....1889
35.4.194 SW_PAD_CTL_PAD_PWM1 (IOMUXC_SPADC_PPWM1)....1891
35.4.195 SW_PAD_CTL_PAD_PWM2 (IOMUXC_SPADC_PPWM2)....1892
35.4.196 SW_PAD_CTL_PAD_OWIRE (IOMUXC_SPADC_POWIRE)....1894
35.4.197 SW_PAD_CTL_PAD_EPITO (IOMUXC_SPADC_PEPITO)....1895
35.4.198 SW_PAD_CTL_PAD_WDOG (IOMUXC_SPADC_PWDOG)....1897
35.4.199 SW_PAD_CTL_PAD_SSI_TXFS (IOMUXC_SPADC_PSSI_TXFS)....1898
35.4.200 SW_PAD_CTL_PAD_SSI_TXC (IOMUXC_SPADC_PSSI_TXC)....1900
35.4.201 SW_PAD_CTL_PAD_SSI_TXD (IOMUXC_SPADC_PSSI_TXD)....1901
35.4.202 SW_PAD_CTL_PAD_SSI_RXD (IOMUXC_SPADC_PSSI_RXD)....1903
35.4.203 SW_PAD_CTL_PAD_SSI_RXFS (IOMUXC_SPADC_PSSI_RXFS)....1904
35.4.204 SW_PAD_CTL_PAD_SSI_RXC (IOMUXC_SPADC_PSSI_RXC)....1906
35.4.205 SW_PAD_CTL_PAD_UART1_TXD (IOMUXC_SPADC_PUART1_TXD)....1907
35.4.206 SW_PAD_CTL_PAD_UART1_RXD (IOMUXC_SPADC_PUART1_RXD)....1909
35.4.207 SW_PAD_CTL_PAD_UART1_CTS (IOMUXC_SPADC_PUART1_CTS)....1910
35.4.208 SW_PAD_CTL_PAD_UART1_RTS (IOMUXC_SPADC_PUART1_RTS)....1912
35.4.209 SW_PAD_CTL_PAD_UART2_TXD (IOMUXC_SPADC_PUART2_TXD)....1913
35.4.210 SW_PAD_CTL_PAD_UART2_RXD (IOMUXC_SPADC_PUART2_RXD)....1915
35.4.211 SW_PAD_CTL_PAD_UART2_CTS (IOMUXC_SPADC_PUART2_CTS)....1916
35.4.212 SW_PAD_CTL_PAD_UART2_RTS (IOMUXC_SPADC_PUART2_RTS)....1918
35.4.213 SW_PAD_CTL_PAD_UART3_TXD (IOMUXC_SPADC_PUART3_TXD)....1919
35.4.214 SW_PAD_CTL_PAD_UART3_RXD (IOMUXC_SPADC_PUART3_RXD)....1921
35.4.215 SW_PAD_CTL_PAD_UART4_TXD (IOMUXC_SPADC_PUART4_TXD)....1922
35.4.216 SW_PAD_CTL_PAD_UART4_RXD (IOMUXC_SPADC_PUART4_RXD)....1924
35.4.217 SW_PAD_CTL_PAD_CSPI_SCLK (IOMUXC_SPADC_PCSPI_SCLK)....1925
35.4.218 SW_PAD_CTL_PAD_CSPI_MOSI (IOMUXC_SPADC_PCSPI_MOSI)....1927
35.4.219 SW_PAD_CTL_PAD_CSPI_MISO (IOMUXC_SPADC_PCSPI_MISO)....1928

35.4.220 SW_PAD_CTL_PAD_CSPI_SS0 (IOMUXC_SPADC_PCSPI_SS0)....1930
35.4.221 SW_PAD_CTL_PAD_ECSPI1_SCLK (IOMUXC_SPADC_PECSPI1_SCLK)....1931
35.4.222 SW_PAD_CTL_PAD_ECSPI1_MOSI (IOMUXC_SPADC_PECSPI1_MOSI)....1933
35.4.223 SW_PAD_CTL_PAD_ECSPI1_MISO (IOMUXC_SPADC_PECSPI1_MISO)....1934
35.4.224 SW_PAD_CTL_PAD_ECSPI1_SS0 (IOMUXC_SPADC_PECSPI1_SS0)....1936
35.4.225 SW_PAD_CTL_PAD_ECSPI2_SCLK (IOMUXC_SPADC_PECSPI2_SCLK)....1937
35.4.226 SW_PAD_CTL_PAD_ECSPI2_MOSI (IOMUXC_SPADC_PECSPI2_MOSI)....1939
35.4.227 SW_PAD_CTL_PAD_ECSPI2_MISO (IOMUXC_SPADC_PECSPI2_MISO)....1940
35.4.228 SW_PAD_CTL_PAD_ECSPI2_SS0 (IOMUXC_SPADC_PECSPI2_SS0)....1942
35.4.229 SW_PAD_CTL_PAD_SD1_CLK (IOMUXC_SPADC_PSD1_CLK)....1943
35.4.230 SW_PAD_CTL_PAD_SD1_CMD (IOMUXC_SPADC_PSD1_CMD)....1945
35.4.231 SW_PAD_CTL_PAD_SD1_D0 (IOMUXC_SPADC_PSD1_D0)....1946
35.4.232 SW_PAD_CTL_PAD_SD1_D1 (IOMUXC_SPADC_PSD1_D1)....1948
35.4.233 SW_PAD_CTL_PAD_SD1_D2 (IOMUXC_SPADC_PSD1_D2)....1949
35.4.234 SW_PAD_CTL_PAD_SD1_D3 (IOMUXC_SPADC_PSD1_D3)....1951
35.4.235 SW_PAD_CTL_PAD_SD2_CLK (IOMUXC_SPADC_PSD2_CLK)....1952
35.4.236 SW_PAD_CTL_PAD_SD2_CMD (IOMUXC_SPADC_PSD2_CMD)....1954
35.4.237 SW_PAD_CTL_PAD_SD2_D0 (IOMUXC_SPADC_PSD2_D0)....1955
35.4.238 SW_PAD_CTL_PAD_SD2_D1 (IOMUXC_SPADC_PSD2_D1)....1957
35.4.239 SW_PAD_CTL_PAD_SD2_D2 (IOMUXC_SPADC_PSD2_D2)....1958
35.4.240 SW_PAD_CTL_PAD_SD2_D3 (IOMUXC_SPADC_PSD2_D3)....1960
35.4.241 SW_PAD_CTL_PAD_SD2_D4 (IOMUXC_SPADC_PSD2_D4)....1961
35.4.242 SW_PAD_CTL_PAD_SD2_D5 (IOMUXC_SPADC_PSD2_D5)....1963
35.4.243 SW_PAD_CTL_PAD_SD2_D6 (IOMUXC_SPADC_PSD2_D6)....1964
35.4.244 SW_PAD_CTL_PAD_SD2_D7 (IOMUXC_SPADC_PSD2_D7)....1966
35.4.245 SW_PAD_CTL_PAD_SD2_WP (IOMUXC_SPADC_PSD2_WP)....1967
35.4.246 SW_PAD_CTL_PAD_SD2_CD (IOMUXC_SPADC_PSD2_CD)....1969
35.4.247 SW_PAD_CTL_PAD_PMIC_ON_REQ (IOMUXC_SPADC_PPMIC_ON_REQ)....1970
35.4.248 SW_PAD_CTL_PAD_PMIC_STBY_REQ (IOMUXC_SPADC_PPMIC_STBY_REQ)....1971

35.4.249 SW_PAD_CTL_PAD_POR_B (IOMUXC_SPADC_PPOR_B)....1972
35.4.250 SW_PAD_CTL_PAD_BOOT_MODE1 (IOMUXC_SPADC_PBOOT_MODE1)....1972
35.4.251 SW_PAD_CTL_PAD_RESET_IN_B (IOMUXC_SPADC_PRESET_IN_B)....1973
35.4.252 SW_PAD_CTL_PAD_BOOT_MODE0 (IOMUXC_SPADC_PBOOT_MODE0)....1974
35.4.253 SW_PAD_CTL_PAD_TEST_MODE (IOMUXC_SPADC_PTEST_MODE)....1974
35.4.254 SW_PAD_CTL_PAD_JTAG_TMS (IOMUXC_SPADC_PJTAG_TMS)....1975
35.4.255 SW_PAD_CTL_PAD_JTAG_MOD (IOMUXC_SPADC_PJTAG_MOD)....1976
35.4.256 SW_PAD_CTL_PAD_JTAG_TRSTB (IOMUXC_SPADC_PJTAG_TRSTB)....1976
35.4.257 SW_PAD_CTL_PAD_JTAG_TDI (IOMUXC_SPADC_PJTAG_TDI)....1977
35.4.258 SW_PAD_CTL_PAD_JTAG_TCK (IOMUXC_SPADC_PJTAG_TCK)....1978
35.4.259 SW_PAD_CTL_PAD_JTAG_TDO (IOMUXC_SPADC_PJTAG_TDO)....1978
35.4.260 SW_PAD_CTL_PAD_DISP_D0 (IOMUXC_SPADC_PDISP_D0)....1980
35.4.261 SW_PAD_CTL_PAD_DISP_D1 (IOMUXC_SPADC_PDISP_D1)....1981
35.4.262 SW_PAD_CTL_PAD_DISP_D2 (IOMUXC_SPADC_PDISP_D2)....1983
35.4.263 SW_PAD_CTL_PAD_DISP_D3 (IOMUXC_SPADC_PDISP_D3)....1984
35.4.264 SW_PAD_CTL_PAD_DISP_D4 (IOMUXC_SPADC_PDISP_D4)....1986
35.4.265 SW_PAD_CTL_PAD_DISP_D5 (IOMUXC_SPADC_PDISP_D5)....1987
35.4.266 SW_PAD_CTL_PAD_DISP_D6 (IOMUXC_SPADC_PDISP_D6)....1989
35.4.267 SW_PAD_CTL_PAD_DISP_D7 (IOMUXC_SPADC_PDISP_D7)....1990
35.4.268 SW_PAD_CTL_PAD_DISP_WR (IOMUXC_SPADC_PDISP_WR)....1992
35.4.269 SW_PAD_CTL_PAD_DISP_RD (IOMUXC_SPADC_PDISP_RD)....1993
35.4.270 SW_PAD_CTL_PAD_DISP_RS (IOMUXC_SPADC_PDISP_RS)....1995
35.4.271 SW_PAD_CTL_PAD_DISP_CS (IOMUXC_SPADC_PDISP_CS)....1996
35.4.272 SW_PAD_CTL_PAD_DISP_BUSY (IOMUXC_SPADC_PDISP_BUSY)....1998
35.4.273 SW_PAD_CTL_PAD_DISP_RESET (IOMUXC_SPADC_PDISP_RESET)....1999
35.4.274 SW_PAD_CTL_PAD_SD3_CMD (IOMUXC_SPADC_PSD3_CMD)....2001
35.4.275 SW_PAD_CTL_PAD_SD3_CLK (IOMUXC_SPADC_PSD3_CLK)....2002
35.4.276 SW_PAD_CTL_PAD_SD3_D0 (IOMUXC_SPADC_PSD3_D0)....2004
35.4.277 SW_PAD_CTL_PAD_SD3_D1 (IOMUXC_SPADC_PSD3_D1)....2005

35.4.278 SW_PAD_CTL_PAD_SD3_D2 (IOMUXC_SPADC_PSD3_D2)....2007
35.4.279 SW_PAD_CTL_PAD_SD3_D3 (IOMUXC_SPADC_PSD3_D3)....2008
35.4.280 SW_PAD_CTL_PAD_SD3_D4 (IOMUXC_SPADC_PSD3_D4)....2010
35.4.281 SW_PAD_CTL_PAD_SD3_D5 (IOMUXC_SPADC_PSD3_D5)....2011
35.4.282 SW_PAD_CTL_PAD_SD3_D6 (IOMUXC_SPADC_PSD3_D6)....2013
35.4.283 SW_PAD_CTL_PAD_SD3_D7 (IOMUXC_SPADC_PSD3_D7)....2014
35.4.284 SW_PAD_CTL_PAD_SD3_WP (IOMUXC_SPADC_PSD3_WP)....2016
35.4.285 SW_PAD_CTL_PAD_DISP_D8 (IOMUXC_SPADC_PDISP_D8)....2017
35.4.286 SW_PAD_CTL_PAD_DISP_D9 (IOMUXC_SPADC_PDISP_D9)....2019
35.4.287 SW_PAD_CTL_PAD_DISP_D10 (IOMUXC_SPADC_PDISP_D10)....2020
35.4.288 SW_PAD_CTL_PAD_DISP_D11 (IOMUXC_SPADC_PDISP_D11)....2022
35.4.289 SW_PAD_CTL_PAD_DISP_D12 (IOMUXC_SPADC_PDISP_D12)....2023
35.4.290 SW_PAD_CTL_PAD_DISP_D13 (IOMUXC_SPADC_PDISP_D13)....2025
35.4.291 SW_PAD_CTL_PAD_DISP_D14 (IOMUXC_SPADC_PDISP_D14)....2026
35.4.292 SW_PAD_CTL_PAD_DISP_D15 (IOMUXC_SPADC_PDISP_D15)....2028
35.4.293 SW_PAD_CTL_PAD_DRAM_OPEN (IOMUXC_SPADC_PDRAM_OPEN)....2029
35.4.294 SW_PAD_CTL_PAD_DRAM_OPENFB (IOMUXC_SPADC_PDRAM_OPENFB)....2030
35.4.295 SW_PAD_CTL_PAD_DRAM_SDCLK_1 (IOMUXC_SPADC_PDRAM_SDCLK_1)....2030
35.4.296 SW_PAD_CTL_PAD_DRAM_SDCLK_0 (IOMUXC_SPADC_PDRAM_SDCLK_0)....2031
35.4.297 SW_PAD_CTL_PAD_DRAM_SDCKE (IOMUXC_SPADC_PDRAM_SDCKE)....2032
35.4.298 SW_PAD_CTL_PAD_DRAM_SDODT0 (IOMUXC_SPADC_PDRAM_SDODT0)....2033
35.4.299 SW_PAD_CTL_PAD_DRAM_D16 (IOMUXC_SPADC_PDRAM_D16)....2034
35.4.300 SW_PAD_CTL_PAD_DRAM_D17 (IOMUXC_SPADC_PDRAM_D17)....2035
35.4.301 SW_PAD_CTL_PAD_DRAM_D18 (IOMUXC_SPADC_PDRAM_D18)....2035
35.4.302 SW_PAD_CTL_PAD_DRAM_D19 (IOMUXC_SPADC_PDRAM_D19)....2036
35.4.303 SW_PAD_CTL_PAD_DRAM_D20 (IOMUXC_SPADC_PDRAM_D20)....2036
35.4.304 SW_PAD_CTL_PAD_DRAM_D21 (IOMUXC_SPADC_PDRAM_D21)....2037
35.4.305 SW_PAD_CTL_PAD_DRAM_D22 (IOMUXC_SPADC_PDRAM_D22)....2038
35.4.306 SW_PAD_CTL_PAD_DRAM_D23 (IOMUXC_SPADC_PDRAM_D23)....2038

35.4.307 SW_PAD_CTL_PAD_DRAM_DQM2 (IOMUXC_SPADC_PDRAM_DQM2).....2039
35.4.308 SW_PAD_CTL_PAD_DRAM_SDQS2 (IOMUXC_SPADC_PDRAM_SDQS2).....2040
35.4.309 SW_PAD_CTL_PAD_DRAM_D0 (IOMUXC_SPADC_PDRAM_D0).....2041
35.4.310 SW_PAD_CTL_PAD_DRAM_D1 (IOMUXC_SPADC_PDRAM_D1).....2041
35.4.311 SW_PAD_CTL_PAD_DRAM_D2 (IOMUXC_SPADC_PDRAM_D2).....2042
35.4.312 SW_PAD_CTL_PAD_DRAM_D3 (IOMUXC_SPADC_PDRAM_D3).....2042
35.4.313 SW_PAD_CTL_PAD_DRAM_D4 (IOMUXC_SPADC_PDRAM_D4).....2043
35.4.314 SW_PAD_CTL_PAD_DRAM_D5 (IOMUXC_SPADC_PDRAM_D5).....2044
35.4.315 SW_PAD_CTL_PAD_DRAM_D6 (IOMUXC_SPADC_PDRAM_D6).....2044
35.4.316 SW_PAD_CTL_PAD_DRAM_D7 (IOMUXC_SPADC_PDRAM_D7).....2045
35.4.317 SW_PAD_CTL_PAD_DRAM_DQM0 (IOMUXC_SPADC_PDRAM_DQM0).....2045
35.4.318 SW_PAD_CTL_PAD_DRAM_SDQS0 (IOMUXC_SPADC_PDRAM_SDQS0).....2046
35.4.319 SW_PAD_CTL_PAD_DRAM_SDODT1 (IOMUXC_SPADC_PDRAM_SDODT1).....2047
35.4.320 SW_PAD_CTL_PAD_DRAM_SDQS1 (IOMUXC_SPADC_PDRAM_SDQS1).....2048
35.4.321 SW_PAD_CTL_PAD_DRAM_DQM1 (IOMUXC_SPADC_PDRAM_DQM1).....2049
35.4.322 SW_PAD_CTL_PAD_DRAM_D8 (IOMUXC_SPADC_PDRAM_D8).....2050
35.4.323 SW_PAD_CTL_PAD_DRAM_D9 (IOMUXC_SPADC_PDRAM_D9).....2051
35.4.324 SW_PAD_CTL_PAD_DRAM_D10 (IOMUXC_SPADC_PDRAM_D10).....2051
35.4.325 SW_PAD_CTL_PAD_DRAM_D11 (IOMUXC_SPADC_PDRAM_D11).....2052
35.4.326 SW_PAD_CTL_PAD_DRAM_D12 (IOMUXC_SPADC_PDRAM_D12).....2053
35.4.327 SW_PAD_CTL_PAD_DRAM_D13 (IOMUXC_SPADC_PDRAM_D13).....2053
35.4.328 SW_PAD_CTL_PAD_DRAM_D14 (IOMUXC_SPADC_PDRAM_D14).....2054
35.4.329 SW_PAD_CTL_PAD_DRAM_D15 (IOMUXC_SPADC_PDRAM_D15).....2054
35.4.330 SW_PAD_CTL_PAD_DRAM_SDQS3 (IOMUXC_SPADC_PDRAM_SDQS3).....2055
35.4.331 SW_PAD_CTL_PAD_DRAM_DQM3 (IOMUXC_SPADC_PDRAM_DQM3).....2056
35.4.332 SW_PAD_CTL_PAD_DRAM_D24 (IOMUXC_SPADC_PDRAM_D24).....2057
35.4.333 SW_PAD_CTL_PAD_DRAM_D25 (IOMUXC_SPADC_PDRAM_D25).....2057
35.4.334 SW_PAD_CTL_PAD_DRAM_D26 (IOMUXC_SPADC_PDRAM_D26).....2058
35.4.335 SW_PAD_CTL_PAD_DRAM_D27 (IOMUXC_SPADC_PDRAM_D27).....2059

35.4.336 SW_PAD_CTL_PAD_DRAM_D28 (IOMUXC_SPADC_PDRAM_D28).....2059
35.4.337 SW_PAD_CTL_PAD_DRAM_D29 (IOMUXC_SPADC_PDRAM_D29).....2060
35.4.338 SW_PAD_CTL_PAD_DRAM_D30 (IOMUXC_SPADC_PDRAM_D30).....2060
35.4.339 SW_PAD_CTL_PAD_DRAM_D31 (IOMUXC_SPADC_PDRAM_D31).....2061
35.4.340 SW_PAD_CTL_PAD_EPDC_D0 (IOMUXC_SPADC_PEPDC_D0).....2062
35.4.341 SW_PAD_CTL_PAD_EPDC_D1 (IOMUXC_SPADC_PEPDC_D1).....2063
35.4.342 SW_PAD_CTL_PAD_EPDC_D2 (IOMUXC_SPADC_PEPDC_D2).....2065
35.4.343 SW_PAD_CTL_PAD_EPDC_D3 (IOMUXC_SPADC_PEPDC_D3).....2066
35.4.344 SW_PAD_CTL_PAD_EPDC_D4 (IOMUXC_SPADC_PEPDC_D4).....2068
35.4.345 SW_PAD_CTL_PAD_EPDC_D5 (IOMUXC_SPADC_PEPDC_D5).....2069
35.4.346 SW_PAD_CTL_PAD_EPDC_D6 (IOMUXC_SPADC_PEPDC_D6).....2071
35.4.347 SW_PAD_CTL_PAD_EPDC_D7 (IOMUXC_SPADC_PEPDC_D7).....2072
35.4.348 SW_PAD_CTL_PAD_EPDC_D8 (IOMUXC_SPADC_PEPDC_D8).....2074
35.4.349 SW_PAD_CTL_PAD_EPDC_D9 (IOMUXC_SPADC_PEPDC_D9).....2075
35.4.350 SW_PAD_CTL_PAD_EPDC_D10 (IOMUXC_SPADC_PEPDC_D10).....2077
35.4.351 SW_PAD_CTL_PAD_EPDC_D11 (IOMUXC_SPADC_PEPDC_D11).....2078
35.4.352 SW_PAD_CTL_PAD_EPDC_D12 (IOMUXC_SPADC_PEPDC_D12).....2080
35.4.353 SW_PAD_CTL_PAD_EPDC_D13 (IOMUXC_SPADC_PEPDC_D13).....2081
35.4.354 SW_PAD_CTL_PAD_EPDC_D14 (IOMUXC_SPADC_PEPDC_D14).....2083
35.4.355 SW_PAD_CTL_PAD_EPDC_D15 (IOMUXC_SPADC_PEPDC_D15).....2084
35.4.356 SW_PAD_CTL_PAD_EPDC_GDCLK (IOMUXC_SPADC_PEPDC_GDCLK).....2086
35.4.357 SW_PAD_CTL_PAD_EPDC_GDSP (IOMUXC_SPADC_PEPDC_GDSP).....2087
35.4.358 SW_PAD_CTL_PAD_EPDC_GDOE (IOMUXC_SPADC_PEPDC_GDOE).....2089
35.4.359 SW_PAD_CTL_PAD_EPDC_GDRL (IOMUXC_SPADC_PEPDC_GDRL).....2090
35.4.360 SW_PAD_CTL_PAD_EPDC_SDCLK (IOMUXC_SPADC_PEPDC_SDCLK).....2092
35.4.361 SW_PAD_CTL_PAD_EPDC_SDOEZ (IOMUXC_SPADC_PEPDC_SDOEZ).....2093
35.4.362 SW_PAD_CTL_PAD_EPDC_SDOED (IOMUXC_SPADC_PEPDC_SDOED).....2095
35.4.363 SW_PAD_CTL_PAD_EPDC_SDOE (IOMUXC_SPADC_PEPDC_SDOE).....2096
35.4.364 SW_PAD_CTL_PAD_EPDC_SDLE (IOMUXC_SPADC_PEPDC_SDLE).....2098

35.4.365 SW_PAD_CTL_PAD_EPDC_SDCLKN (IOMUXC_SPADC_PEPDC_SDCLKN).....2099
35.4.366 SW_PAD_CTL_PAD_EPDC_SDSHR (IOMUXC_SPADC_PEPDC_SDSHR).....2101
35.4.367 SW_PAD_CTL_PAD_EPDC_PWRCOM (IOMUXC_SPADC_PEPDC_PWRCOM).....2102
35.4.368 SW_PAD_CTL_PAD_EPDC_PWRSTAT (IOMUXC_SPADC_PEPDC_PWRSTAT).....2104
35.4.369 SW_PAD_CTL_PAD_EPDC_PWRCTRL0 (IOMUXC_SPADC_PEPDC_PWRCTRL0).....2105
35.4.370 SW_PAD_CTL_PAD_EPDC_PWRCTRL1 (IOMUXC_SPADC_PEPDC_PWRCTRL1).....2107
35.4.371 SW_PAD_CTL_PAD_EPDC_PWRCTRL2 (IOMUXC_SPADC_PEPDC_PWRCTRL2).....2108
35.4.372 SW_PAD_CTL_PAD_EPDC_PWRCTRL3 (IOMUXC_SPADC_PEPDC_PWRCTRL3).....2110
35.4.373 SW_PAD_CTL_PAD_EPDC_VCOM0 (IOMUXC_SPADC_PEPDC_VCOM0).....2111
35.4.374 SW_PAD_CTL_PAD_EPDC_VCOM1 (IOMUXC_SPADC_PEPDC_VCOM1).....2113
35.4.375 SW_PAD_CTL_PAD_EPDC_BDR0 (IOMUXC_SPADC_PEPDC_BDR0).....2114
35.4.376 SW_PAD_CTL_PAD_EPDC_BDR1 (IOMUXC_SPADC_PEPDC_BDR1).....2116
35.4.377 SW_PAD_CTL_PAD_EPDC_SDCE0 (IOMUXC_SPADC_PEPDC_SDCE0).....2117
35.4.378 SW_PAD_CTL_PAD_EPDC_SDCE1 (IOMUXC_SPADC_PEPDC_SDCE1).....2119
35.4.379 SW_PAD_CTL_PAD_EPDC_SDCE2 (IOMUXC_SPADC_PEPDC_SDCE2).....2120
35.4.380 SW_PAD_CTL_PAD_EPDC_SDCE3 (IOMUXC_SPADC_PEPDC_SDCE3).....2122
35.4.381 SW_PAD_CTL_PAD_EPDC_SDCE4 (IOMUXC_SPADC_PEPDC_SDCE4).....2123
35.4.382 SW_PAD_CTL_PAD_EPDC_SDCE5 (IOMUXC_SPADC_PEPDC_SDCE5).....2125
35.4.383 SW_PAD_CTL_PAD_EIM_DA0 (IOMUXC_SPADC_PEIM_DA0).....2126
35.4.384 SW_PAD_CTL_PAD_EIM_DA1 (IOMUXC_SPADC_PEIM_DA1).....2128
35.4.385 SW_PAD_CTL_PAD_EIM_DA2 (IOMUXC_SPADC_PEIM_DA2).....2129
35.4.386 SW_PAD_CTL_PAD_EIM_DA3 (IOMUXC_SPADC_PEIM_DA3).....2131
35.4.387 SW_PAD_CTL_PAD_EIM_DA4 (IOMUXC_SPADC_PEIM_DA4).....2132
35.4.388 SW_PAD_CTL_PAD_EIM_DA5 (IOMUXC_SPADC_PEIM_DA5).....2134
35.4.389 SW_PAD_CTL_PAD_EIM_DA6 (IOMUXC_SPADC_PEIM_DA6).....2135
35.4.390 SW_PAD_CTL_PAD_EIM_DA7 (IOMUXC_SPADC_PEIM_DA7).....2137
35.4.391 SW_PAD_CTL_PAD_EIM_DA8 (IOMUXC_SPADC_PEIM_DA8).....2138
35.4.392 SW_PAD_CTL_PAD_EIM_DA9 (IOMUXC_SPADC_PEIM_DA9).....2140
35.4.393 SW_PAD_CTL_PAD_EIM_DA10 (IOMUXC_SPADC_PEIM_DA10).....2141

35.4.394 SW_PAD_CTL_PAD_EIM_DA11 (IOMUXC_SPADC_PEIM_DA11)....2143
35.4.395 SW_PAD_CTL_PAD_EIM_DA12 (IOMUXC_SPADC_PEIM_DA12)....2144
35.4.396 SW_PAD_CTL_PAD_EIM_DA13 (IOMUXC_SPADC_PEIM_DA13)....2146
35.4.397 SW_PAD_CTL_PAD_EIM_DA14 (IOMUXC_SPADC_PEIM_DA14)....2147
35.4.398 SW_PAD_CTL_PAD_EIM_DA15 (IOMUXC_SPADC_PEIM_DA15)....2149
35.4.399 SW_PAD_CTL_PAD_EIM_CS2 (IOMUXC_SPADC_PEIM_CS2)....2150
35.4.400 SW_PAD_CTL_PAD_EIM_CS1 (IOMUXC_SPADC_PEIM_CS1)....2152
35.4.401 SW_PAD_CTL_PAD_EIM_CS0 (IOMUXC_SPADC_PEIM_CS0)....2153
35.4.402 SW_PAD_CTL_PAD_EIM_EB0 (IOMUXC_SPADC_PEIM_EB0)....2155
35.4.403 SW_PAD_CTL_PAD_EIM_EB1 (IOMUXC_SPADC_PEIM_EB1)....2156
35.4.404 SW_PAD_CTL_PAD_EIM_WAIT (IOMUXC_SPADC_PEIM_WAIT)....2158
35.4.405 SW_PAD_CTL_PAD_EIM_BCLK (IOMUXC_SPADC_PEIM_BCLK)....2159
35.4.406 SW_PAD_CTL_PAD_EIM_RDY (IOMUXC_SPADC_PEIM_RDY)....2161
35.4.407 SW_PAD_CTL_PAD_EIM_OE (IOMUXC_SPADC_PEIM_OE)....2162
35.4.408 SW_PAD_CTL_PAD_EIM_RW (IOMUXC_SPADC_PEIM_RW)....2164
35.4.409 SW_PAD_CTL_PAD_EIM_LBA (IOMUXC_SPADC_PEIM_LBA)....2165
35.4.410 SW_PAD_CTL_PAD_EIM_CRE (IOMUXC_SPADC_PEIM_CRE)....2167
35.4.411 SW_PAD_CTL_GRP_ADDDS (IOMUXC_SPAD_GADDDS)....2168
35.4.412 SW_PAD_CTL_GRP_DDRMODE_CTL (IOMUXC_SPAD_GDDRMODE_CTL)....2169
35.4.413 SW_PAD_CTL_GRP_DDRPKE (IOMUXC_SPAD_GDDRPKE)....2169
35.4.414 SW_PAD_CTL_GRP_EIM (IOMUXC_SPAD_GEIM)....2170
35.4.415 SW_PAD_CTL_GRP_EPDC (IOMUXC_SPAD_GEPDC)....2171
35.4.416 SW_PAD_CTL_GRP_UART (IOMUXC_SPAD_GUART)....2171
35.4.417 SW_PAD_CTL_GRP_DDRPK (IOMUXC_SPAD_GDDRPK)....2172
35.4.418 SW_PAD_CTL_GRP_DDRHYS (IOMUXC_SPAD_GDDRHYS)....2173
35.4.419 SW_PAD_CTL_GRP_KEYPAD (IOMUXC_SPAD_GKEYPAD)....2173
35.4.420 SW_PAD_CTL_GRP_DDRMODE (IOMUXC_SPAD_GDDRMODE)....2174
35.4.421 SW_PAD_CTL_GRP_SSI (IOMUXC_SPAD_GSSI)....2175
35.4.422 SW_PAD_CTL_GRP_SD1 (IOMUXC_SPAD_GSD1)....2175

35.4.423 SW_PAD_CTL_GRP_B0DS (IOMUXC_SPAD_GB0DS).....2176
35.4.424 SW_PAD_CTL_GRP_SD2 (IOMUXC_SPAD_GSD2).....2176
35.4.425 SW_PAD_CTL_GRP_B1DS (IOMUXC_SPAD_GB1DS).....2177
35.4.426 SW_PAD_CTL_GRP_CTLDS (IOMUXC_SPAD_GCTLDS).....2178
35.4.427 SW_PAD_CTL_GRP_B2DS (IOMUXC_SPAD_GB2DS).....2178
35.4.428 SW_PAD_CTL_GRP_DDR_TYPE (IOMUXC_SPAD_GDDR_TYPE).....2179
35.4.429 SW_PAD_CTL_GRP_LCD (IOMUXC_SPAD_GLCD).....2180
35.4.430 SW_PAD_CTL_GRP_B3DS (IOMUXC_SPAD_GB3DS).....2180
35.4.431 SW_PAD_CTL_GRP_MISC (IOMUXC_SPAD_GMISC).....2181
35.4.432 SW_PAD_CTL_GRP_SPI (IOMUXC_SPAD_GSPI).....2182
35.4.433 SW_PAD_CTL_GRP_NANDF (IOMUXC_SPAD_GNANDF).....2182
35.4.434 AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT (IOMUXC_API_DA_AMX_SI).....2183
35.4.435 AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT (IOMUXC_API_DB_AMX_SI).....2183
35.4.436 AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT (IOMUXC_API_RXCLK_AMX_SI).....2184
35.4.437 AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT (IOMUXC_API_RXFS_AMX_SI).....2184
35.4.438 AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT (IOMUXC_API_TXCLK_AMX_SI).....2185
35.4.439 AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT (IOMUXC_API_TXFS_AMX_SI).....2185
35.4.440 CCM_PLL1_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL1_BYPASS_CLK_SI).....2186
35.4.441 CCM_PLL2_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL2_BYPASS_CLK_SI).....2186
35.4.442 CCM_PLL3_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL3_BYPASS_CLK_SI).....2187
35.4.443 CSPI_IPP_IND_DATAREADY_B_SELECT_INPUT (IOMUXC_CSPI_DRDY_SI).....2187
35.4.444 CSPI_IPP_IND_SS1_B_SELECT_INPUT (IOMUXC_CSPI_SS1_SI).....2188
35.4.445 CSPI_IPP_IND_SS2_B_SELECT_INPUT (IOMUXC_CSPI_SS2_SI).....2188
35.4.446 CSPI_IPP_IND_SS3_B_SELECT_INPUT (IOMUXC_CSPI_SS3_SI).....2189
35.4.447 ELCDIF_LCDIF_BUSY_SELECT_INPUT (IOMUXC_ELCDIFL_BUSY_SI).....2189
35.4.448 ELCDIF_LCDIF_RXDATA_0_SELECT_INPUT (IOMUXC_ELCDIFL_R0_SI).....2190
35.4.449 ELCDIF_LCDIF_RXDATA_1_SELECT_INPUT (IOMUXC_ELCDIFL_R1_SI).....2190
35.4.450 ELCDIF_LCDIF_RXDATA_2_SELECT_INPUT (IOMUXC_ELCDIFL_R2_SI).....2191
35.4.451 ELCDIF_LCDIF_RXDATA_3_SELECT_INPUT (IOMUXC_ELCDIFL_R3_SI).....2191

35.4.452 ELCDIF_LCDIF_RXDATA_4_SELECT_INPUT (IOMUXC_ELCDIFL_R4_SI).....2192
35.4.453 ELCDIF_LCDIF_RXDATA_5_SELECT_INPUT (IOMUXC_ELCDIFL_R5_SI).....2192
35.4.454 ELCDIF_LCDIF_RXDATA_6_SELECT_INPUT (IOMUXC_ELCDIFL_R6_SI).....2193
35.4.455 ELCDIF_LCDIF_RXDATA_7_SELECT_INPUT (IOMUXC_ELCDIFL_R7_SI).....2193
35.4.456 ELCDIF_LCDIF_RXDATA_8_SELECT_INPUT (IOMUXC_ELCDIFL_R8_SI).....2194
35.4.457 C_ELCDIF_LCDIF_RXDATA_9_SELECT_INPUT (IOMUXC_C_ELCDIFL_R9_SI).....2194
35.4.458 ELCDIF_LCDIF_RXDATA_10_SELECT_INPUT (IOMUXC_ELCDIFL_R10_SI).....2195
35.4.459 ELCDIF_LCDIF_RXDATA_11_SELECT_INPUT (IOMUXC_ELCDIFL_R11_SI).....2195
35.4.460 ELCDIF_LCDIF_RXDATA_12_SELECT_INPUT (IOMUXC_ELCDIFL_R12_SI).....2196
35.4.461 ELCDIF_LCDIF_RXDATA_13_SELECT_INPUT (IOMUXC_ELCDIFL_R13_SI).....2196
35.4.462 ELCDIF_LCDIF_RXDATA_14_SELECT_INPUT (IOMUXC_ELCDIFL_R14_SI).....2197
35.4.463 ELCDIF_LCDIF_RXDATA_15_SELECT_INPUT (IOMUXC_ELCDIFL_R15_SI).....2197
35.4.464 ELCDIF_VSYNC_I_SELECT_INPUT (IOMUXC_ELCDIF_VSYNC_I_SI).....2198
35.4.465 ESDHC2_IPP_CARD_DET_SELECT_INPUT (IOMUXC_ESDHC2_CDET_SI).....2198
35.4.466 ESDHC2_IPP_WP_ON_SELECT_INPUT (IOMUXC_ESDHC2_WP_ON_SI).....2199
35.4.467 ESDHC4_IPP_CARD_CLK_IN_SELECT_INPUT (IOMUXC_ESDHC4_CCLK_IN_SI).....2199
35.4.468 ESDHC4_IPP_CMD_IN_SELECT_INPUT (IOMUXC_ESDHC4_CMD_IN_SI).....2200
35.4.469 ESDHC4_IPP_DAT0_IN_SELECT_INPUT (IOMUXC_ESDHC4_DAT0_IN_SI).....2200
35.4.470 XC_ESDHC4_IPP_DAT1_IN_SELECT_INPUT (IOMUXC_XC_ESDHC4_1_IN_SI).....2201
35.4.471 ESDHC4_IPP_DAT2_IN_SELECT_INPUT (IOMUXC_ESDHC4_2_IN_SI).....2201
35.4.472 ESDHC4_IPP_DAT3_IN_SELECT_INPUT (IOMUXC_ESDHC4_3_IN_SI).....2202
35.4.473 ESDHC4_IPP_DAT4_IN_SELECT_INPUT (IOMUXC_ESDHC4_4_IN_SI).....2202
35.4.474 ESDHC4_IPP_DAT5_IN_SELECT_INPUT (IOMUXC_ESDHC4_5_IN_SI).....2203
35.4.475 ESDHC4_IPP_DAT6_IN_SELECT_INPUT (IOMUXC_ESDHC4_6_IN_SI).....2203
35.4.476 ESDHC4_IPP_DAT7_IN_SELECT_INPUT (IOMUXC_ESDHC4_7_IN_SI).....2204
35.4.477 FEC_FEC_COL_SELECT_INPUT (IOMUXC_FEC_COL_SI).....2204
35.4.478 FEC_FEC_MDI_SELECT_INPUT (IOMUXC_FEC_MDI_SI).....2205
35.4.479 FEC_FEC_RDATA_0_SELECT_INPUT (IOMUXC_FEC_RD0_SI).....2205
35.4.480 FEC_FEC_RDATA_1_SELECT_INPUT (IOMUXC_FEC_RD1_SI).....2206

35.4.481 FEC_FEC_RX_CLK_SELECT_INPUT (IOMUXC_FEC_RX_CLK_SI)....2206
35.4.482 FEC_FEC_RX_DV_SELECT_INPUT (IOMUXC_FEC_RX_DV_SI)....2207
35.4.483 FEC_FEC_RX_ER_SELECT_INPUT (IOMUXC_FEC_RX_ER_SI)....2207
35.4.484 FEC_FEC_TX_CLK_SELECT_INPUT (IOMUXC_FEC_TX_CLK_SI)....2208
35.4.485 KPP_IPP_IND_COL_4_SELECT_INPUT (IOMUXC_KPP_FC4_SI)....2208
35.4.486 KPP_IPP_IND_COL_5_SELECT_INPUT (IOMUXC_KPP_FC5_SI)....2209
35.4.487 KPP_IPP_IND_COL_6_SELECT_INPUT (IOMUXC_KPP_FC6_SI)....2209
35.4.488 KPP_IPP_IND_COL_7_SELECT_INPUT (IOMUXC_KPP_FC7_SI)....2210
35.4.489 KPP_IPP_IND_ROW_4_SELECT_INPUT (IOMUXC_KPP_FR4_SI)....2210
35.4.490 KPP_IPP_IND_ROW_5_SELECT_INPUT (IOMUXC_KPP_FR5_SI)....2211
35.4.491 KPP_IPP_IND_ROW_6_SELECT_INPUT (IOMUXC_KPP_FR6_SI)....2211
35.4.492 KPP_IPP_IND_ROW_7_SELECT_INPUT (IOMUXC_KPP_FR7_SI)....2212
35.4.493 RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SELECT_INPUT (IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SI)....2212
35.4.494 RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SELECT_INPUT (IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SI)....2213
35.4.495 SDMA_EVENTS_14_SELECT_INPUT (IOMUXC_SDMA_EVENTS_14_SI)....2213
35.4.496 SDMA_EVENTS_15_SELECT_INPUT (IOMUXC_SDMA_EVENTS_15_SI)....2214
35.4.497 UART1_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U1U_RTS_B_SI)....2214
35.4.498 UART1_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U1U_RXD_MUX_SI)....2215
35.4.499 UART2_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U2U_RTS_B_SI)....2215
35.4.500 UART2_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U2U_RXD_MUX_SI)....2216
35.4.501 UART3_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U3U_RTS_B_SI)....2217
35.4.502 UART3_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U3U_RXD_MUX_SI)....2217
35.4.503 UART4_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U4U_RTS_B_SI)....2218
35.4.504 UART4_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U4U_RXD_MUX_SI)....2218
35.4.505 UART5_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U5U_RTS_B_SI)....2219
35.4.506 UART5_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U5U_RXD_MUX_SI)....2219
35.4.507 USBOH1_IPP_IND_OTG_OC_SELECT_INPUT (IOMUXC_USBOH1_OTG_OC_SI)....2220

35.4.508 WEIMV2_IPP_IND_READ_DATA_0_SELECT_INPUT (IOMUXC_WEIMV2_RD0_SI).....2220
35.4.509 WEIMV2_IPP_IND_READ_DATA_1_SELECT_INPUT (IOMUXC_WEIMV2_RD1_SI).....2221
35.4.510 WEIMV2_IPP_IND_READ_DATA_2_SELECT_INPUT (IOMUXC_WEIMV2_RD2_SI).....2221
35.4.511 WEIMV2_IPP_IND_READ_DATA_3_SELECT_INPUT (IOMUXC_WEIMV2_RD3_SI).....2222
35.4.512 WEIMV2_IPP_IND_READ_DATA_4_SELECT_INPUT (IOMUXC_WEIMV2_RD4_SI).....2222
35.4.513 WEIMV2_IPP_IND_READ_DATA_5_SELECT_INPUT (IOMUXC_WEIMV2_RD5_SI).....2223
35.4.514 WEIMV2_IPP_IND_READ_DATA_6_SELECT_INPUT (IOMUXC_WEIMV2_RD6_SI).....2223
35.4.515 WEIMV2_IPP_IND_READ_DATA_7_SELECT_INPUT (IOMUXC_WEIMV2_RD7_SI).....2224
35.4.516 WEIMV2_IPP_IND_READ_DATA_8_SELECT_INPUT (IOMUXC_WEIMV2_RD8_SI).....2224
35.4.517 WEIMV2_IPP_IND_READ_DATA_9_SELECT_INPUT (IOMUXC_WEIMV2_RD9_SI).....2225
35.4.518 WEIMV2_IPP_IND_READ_DATA_10_SELECT_INPUT (IOMUXC_WEIMV2_RD10_SI).....2225
35.4.519 WEIMV2_IPP_IND_READ_DATA_11_SELECT_INPUT (IOMUXC_WEIMV2_RD11_SI).....2226
35.4.520 WEIMV2_IPP_IND_READ_DATA_12_SELECT_INPUT (IOMUXC_WEIMV2_RD12_SI).....2226
35.4.521 WEIMV2_IPP_IND_READ_DATA_13_SELECT_INPUT (IOMUXC_WEIMV2_RD13_SI).....2227
35.4.522 WEIMV2_IPP_IND_READ_DATA_14_SELECT_INPUT (IOMUXC_WEIMV2_RD14_SI).....2227
35.4.523 WEIMV2_IPP_IND_READ_DATA_15_SELECT_INPUT (IOMUXC_WEIMV2_RD15_SI).....2228

Chapter 36
Keypad Port (KPP)

36.1 Overview 2229
36.1.1 Features.....2230
36.1.2 Modes and Operations.....2231
36.2 External Signals.....2231
36.2.1 External Signals Overview.....2231
36.2.1.1 Input Pins.....2231
36.2.1.2 Output Pins.....2232
36.2.1.3 Generation of Transfer Error Signal on Peripheral Bus.....2232
36.3 Functional Description.....2233
36.3.1 Keypad Matrix Construction.....2233
36.3.2 Keypad Port Configuration.....2233

36.3.3 Keypad Matrix Scanning....2233
36.3.4 Keypad Standby....2234
36.3.5 Glitch Suppression on Keypad Inputs....2234
36.3.6 Multiple Key Closures....2236
36.3.6.1 Ghost Key Problem and Correction....2238
36.3.7 3-Point Contact Keys Support....2240
36.4 Initialization/Application Information....2241
36.4.1 Typical Keypad Configuration and Scanning Sequence....2241
36.4.2 Key Press Interrupt Scanning Sequence....2242
36.4.3 Additional Comments....2242
36.5 Programmable Registers....2243
36.5.1 Keypad Control Register (KPP_KPCR)....2243
36.5.2 Keypad Status Register (KPP_KPSR)....2244
36.5.3 Keypad Data Direction Register (KPP_KDDR)....2246
36.5.4 Keypad Data Register (KPP_KPDR)....2246

Chapter 37
Enhanced LCD Interface (eLCDIF)

37.1 Overview....2249
37.2 Operation....2249
37.2.1 Bus Interface Mechanisms....2250
37.2.1.1 Bus Master Operation in Write/Display Modes....2251
37.2.1.2 System Bus Master Performance....2251
37.2.1.3 DMA Operation in MPU Read Mode....2252
37.2.2 Write Data Path....2253
37.2.3 Read Data Path....2259
37.2.4 eLCDIF Interrupts....2264
37.2.5 Initializing the eLCDIF....2264
37.2.5.1 Write Modes....2264
37.2.5.2 MPU Read Mode....2265

37.2.6 MPU Interface....2266
37.2.6.1 Code Example to Initialize the eLCDIF in MPU Write Mode....2268
37.2.7 VSYNC Interface....2268
37.2.7.1 Code Example to Initialize eLCDIF in VSYNC Mode....2269
37.2.8 DOTCLK Interface....2270
37.2.8.1 Code Example....2272
37.2.9 ITU-R BT.656 Digital Video Interface (DVI)....2272
37.2.10 eLCDIF Pin Usage by Interface Mode....2273
37.3 Behavior During Reset....2277
37.4 Programmable Registers....2277
37.4.1 eLCDIF General Control Register (LCDIF_CTRL*n*)....2280
37.4.2 eLCDIF General Control1 Register (LCDIF_CTRL1*n*)....2283
37.4.3 eLCDIF General Control2 Register (LCDIF_CTRL2*n*)....2286
37.4.4 eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIF_TRANSFER_COUNT)....2288
37.4.5 LCD Interface Current Buffer Address Register (LCDIF_CUR_BUF)....2288
37.4.6 LCD Interface Next Buffer Address Register (LCDIF_NEXT_BUF)....2289
37.4.7 LCD Interface Timing Register (LCDIF_TIMING)....2290
37.4.8 eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0*n*)....2290
37.4.9 eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF_VDCTRL1)....2292
37.4.10 LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF_VDCTRL2)....2292
37.4.11 eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF_VDCTRL3)....2293
37.4.12 eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF_VDCTRL4)....2294
37.4.13 Digital Video Interface Control0 Register (LCDIF_DVICTRL0)....2295
37.4.14 Digital Video Interface Control1 Register (LCDIF_DVICTRL1)....2296
37.4.15 Digital Video Interface Control2 Register (LCDIF_DVICTRL2)....2297
37.4.16 Digital Video Interface Control3 Register (LCDIF_DVICTRL3)....2298
37.4.17 Digital Video Interface Control4 Register (LCDIF_DVICTRL4)....2299
37.4.18 RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIF_CSC_COEFF0)....2300
37.4.19 RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIF_CSC_COEFF1)....2301

37.4.20 RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIF_CSC_COEFF2)....2302
37.4.21 RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIF_CSC_COEFF3)....2303
37.4.22 RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIF_CSC_COEFF4)....2303
37.4.23 RGB to YCbCr 4:2:2 CSC Offset Register (LCDIF_CSC_OFFSET)....2304
37.4.24 RGB to YCbCr 4:2:2 CSC Limit Register (LCDIF_CSC_LIMIT)....2305
37.4.25 LCD Interface Data Register (LCDIF_DATA)....2306
37.4.26 Bus Master Error Status Register (LCDIF_BM_ERROR_STAT)....2306
37.4.27 CRC Status Register (LCDIF_CRC_STAT)....2307
37.4.28 LCD Interface Status Register (LCDIF_STAT)....2307
37.4.29 LCD Interface Version Register (LCDIF_VERSION)....2309
37.4.30 LCD Interface Debug0 Register (LCDIF_DEBUG0)....2309
37.4.31 LCD Interface Debug1 Register (LCDIF_DEBUG1)....2311
37.4.32 LCD Interface Debug2 Register (LCDIF_DEBUG2)....2312
37.4.33 eLCDIF Threshold Register (LCDIF_THRES)....2312

Chapter 38
On-Chip OTP (OCOTP) Controller

38.1 Introduction....2315
38.2 Overview of On-Chip OTP APB Slave (OCOTP)....2315
38.3 Top-Level Symbol and Functional Overview....2316
38.3.1 Operation....2316
38.3.1.1 Fuse and Shadow Register Reads....2317
38.3.1.2 Fuse and Shadow Register Writes....2318
38.3.1.3 Write Postamble....2319
38.3.2 Fuse Shadow Memory Footprint....2319
38.3.3 OTP Read/Write Timing Parameters....2320
38.3.4 Hardware Visible Fuses....2322
38.3.5 Behavior During Reset....2322
38.4 OCOTP Memory Map/Register Definition....2323
38.4.1 OTP Controller Control Register (OCOTP_CTRL*n*)....2325

38.4.2 OTP Controller Timing Register (OCOTP_TIMING).....2327
38.4.3 OTP Controller Write Data Register (OCOTP_DATA).....2328
38.4.4 Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK).....2328
38.4.5 Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0).....2330
38.4.6 Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1).....2331
38.4.7 Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2).....2331
38.4.8 Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3).....2332
38.4.9 Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4).....2332
38.4.10 Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5).....2333
38.4.11 Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6).....2333
38.4.12 Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0).....2334
38.4.13 Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1).....2334
38.4.14 Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2).....2335
38.4.15 Value of OTP Bank1 Word3 (Memory Related Info.) (OCOTP_MEM3).....2335
38.4.16 Value of OTP Bank1 Word4 (Memory Related Info.) (OCOTP_MEM4).....2336
38.4.17 Value of OTP Bank1 Word5 (Memory Related Info.) (OCOTP_MEM5).....2336
38.4.18 Value of OTP Bank1 Word6 (General Purpose Customer Defined Info.) (OCOTP_GP0).....2337
38.4.19 Value of OTP Bank1 Word7 (General Purpose Customer Defined Info.) (OCOTP_GP1).....2337
38.4.20 Shadow Register for OTP Bank2 Word0 (SCC and CRYPTO Key) (OCOTP_SCC0).....2338
38.4.21 Shadow Register for OTP Bank2 Word1 (SCC and CRYPTO Key) (OCOTP_SCC1).....2338
38.4.22 Shadow Register for OTP Bank2 Word2 (SCC and CRYPTO Key) (OCOTP_SCC2).....2339
38.4.23 Shadow Register for OTP Bank2 Word3 (SCC and CRYPTO Key) (OCOTP_SCC3).....2339
38.4.24 Shadow Register for OTP Bank2 Word4 (SCC Key) (OCOTP_SCC4).....2340
38.4.25 Shadow Register for OTP Bank2 Word5 (SCC Key) (OCOTP_SCC5).....2340
38.4.26 Shadow Register for OTP Bank2 Word6 (SCC Key) (OCOTP_SCC6).....2341
38.4.27 Shadow Register for OTP Bank2 Word7 (SCC Key) (OCOTP_SCC7).....2341
38.4.28 Shadow Register for OTP Bank3 Word0 (SRK Hash) (OCOTP_SRK0).....2342
38.4.29 Shadow Register for OTP Bank3 Word1 (SRK Hash) (OCOTP_SRK1).....2342
38.4.30 Shadow Register for OTP Bank3 Word2 (SRK Hash) (OCOTP_SRK2).....2343

38.4.31 Shadow Register for OTP Bank3 Word3 (SRK Hash) (OCOTP_SRK3).....2343
38.4.32 Shadow Register for OTP Bank3 Word4 (SRK Hash) (OCOTP_SRK4).....2344
38.4.33 Shadow Register for OTP Bank3 Word5 (SRK Hash) (OCOTP_SRK5).....2344
38.4.34 Shadow Register for OTP Bank3 Word6 (SRK Hash) (OCOTP_SRK6).....2345
38.4.35 Shadow Register for OTP Bank3 Word7 (SRK Hash) (OCOTP_SRK7).....2345
38.4.36 Value of OTP Bank4 Word0 (Secure JTAG Response Field) (OCOTP_SJC_RESP0).....2346
38.4.37 Value of OTP Bank4 Word1 (Secure JTAG Response Field) (OCOTP_SJC_RESP1).....2346
38.4.38 Value of OTP Bank4 Word2 (MAC Address) (OCOTP_MAC0).....2347
38.4.39 Value of OTP Bank4 Word3 (MAC Address) (OCOTP_MAC1).....2347
38.4.40 Value of OTP Bank4 Word4 (HW Capabilities) (OCOTP_HWCAP0).....2348
38.4.41 Value of OTP Bank4 Word5 (HW Capabilities) (OCOTP_HWCAP1).....2348
38.4.42 Value of OTP Bank4 Word6 (HW Capabilities) (OCOTP_HWCAP2).....2349
38.4.43 Value of OTP Bank4 Word7 (SW Capabilities) (OCOTP_SWCAP).....2349
38.4.44 Software Controllable Signals Register (OCOTP_SCS*n*).....2350
38.4.45 OTP Controller Version Register (OCOTP_VERSION).....2350

Chapter 39
On-Chip RAM (OCRAM)

39.1 Overview.....2353
39.1.1 Memory Map.....2354
39.1.2 Read/Write Arbitration.....2355
39.2 Advanced Features.....2355
39.2.1 Read Data Wait State.....2355
39.2.2 Read Address Pipeline.....2355
39.2.3 Write Data Pipeline.....2356
39.2.4 Write Address Pipeline.....2356
39.3 Programmable Registers.....2356

Chapter 40
1-Wire Block (OWIRE)

40.1 Overview....2357
40.1.1 Features....2357
40.1.2 Modes of Operation....2358
40.2 External Signals....2358
40.3 Functional Description....2358
40.3.1 Normal Operating Modes....2359
40.3.1.1 Reset/Presence-detect Pulse....2359
40.3.1.2 Bit Transfers....2359
40.3.1.2.1 Write-0 Sequence....2359
40.3.1.2.2 Write-1 / Read Sequence....2360
40.3.1.3 Byte Transfers....2360
40.3.1.4 Search ROM Accelerator Mode....2361
40.3.2 Low Power Mode....2361
40.3.3 Clocks....2361
40.3.4 Reset....2362
40.3.4.1 Hardware Reset....2362
40.3.4.2 Software Reset....2362
40.3.5 Interrupts....2362
40.4 Programmable Registers....2363
40.4.1 Control register (OWIRE_CONTROL)....2364
40.4.2 Time Divider register (OWIRE_TIME_DIVIDER)....2365
40.4.3 Reset register (OWIRE_RESET)....2365
40.4.4 Command Register (OWIRE_COMMAND)....2366
40.4.5 Transmit/Receive Register (OWIRE_TX/RX)....2367
40.4.6 Interrupt Register (OWIRE_INTERRUPT)....2367
40.4.7 Interrupt Enable Register (OWIRE_INTERRUPT_EN)....2369

Chapter 41
Performance Monitor (PerfMon)

41.1 Introduction....2371
41.2 Overview....2371
41.3 Operation....2372
41.4 Programmable Registers....2373
41.4.1 PerfMon Control Register (PERFMON_CTRL*n*)....2374
41.4.2 PerfMon Master Enable Register (PERFMON_MASTER_EN)....2376
41.4.3 PerfMon Trap Range Low Address Register (PERFMON_TRAP_ADDR_LOW)....2377
41.4.4 PerfMon Trap Range High Address Register (PERFMON_TRAP_ADDR_HIGH)....2378
41.4.5 PerfMon Latency Threshold Register (PERFMON_LAT_THRESHOLD)....2379
41.4.6 PerfMon AXI Active Cycle Count Register (PERFMON_ACTIVE_CYCLE)....2379
41.4.7 PerfMon Transfer Count Register (PERFMON_TRANSFER_COUNT)....2380
41.4.8 PerfMon Total Latency Count Register (PERFMON_TOTAL_LATENCY)....2380
41.4.9 PerfMon Total Data Count Register (PERFMON_DATA_COUNT)....2381
41.4.10 PerfMon Maximum Latency Register (PERFMON_MAX_LATENCY)....2381
41.4.11 PerfMon Debug Register (PERFMON_DEBUG)....2382
41.4.12 PerfMon Version Register (PERFMON_VERSION)....2383

Chapter 42
Pulse Width Modulation (PWM)

42.1 Overview....2385
42.2 Signal Description....2387
42.2.1 External Signals....2387
42.3 Functional Description....2387
42.3.1 Operation....2387
42.3.1.1 Clocks....2388
42.3.1.2 FIFO....2388
42.3.1.3 Rollover and Compare Event....2389
42.3.1.4 Low Power Mode Behavior....2389

42.3.1.5 Debug Mode Behavior....2390
42.4 Programmable Registers....2390
42.4.1 PWM Control Register (PWM*x*_PWMCR)....2391
42.4.2 PWM Status Register (PWM*x*_PWMSR)....2393
42.4.3 PWM Interrupt Register (PWM*x*_PWMIR)....2394
42.4.4 PWM Sample Register (PWM*x*_PWMSAR)....2395
42.4.5 PWM Period Register (PWM*x*_PWMPR)....2396
42.4.6 PWM Counter Register (PWM*x*_PWMCNR)....2396

Chapter 43
Enhanced Pixel Pipeline (ePXP)

43.1 Overview....2399
43.1.1 Image Support....2400
43.1.2 Block Size Selection....2401
43.1.3 ePXP Limitations/Issues....2401
43.2 Operation....2402
43.2.1 Pixel Handling....2403
43.2.2 S0 Cropping/Masking....2404
43.2.3 Scaling....2407
43.2.3.1 Bilinear Image Scaling Filter....2408
43.2.3.2 YUV 4:2:2 Image Scaling....2410
43.2.3.3 YUV 4:2:0 Image Scaling....2411
43.2.4 Color Space Conversion (CSC)....2411
43.2.4.1 CSC1 Operation....2412
43.2.4.2 YUV versus YCbCr Support....2413
43.2.4.3 CSC2 Operation....2413
43.2.5 Overlays....2414
43.2.6 Alpha Blending....2415
43.2.7 Color Key....2416
43.2.8 Raster Operations (ROPs)....2417

43.2.9 Lookup Table (LUT) Memory....2418
43.2.10 Histogram....2419
43.2.11 Rotation....2419
43.2.12 In-place Rendering....2423
43.2.13 Interlaced Video Support....2423
43.2.14 eLCDIF Interlock....2423
43.2.15 Queueing Frame Operations....2425
43.3 Examples....2427
43.3.1 Basic QVGA Example....2427
43.3.2 Basic QVGA with Overlays....2428
43.3.3 Cropped QVGA Example....2431
43.3.4 Upscale QVGA to VGA with Overlays....2433
43.3.5 Downscale VGA to WQVGA (480x272) to fill screen....2435
43.3.6 Downscale VGA to QVGA with Overlapping Overlays....2437
43.4 Programmable Registers....2440
43.4.1 ePXP Control Register 0 (ePXP_CTRL*n*)....2444
43.4.2 ePXP Status Register (ePXP_STAT*n*)....2448
43.4.3 Output Frame Buffer Pointer (ePXP_OUTBUF)....2449
43.4.4 Output Frame Buffer Pointer #2 (ePXP_OUTBUF2)....2449
43.4.5 ePXP Output Buffer Size (ePXP_OUTSIZE)....2450
43.4.6 ePXP Source 0 (video) Input Buffer Pointer (ePXP_S0BUF)....2451
43.4.7 Source 0 U/Cb or 2 Plane UV Input Buffer Pointer (ePXP_S0UBUF)....2451
43.4.8 Source 0 V/Cr Input Buffer Pointer (ePXP_S0VBUF)....2452
43.4.9 ePXP Source 0 (video) Buffer Parameters (ePXP_S0PARAM)....2452
43.4.10 Source 0 Background Color (ePXP_S0BACKGROUND)....2453
43.4.11 Source 0 Cropping Register (ePXP_S0CROP)....2454
43.4.12 Source 0 Scale Factor Register (ePXP_S0SCALE)....2455
43.4.13 Source 0 Scale Offset Register (ePXP_S0OFFSET)....2456
43.4.14 Color Space Conversion Coefficient Register 0 (ePXP_CSCCOEF0)....2457

43.4.15 Color Space Conversion Coefficient Register 1 (ePXP_CSCCOEF1).....2458
43.4.16 Color Space Conversion Coefficient Register 2 (ePXP_CSCCOEF2).....2459
43.4.17 ePXP Next Frame Pointer (ePXP_NEXT*n*).....2460
43.4.18 ePXP S0 Color Key Low (ePXP_S0COLORKEYLOW).....2462
43.4.19 ePXP S0 Color Key High (ePXP_S0COLORKEYHIGH).....2462
43.4.20 ePXP Overlay Color Key Low (ePXP_OLCOLORKEYLOW).....2463
43.4.21 ePXP Overlay Color Key High (ePXP_OLCOLORKEYHIGH).....2464
43.4.22 ePXP Debug Register (ePXP_DEBUG).....2464
43.4.23 ePXP Overlay n Buffer Pointer (ePXP_OL*n*).....2465
43.4.24 ePXP Overlay n Size (ePXP_OL__SIZE*n*).....2465
43.4.25 ePXP Overlay n Parameters (ePXP_OL__PARAM*n*).....2466
43.4.26 Color Space Conversion Control Register. (ePXP_CSC2CTRL).....2468
43.4.27 Color Space Conversion Coefficient Register 0 (ePXP_CSC2COEF0).....2469
43.4.28 Color Space Conversion Coefficient Register 1 (ePXP_CSC2COEF1).....2470
43.4.29 Color Space Conversion Coefficient Register 2 (ePXP_CSC2COEF2).....2470
43.4.30 Color Space Conversion Coefficient Register 3 (ePXP_CSC2COEF3).....2471
43.4.31 Color Space Conversion Coefficient Register 4 (ePXP_CSC2COEF4).....2471
43.4.32 Color Space Conversion Coefficient Register 5 (ePXP_CSC2COEF5).....2472
43.4.33 Lookup Table Control Register. (ePXP_LUT_CTRL).....2473
43.4.34 Lookup Table Data Register. (ePXP_LUT).....2473
43.4.35 Histogram Control Register. (ePXP_HIST_CTRL).....2474
43.4.36 2-level Histogram Parameter Register. (ePXP_HIST2_PARAM).....2475
43.4.37 4-level Histogram Parameter Register. (ePXP_HIST4_PARAM).....2476
43.4.38 8-level Histogram Parameter 0 Register. (ePXP_HIST8_PARAM0).....2477
43.4.39 8-level Histogram Parameter 1 Register. (ePXP_HIST8_PARAM1).....2478
43.4.40 16-level Histogram Parameter 0 Register. (ePXP_HIST16_PARAM0).....2479
43.4.41 16-level Histogram Parameter Register. (ePXP_HIST16_PARAM1).....2480
43.4.42 16-level Histogram Parameter Register. (ePXP_HIST16_PARAM2).....2481
43.4.43 16-level Histogram Parameter Register. (ePXP_HIST16_PARAM3).....2482

Chapter 44
Quality of Service Controller (QoSC)

44.1 Introduction....2483

44.2 Overview of Quality of Service Controller APB Slave (QoSC)....2483

44.3 Top-Level Symbol and Functional Overview....2483

44.4 APB Slave....2486

44.5 Master Side Signal Control....2487

44.6 Slave Side Signal Control....2487

44.6.1 Passthrough Priority Mode....2488

44.6.2 Manual Priority Mode....2488

44.7 Boost Functionality....2489

44.8 AXI Fabric Master Port Disable....2489

44.9 Programmable Registers....2490

44.9.1 QOS Control Register (QOS_CTRL*n*)....2491

44.9.2 AXI QOS Register (QOS_AXI_QOS0*n*)....2493

44.9.3 AXI QOS Register (QOS_AXI_QOS1*n*)....2495

44.9.4 AXI QOS Register (QOS_AXI_QOS2*n*)....2496

44.9.5 EMI priority Registers (QOS_EMI_PRIORITY0*n*)....2498

44.9.6 EMI priority Registers (QOS_EMI_PRIORITY1*n*)....2500

44.9.7 EMI priority Registers (QOS_EMI_PRIORITY2*n*)....2502

44.9.8 AXI Master Disble Register (QOS_DISABLE*n*)....2504

44.9.9 QOS Version Register (QOS_VERSION)....2506

Chapter 45
Read Only Memory Controller Patch (ROMCP)

45.1 Introduction....2507

45.1.1 Overview....2507

45.1.2 Features....2508

45.1.3 Modes of Operation....2508

45.1.3.1 Normal Operating Modes....2509

45.1.3.2 Low Power Modes....2509
45.2 Functional Description....2509
45.2.1 ROM Controller (ROMC) Functional Description....2509
45.2.1.1 Functionality Overview....2509
45.2.1.2 ROMC Architecture Diagram....2510
45.2.1.3 AHB-Lite Interface....2510
45.2.2 ROMPATCH Functional Description....2511
45.2.2.1 ROMPATCH Disabling....2511
45.2.2.2 ROMPATCH Event Priority....2511
45.2.2.3 Data Fixing....2511
45.2.2.4 Opcode Patching....2512
45.2.2.4.1 Typical Software Response to Opcode Patch....2513
45.2.2.5 Alternate Masters and ROMPATCH....2514
45.3 Programmable Registers....2514
45.3.1 ROMPATCH Data Registers (ROMCP_ROMPATCHD*n*)....2517
45.3.2 ROMPATCH Control Register (ROMCP_ROMPATCHCNTL)....2518
45.3.3 ROMPATCH Enable Registers Low (ROMCP_ROMPATCHENL)....2519
45.3.4 ROMPATCH Address Registers (ROMCP_ROMPATCHA*n*)....2520
45.3.5 ROMPATCH Status Register (ROMCP_ROMPATCHSR)....2521

Chapter 46
Smart Direct Memory Access Controller (SDMA)

46.1 Introduction....2523
46.1.1 Overview....2523
46.1.2 Features....2525
46.2 Functional Description....2527
46.3 SDMA Core....2529
46.3.1 SDMA Core Structure....2529
46.3.2 Program Control Unit (PCU)....2532
46.3.2.1 Instruction Types....2532

46.3.2.2 PCU States....2533
46.3.3 SDMA Core Memory....2536
46.4 Scheduler....2536
46.4.1 Primary Functions....2536
46.4.2 Channels and DMA Requests....2537
46.4.2.1 Channels....2537
46.4.2.2 DMA Requests....2537
46.4.2.3 Mapping from DMA Requests to Channels and Priorities....2537
46.4.3 Scheduler Functional Description....2537
46.4.3.1 Scheduler Overview....2537
46.4.3.2 DMA Requests Scanning....2539
46.4.3.3 Mapping DMA Requests to Pending Channels....2539
46.4.3.4 Channel Overflow....2543
46.4.3.5 Runnable Channels Evaluation....2543
46.4.3.6 Next Channel Decision Tree....2545
46.4.3.7 Scheduler State Diagram....2547
46.4.3.8 Scheduler Pipeline Timing Diagram....2549
46.4.3.9 Channel-DMA Request Mapping....2549
46.4.3.10 Examples: How to Start a Channel....2549
46.4.4 Context Switching....2550
46.4.4.1 Context Switch Modes....2551
46.4.4.2 Context Switch Procedure....2552
46.4.4.3 Context Map in Memory....2553
46.5 Functional Units....2553
46.5.1 CRC Calculation Unit....2553
46.5.1.1 CRC Structure....2554
46.5.1.2 CRC Data Processing....2554
46.5.1.3 CRC Registers....2555
46.5.1.4 CRC Summary....2555

46.5.2 Burst DMA Unit....2556
46.5.2.1 Burst DMA Structure....2557
46.5.2.2 Burst DMA Registers....2558
46.5.2.3 Burst DMA Data Transfers....2559
46.5.2.3.1 Data Retrieval from the ARM platform Memory....2559
46.5.2.3.2 Storing Data Into the ARM platform Memory....2560
46.5.2.3.3 Transferring Data Between Two ARM platform Memory Locations-Burst DMA Unit....2560
46.5.3 Peripheral DMA Unit....2560
46.5.3.1 Peripheral DMA Structure....2561
46.5.3.2 Peripheral DMA Registers....2562
46.5.3.3 Peripheral DMA Data Transfers....2563
46.5.3.3.1 Data Retrieval from the ARM platform Memory or Peripheral....2563
46.5.3.3.2 Storing Data into the ARM platform Memory or Peripheral....2563
46.5.3.3.3 Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit....2564
46.6 SDMA Security Support....2564
46.6.1 Locked Mode....2564
46.7 OnCE and PCU Debug States....2565
46.8 SDMA Clocks and Low Power Modes....2567
46.8.1 Clock Gating and Low Power Modes....2568
46.8.1.1 Coarse Clock Gating....2568
46.8.1.2 Refined Clock Gating....2569
46.8.1.3 Low Power Modes and User Control....2569
46.8.1.3.1 SLEEP Mode....2570
46.8.1.3.2 RUN Mode....2570
46.8.1.3.3 DEBUG Mode....2571
46.8.1.4 Stop Mode Response....2571
46.8.2 Reset....2571

46.9 Software Interface....2571
46.10 Initialization Information....2572
46.10.1 Hardware Reset....2572
46.10.2 Channel Script Execution....2573
46.10.3 Initialization and Script Execution Setup Sequence....2573
46.11 SDMA Programming Model....2574
46.11.1 State and Registers Per Channel....2575
46.11.2 General Purpose Registers....2575
46.11.3 Functional Unit State....2575
46.11.3.1 Program Counter Register (PC)....2575
46.11.3.2 Flags....2575
46.11.3.3 Return Program Counter (RPC)....2576
46.11.3.4 Loop Mode Start Program Counter (SPC)....2577
46.11.3.5 Loop Mode End Program Counter (EPC)....2577
46.11.4 Context Switching-Programming....2577
46.11.5 Address Space....2578
46.11.5.1 Instruction Memory Map....2579
46.11.5.2 Data Memory Map....2579
46.12 SDMA Initialization....2581
46.12.1 Hardware Reset-SDMA....2581
46.12.2 Standard Boot Sequence....2582
46.12.3 User-Defined Boot Sequence....2582
46.12.4 Script Loading and Context Initialization....2583
46.13 Instruction Description....2583
46.13.1 Scheduling Instructions....2583
46.13.2 Conditional Branch Instructions....2584
46.13.3 Unconditional Jump Instructions....2584
46.13.4 Subroutine Return Instructions....2585
46.13.5 Loop Instruction....2585

46.13.6 Miscellaneous Instructions....2585
46.13.7 Logic Instructions....2585
46.13.8 Arithmetic Instructions....2586
46.13.9 Compare Instructions....2586
46.13.10 Test Instructions....2587
46.13.11 Byte Permutation Instructions....2587
46.13.12 Bit Shift Instructions....2587
46.13.13 Bit Manipulation Instructions....2587
46.13.14 SDMA Memory Access Instructions....2588
46.13.15 Functional Unit Instructions....2588
46.13.16 Illegal Instructions....2588
46.13.17 Debug Instructions....2589
46.14 Functional Units Programming Model....2589
46.14.1 Burst DMA Unit Programming....2590
46.14.1.1 Memory Source Address Register (MSA)....2591
46.14.1.2 Memory Destination Address Register (MDA)....2591
46.14.1.3 Memory Data Buffer Register (MD)....2592
46.14.1.4 State Register (MS)....2592
46.14.1.5 Burst DMA Write (stf)....2594
46.14.1.6 Burst DMA Read (ldf)....2597
46.14.1.7 Prefetch/Flush and Auto-Flush Management-Burst DMA Unit....2598
46.14.1.8 Data Alignment and Endianness-Burst DMA Unit....2600
46.14.1.8.1 Burst DMA in Read Mode....2600
46.14.1.8.2 Burst DMA in Write Mode....2601
46.14.1.8.3 Endianness-Burst DMA Unit....2603
46.14.1.9 Burst DMA Unit Copy Mode....2603
46.14.1.10 Burst DMA Unit Error Management....2604
46.14.1.11 Conditional Yielding-Burst DMA Unit....2606

46.14.2 Peripheral DMA Unit Programming....2607
46.14.2.1 Peripheral Source Address Register (PSA)....2608
46.14.2.2 Peripheral Destination Address Register (PDA)....2609
46.14.2.3 Peripheral Data Register (PD)....2609
46.14.2.4 Peripheral State Register (PS)....2610
46.14.2.5 Peripheral DMA Write (stf)-Write Mode....2611
46.14.2.6 Peripheral DMA Read (ldf)-Read Mode....2614
46.14.2.7 Peripheral DMA Unit Copy Mode....2615
46.14.2.8 Error Management....2615
46.14.2.8.1 Immediate Errors....2616
46.14.2.8.2 Data Transfer Errors....2616
46.14.2.8.3 Read Error (First Phase)....2617
46.14.2.8.4 Write Error and Read Error (Second Phase)....2617
46.14.2.8.5 Copy Mode Errors....2618
46.14.2.8.6 Error Check Example....2618
46.14.2.9 Peripheral DMA Unit Prefetch/Flush Management....2619
46.14.3 CRC Unit....2619
46.14.3.1 Polynomial Register (CA)....2619
46.14.3.2 Accumulator Register (CS)....2620
46.14.3.3 Write Instruction (stf)....2621
46.14.3.4 Read Instruction (ldf)....2621
46.14.3.5 Operating Mode....2622
46.14.4 OnCE and Real-Time Debug....2623
46.14.4.1 Memory and Register Access....2623
46.14.4.2 Hardware Breakpoints....2623
46.14.4.3 Watchpoints....2623
46.14.4.4 Software Breakpoints....2624
46.14.4.5 Core Control....2624

46.15 The OnCE Controller....2624
46.15.1 OnCE Commands....2624
46.15.2 Sending Commands to the OnCE Controller....2625
46.15.2.1 Using the JTAG Interface....2625
46.15.2.2 Using the ARM platform....2626
46.15.2.3 Conflicts Between the JTAG and the ARM platform Accesses....2627
46.15.3 Executing a Command from the OnCE....2628
46.15.3.1 Nature of the Commands....2628
46.15.3.2 Execution Request....2628
46.15.3.3 Command Execution....2629
46.15.4 Registers Descriptions....2631
46.15.4.1 Event Cell Counter Register (ECOUNT)....2631
46.15.4.2 Event Cell Address Registers (EAA or EAB)....2631
46.15.4.3 Event Cell Address Mask Register (EAM)....2632
46.15.4.4 Event Cell Data Register (ED)....2632
46.15.4.5 Event Cell Data Mask Register (EDM)....2632
46.15.4.6 Real Time Buffer Register (RTB)....2632
46.15.4.7 Event Control Register (ECTL)....2633
46.15.4.8 Trace Buffer (TB)....2633
46.15.4.9 OnCE Status Register (OSTAT)....2633
46.15.5 JTAG Interface Requirements....2634
46.15.5.1 TCK Speed Limitation....2634
46.15.5.2 Synchronization Implementation....2634
46.15.5.3 JTAG Controller Start-Up Recommended Procedure....2636
46.16 Using the OnCE....2636
46.16.1 Activating Clocks in Debug Mode....2636
46.16.2 Getting the Current Status....2637
46.16.3 Methods of Entering Debug Mode....2637
46.16.3.1 External Debug Request During Reset....2637

46.16.3.2 Debug Request During Normal Activity....2638
46.16.3.3 Software Breakpoint Instruction....2638
46.16.3.4 Event Detection Unit Matching Condition....2638
46.16.4 Executing Instructions in Debug Mode....2638
46.16.5 Command Sequences Examples....2639
46.16.5.1 Getting the SDMA Status....2639
46.16.5.2 Saving the Context....2640
46.16.5.3 Restoring the Context....2641
46.16.5.4 Accessing the Memory....2642
46.16.5.5 Resuming Program Execution....2642
46.16.5.6 Single Stepping in RAM....2643
46.16.5.7 Single Stepping in ROM....2643
46.16.6 OnCE Event Detection Unit....2644
46.16.7 Clock Gating and Reset....2645
46.16.7.1 Clocks....2645
46.16.7.2 Resets....2646
46.16.8 Real Time Features....2646
46.16.8.1 Trace Buffer....2646
46.16.8.2 Real Time Buffer....2648
46.16.8.3 Emulation Pin....2648
46.16.8.4 Real-Time Debug Outputs....2648
46.17 Instruction Set....2652
46.17.1 Instruction Encoding....2652
46.17.2 SDMA Instruction Set....2654
46.17.2.1 ADD (Addition)....2657
46.17.2.2 ADDI (Add with Immediate Value)....2658
46.17.2.3 AND (Logical AND)....2659
46.17.2.4 ANDI (Logical AND with Immediate Value)....2660
46.17.2.5 ANDN (Logical AND NOT)....2661

46.17.2.6 ANDNI (Logical AND with Negated Immediate Value).....2662
46.17.2.7 ASR1 (Arithmetic Shift Right by 1 Bit).....2663
46.17.2.8 BCLRI1 (Bit Clear Immediate).....2663
46.17.2.9 BDF (Conditional Branch if Destination Fault).....2664
46.17.2.10 BF (Conditional Branch if False).....2665
46.17.2.11 BSETI (Bit Set Immediate).....2667
46.17.2.12 BSF (Conditional Branch if Source Fault).....2668
46.17.2.13 BT (Conditional Branch if True).....2669
46.17.2.14 BTSTI (Bit Test immediate).....2670
46.17.2.15 CLRF (Clear ARM platform flags).....2671
46.17.2.16 CMPEQ (Compare for Equal).....2671
46.17.2.17 CMPEQI (Compare with Immediate for Equal).....2672
46.17.2.18 CMPHS (Compare for Higher or Same).....2673
46.17.2.19 CMPLT (Compare for Less Than).....2674
46.17.2.20 cpShReg (Update Context of PCU Registers and Flag).....2675
46.17.2.21 DONE (DONE, Yield) 2675
46.17.2.22 ILLEGAL (ILLEGAL Instruction).....2677
46.17.2.23 JMP (Unconditional Jump Immediate).....2678
46.17.2.24 JMPR (Unconditional Jump).....2679
46.17.2.25 JSR (Unconditional Jump to Subroutine Immediate).....2679
46.17.2.26 JSRR (Unconditional Jump to Subroutine).....2680
46.17.2.27 LD (Load Register).....2681
46.17.2.28 LDF (Load Register from Functional Unit).....2682
46.17.2.29 LDI (Load Register with Immediate Value).....2684
46.17.2.30 LDRPC (Load from RPC to Register).....2685
46.17.2.31 LOOP (Hardware Loop).....2686
46.17.2.32 LSL1 (Logical Shift Left by 1 Bit).....2689
46.17.2.33 LSR1 (Logical Shift Right by 1 Bit).....2689
46.17.2.34 MOV (Logical Move).....2690

46.17.2.35 NOTIFY (Notify to ARM platform)....2691
46.17.2.36 OR (Logical OR)....2692
46.17.2.37 ORI (Logical OR with Immediate Value)....2693
46.17.2.38 RET (Return from Subroutine)....2694
46.17.2.39 REVB (Reverse Byte Order)....2695
46.17.2.40 Reverse Low Order Bytes(REVBLO)....2696
46.17.2.41 ROR1 (Rotate Right by 1 Bit)....2696
46.17.2.42 RORB (Rotate Right by 1 Byte)....2697
46.17.2.43 SOFTBKPT (Software Breakpoint)....2698
46.17.2.44 ST (Store Register)....2698
46.17.2.45 STF (Store Register in Functional Unit)....2700
46.17.2.46 SUB (Subtract)....2703
46.17.2.47 SUBI (Subtract with Immediate)....2704
46.17.2.48 TST (Test with Zero)....2705
46.17.2.49 TSTI (Test Immediate)....2706
46.17.2.50 XOR (Logical Exclusive OR)....2707
46.17.2.51 XORI (Exclusive OR with Immediate)....2708
46.17.2.52 YIELD, YIELDGE (DONE, Yield)....2709
46.18 Software Restrictions....2709
46.18.1 Unsupported Burst DMA Access Sequence....2709
46.19 Application Notes....2710
46.19.1 Data Structures for Boot Code and Channel Scripts....2710
46.19.1.1 Buffer Descriptor Format....2711
46.19.1.2 Buffer Descriptor Commands for Bootload scripts....2714
46.19.1.3 Example of Buffer Descriptors for Channel 0....2716
46.19.1.4 Channel Context....2719
46.19.2 Typical Data Transfer Supported by SDMA DMA Units....2721
46.19.2.1 External Memory to External Memory....2722

46.19.2.2 Peripheral to Peripheral Transfer....2723
46.19.2.2.1 Source and Destination Target Have the Same Data Path Width....2723
46.19.2.2.2 Source and Destination Target Have a Different Data Path Width....2724
46.19.2.3 Transfer Between Peripheral and External Memory....2725
46.19.2.3.1 Peripheral to External Memory Transfer....2725
46.19.2.3.2 External Memory to Peripheral Transfer....2727
46.19.2.4 Transfer Between External Memory and Internal Memory....2728
46.19.2.4.1 Internal Memory to Internal Memory....2728
46.19.2.4.2 Transfer Between Peripheral and Internal Memory....2728
46.20 ARM Platform Memory Map and Control Register Definitions....2729
46.20.1 ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)....2734
46.20.2 Channel Interrupts (SDMAARM_INTR)....2735
46.20.3 Channel Stop/Channel Status (SDMAARM_STOP_STAT)....2735
46.20.4 Channel Start (SDMAARM_HSTART)....2735
46.20.5 Channel Event Override (SDMAARM_EVTOVR)....2736
46.20.6 Channel BP Override (SDMAARM_DSPOVR)....2736
46.20.7 Channel ARM platform Override (SDMAARM_HOSTOVR)....2737
46.20.8 Channel Event Pending (SDMAARM_EVTPEND)....2737
46.20.9 Reset Register (SDMAARM_RESET)....2738
46.20.10 DMA Request Error Register (SDMAARM_EVTERR)....2738
46.20.11 Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)....2739
46.20.12 Schedule Status (SDMAARM_PSW)....2739
46.20.13 DMA Request Error Register (SDMAARM_EVTERRDBG)....2740
46.20.14 Configuration Register (SDMAARM_CONFIG)....2741
46.20.15 SDMA LOCK (SDMAARM_SDMA_LOCK)....2742
46.20.16 OnCE Enable (SDMAARM_ONCE_ENB)....2743
46.20.17 OnCE Data Register (SDMAARM_ONCE_DATA)....2743
46.20.18 OnCE Instruction Register (SDMAARM_ONCE_INSTR)....2744
46.20.19 OnCE Status Register (SDMAARM_ONCE_STAT)....2744

46.20.20 OnCE Command Register (SDMAARM_ONCE_CMD).....2746
46.20.21 Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR).....2746
46.20.22 Channel 0 Boot Address (SDMAARM_CHN0ADDR).....2747
46.20.23 DMA Requests (SDMAARM_EVT_MIRROR).....2748
46.20.24 DMA Requests 2 (SDMAARM_EVT_MIRROR2).....2748
46.20.25 Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1).....2749
46.20.26 Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2).....2750
46.20.27 Channel Priority Registers (SDMAARM_SDMA_CHNPRI*n*).....2751
46.20.28 Channel Enable RAM (SDMAARM_SDMA.CHNENBL*n*).....2752
46.21 BP Memory Map and Control Register Definitions.....2752
46.21.1 Channel 0 Pointer (SDMABP_DC0PTR).....2753
46.21.2 Channel Interrupts (SDMABP_INTR).....2753
46.21.3 Channel Stop/Channel Status (SDMABP_STOP_STAT).....2754
46.21.4 Channel Start (SDMABP_DSTART).....2754
46.21.5 DMA Request Error Register (SDMABP_EVTERR).....2755
46.21.6 Channel DSP Interrupt Mask (SDMABP_INTRMASK).....2755
46.21.7 DMA Request Error Register (SDMABP_EVTERRDBG).....2756
46.22 SDMA Internal (Core) Memory Map and Internal Register Definitions.....2756
46.22.1 ARM platform Channel 0 Pointer (SDMACORE_MC0PTR).....2757
46.22.2 Current Channel Pointer (SDMACORE_CCPTR).....2758
46.22.3 Current Channel Register (SDMACORE_CCR).....2758
46.22.4 Highest Pending Channel Register (SDMACORE_NCR).....2759
46.22.5 External DMA Requests Mirror (SDMACORE_EVENTS).....2760
46.22.6 Current Channel Priority (SDMACORE_CCPRI).....2761
46.22.7 Next Channel Priority (SDMACORE_NCPRI).....2761
46.22.8 OnCE Event Cell Counter (SDMACORE_ECOUNT).....2762
46.22.9 OnCE Event Cell Control Register (SDMACORE_ECTL).....2762
46.22.10 OnCE Event Address Register A (SDMACORE_EAA).....2764
46.22.11 OnCE Event Cell Address Register B (SDMACORE_EAB).....2764

46.22.12 OnCE Event Cell Address Mask (SDMACORE_EAM)....2765
46.22.13 OnCE Event Cell Data Register (SDMACORE_ED)....2765
46.22.14 OnCE Event Cell Data Mask (SDMACORE_EDM)....2765
46.22.15 OnCE Real-Time Buffer (SDMACORE_RTB)....2766
46.22.16 OnCE Trace Buffer (SDMACORE_TB)....2766
46.22.17 OnCE Status (SDMACORE_OSTAT)....2767
46.22.18 Channel 0 Boot Address (SDMACORE_MCHN0ADDR)....2769
46.22.19 ENDIAN Status Register (SDMACORE_ENDIANNESS)....2770
46.22.20 Lock Status Register (SDMACORE_SDMA_LOCK)....2771
46.22.21 External DMA Requests Mirror #2 (SDMACORE_EVENTS2)....2771
46.23 SDMA Peripheral Registers....2772

Chapter 47
System JTAG Controller (SJC)

47.1 Introduction....2773
47.1.1 Overview....2774
47.2 Modes of Operation....2775
47.3 TAP Selection Block (TSB)....2777
47.3.1 Select Mode Using Software....2777
47.4 External Signal Description....2778
47.4.1 External Signal Overview....2778
47.4.2 TAP Controller....2780
47.4.3 Accessing ExtraDebug Registers....2782
47.5 Boundary Scan Register (BSR)....2785
47.6 SoC JTAG Instruction Register (SJIR)....2785
47.6.1 ID_CODE Instruction (IDCODE)....2786
47.6.2 SAMPLE/PRELOAD Instruction....2787
47.6.3 EXTEST Instruction....2788
47.6.4 HIGHZ Instruction....2788
47.6.5 BYPASS Instruction....2789

47.6.6 ENABLE_ExtraDebug Instruction....2789
47.6.7 ENTER_DEBUG instruction....2789
47.6.8 TAP Select Instruction....2790
47.7 Security....2791
47.7.1 JTAG Security Modes....2791
47.7.1.1 Mode 1: No Debug - Maximum Security....2791
47.7.1.2 Mode 2: Secure JTAG - High Security....2792
47.7.1.2.1 Challenge/Response Mechanism in System JTAG Mode....2792
47.7.1.3 Mode 3: JTAG Enabled - Low Security....2793
47.7.2 Software Enabled JTAG....2793
47.7.3 Kill Trace....2794
47.7.4 SJC Disable Fuse....2794
47.8 Functional Description....2795
47.8.1 Static Core Debug....2795
47.8.2 Reset Mechanism....2795
47.9 Initialization/Application Information....2796
47.10 Programmable Registers....2797
47.10.1 General Purpose Unsecured Status Register 1 (SJC_GPUSR1)....2798
47.10.2 General Purpose Unsecured Status Register 2 (SJC_GPUSR2)....2799
47.10.3 General Purpose Unsecured Status Register 3 (SJC_GPUSR3)....2800
47.10.4 General Purpose Secured Status Register (SJC_GPSSR)....2800
47.10.5 Debug Control Register (SJC_DCR)....2801
47.10.6 Security Status Register (SJC_SSR)....2802
47.10.7 General Purpose Clocks Control Register (SJC_GPCCR)....2804
47.10.8 General Purpose Unsecured Control Register n (SJC_GPUCR)....2805
47.10.9 General Purpose Secured Control Register (SJC_GPSCR)....2805

Chapter 48
Shared Peripheral Bus Arbiter (SPBA)

48.1 Introduction....2807

48.2 General Overview....2809
48.3 Features....2809
48.4 Modes of Operation....2809
48.5 Functional Description....2810
48.5.1 Masters Arbitration....2810
48.6 Resource Ownership Control....2813
48.6.1 Access Control....2813
48.6.1.1 Peripheral Access....2813
48.6.1.2 Peripheral Right Register Access....2814
48.6.2 Owner Election....2815
48.6.3 Ending Ownership....2815
48.6.3.1 Software Controlled Ownership Ending....2815
48.6.4 The Un-owned State....2816
48.7 Memory Map/Register Definition....2816
48.7.1 SPBA Register Definition....2817
48.7.1.1 Peripheral Right Register (SPBA_PRRn)....2820

Chapter 49
System Reset Controller (SRC)

49.1 Introduction....2823
49.1.1 Overview....2823
49.1.2 Features....2824
49.2 Functional Description....2824
49.2.1 Reset Control....2825
49.2.1.1 Reset Inputs and Outputs....2825
49.2.1.2 Reset Handling....2828
49.2.1.2.1 Reset Qualification....2828
49.2.1.2.2 Reset Sequence and De-Assertion....2828
49.2.1.2.2.1 IPP_RESET_B(POR)....2829
49.2.1.2.2.2 COLD RESET....2829

49.2.1.2.2.3 WARM RESET....2830
49.2.1.2.2.4 WARM BOOT....2831
49.2.1.2.3 SMS Sub Module....2835
49.2.1.2.4 SRTC_RST_B generation....2838
49.2.2 SJC_POR_RST_B Generation....2838
49.2.3 Parallel Reset Requests....2838
49.2.4 DFT Mux on Reset Outputs....2839
49.2.5 Boot Mode Control....2840
49.2.5.1 BOOT_MODE Pin Latching....2841
49.2.5.2 Correspondence between SRC SBMR Bit/Signal Names and Fuse Names....2842
49.2.5.3 Output Boot Signals....2844
49.3 Programmable Registers....2845
49.3.1 SRC Control Register (SRC_SCR)....2846
49.3.2 SRC Boot Mode Register (SRC_SBMR)....2847
49.3.3 SRC Reset Status Register (SRC_SRSR)....2848
49.3.4 SRC Interrupt Status Register (SRC_SISR)....2850
49.3.5 SRC Interrupt Mask Register (SRC_SIMR)....2851

Chapter 50
State Retention Power Gating Controller (SRPGC)

50.1 Introduction....2853
50.1.1 Features....2853
50.1.2 Modes of Operation....2853
50.1.2.1 Functional Mode....2853
50.1.2.2 Debug Mode....2854
50.2 Power Gating Cells....2854
50.2.1 Power Supply Switch Cells....2854
50.2.2 Power Supply Shorting Cells....2855
50.2.3 State Retention Power Gating (SRPG) Cell....2857

50.3 SRPGC Interface....2860
50.3.1 Power-Down Sequence (SRPG Entry Sequence)....2860
50.3.1.1 Power-Down Sequence Timing Waveforms....2861
50.3.2 Power-Up Sequence (SRPG Exit Sequence)....2862
50.3.2.1 Power-up Sequence Timing Waveforms....2862
50.4 Programmable Registers....2863
50.4.1 SPRGC Control Register (SRPGC_SRPGCR)....2864
50.4.2 Power-up Sequence Control Register (SRPGC_PUPSCR)....2864
50.4.3 Power-down Sequence Control Register (SRPGC_PDNSCR)....2865
50.4.4 SPRGC Status Register (SRPGC_SRPGSR)....2866
50.4.5 SPRGC Debug Register (SRPGC_SRPGDR)....2867

Chapter 51
Secure Real Time Clock (SRTC)

51.1 Overview....2869
51.1.1 Low Power SRTC (SRTC LP) Overview....2869
51.1.2 High Power SRTC (SRTC HP) Overview....2870
51.1.3 Features....2872
51.1.4 Modes of Operations....2873
51.2 External Signal Description....2873
51.3 Functional Description....2874
51.3.1 Power and Clock Source....2874
51.3.1.1 Clocks....2875
51.3.2 High Power SRTC (SRTC HP) Description....2875
51.3.2.1 SRTC HP nonsecured Counter....2875
51.3.2.1.1 SRTC HP Counter Calibration....2876
51.3.2.2 SRTC HP nonsecured Counter Alarm....2876
51.3.2.3 SRTC HP Periodic Alarm....2877
51.3.3 Low Power SRTC (SRTC LP) Description....2878
51.3.3.1 SRTC LP Behavior during System Power Down and POR....2878

51.3.3.2 SRTC LP Secured Counter....2879
51.3.3.2.1 SRTC LP Counter Calibration....2879
51.3.3.3 SRTC LP Secured Counter Alarm....2880
51.3.3.4 Monotonic Counter....2880
51.3.3.4.1 Monotonic Counter Roll-Over Protection Mechanism....2881
51.3.3.5 General Purpose Always-Powered Registers....2884
51.3.3.6 SRTC LP Monitor....2884
51.3.3.6.1 SRTC State Machine....2884
51.3.3.6.2 Power Supply Glitch Detector (PGD)....2885
51.3.3.6.3 Clock Tampering Detector (CTD)....2886
51.3.3.6.4 Voltage Level Tampering Detector....2887
51.3.3.6.5 Power Fail Detector (PFD)....2887
51.4 SRTC Reset and System Power-Up....2888
51.5 SRTC Interrupts and Alarms....2888
51.6 Initialization Information/Application Information....2889
51.6.1 Flow Chart of SRTC LP Operation....2889
51.6.2 Flow Chart of SRTC HP Operation....2890
51.7 Software Restrictions....2891
51.8 Programmable Registers....2892
51.8.1 LP Secure Counter MSB Register (SRTC_LPSCMR)....2894
51.8.2 LP Secure Counter LSB Register (SRTC_LPSCLR)....2894
51.8.3 LP Secure Alarm Register (SRTC_LPSAR)....2895
51.8.4 LP Secure Monotonic Counter Register (SRTC_LPSMCR)....2895
51.8.5 LP Control Register (SRTC_LPCR)....2896
51.8.6 LP Status Register (SRTC_LPSR)....2899
51.8.7 LP Power Supply Glitch Detector Register (SRTC_LPPDR)....2902
51.8.8 LP General Purpose Register (SRTC_LPGR)....2902
51.8.9 HP Counter MSB Register (SRTC_HPCMR)....2903
51.8.10 HP Counter LSB Register (SRTC_HPCLR)....2904

51.8.11 HP Alarm MSB Register (SRTC_HPAMR)....2904
51.8.12 HP Alarm LSB Register (SRTC_HPALR)....2905
51.8.13 HP Control Register (SRTC_HPCR)....2905
51.8.14 HP Interrupt Status Register (SRTC_HPISR)....2907
51.8.15 HP Interrupt Enable Register (SRTC_HPIENR)....2910

Chapter 52
Synchronous Serial Interface (SSI)

52.1 Overview....2913
52.1.1 Features....2914
52.1.2 Modes of Operation....2915
52.2 External Signal Description....2915
52.2.1 Signals Overview....2915
52.3 SSI Transmit FIFO 0 & 1 Registers....2919
52.4 SSI Transmit Shift Register (TXSR)....2919
52.5 SSI Receive FIFO 0 and 1 Registers....2922
52.6 SSI Receive Shift Register (RXSR)....2922
52.7 Functional Description....2924
52.7.1 Operating Modes....2924
52.7.1.1 Normal Mode....2926
52.7.1.1.1 Normal Mode Transmit....2926
52.7.1.1.2 Normal Mode Receive....2927
52.7.1.2 Network Mode....2929
52.7.1.2.1 Network Mode Transmit....2930
52.7.1.2.2 Network Mode Receive....2931
52.7.1.3 Gated Clock Mode....2933
52.7.1.4 I2S Mode....2936
52.7.1.5 AC97 Mode....2938
52.7.1.5.1 AC97 Fixed Mode (SSI.SACNT[1]=0)....2940
52.7.1.5.2 AC97 Variable Mode (SSI.SACNT[1]=1)....2940

52.7.2 External Frame and Clock Operation....2941
52.7.2.1 Data Alignment Formats Supported....2941
52.7.3 SSI Architecture....2942
52.7.4 SSI Clocking....2943
52.7.4.1 SSI Clock and Frame Sync Generation....2944
52.7.4.2 DIV2, PSR and PM Bit Description....2945
52.7.5 Receive Interrupt Enable Bit Description....2947
52.7.6 Transmit Interrupt Enable Bit Description....2948
52.7.7 Internal Frame and Clock Shutdown....2949
52.7.8 Peripheral Bus Interface....2951
52.7.8.1 Transfer Lengths Supported....2951
52.7.8.2 Transfer Bus Errors....2951
52.7.8.3 Clock Rate....2952
52.7.9 Reset....2952
52.8 Programmable Registers....2952
52.8.1 SSI Transmit Data Register n (SSI*x*_SSI_STX*n*)....2955
52.8.2 SSI Receive Data Register n (SSI*x*_SSI_SRX*n*)....2955
52.8.3 SSI Control Register (SSI*x*_SSI_SCR)....2956
52.8.4 SSI Interrupt Status Register (SSI*x*_SSI_SISR)....2958
52.8.5 SSI Interrupt Enable Register (SSI*x*_SIER)....2963
52.8.6 SSI Transmit Configuration Register (SSI*x*_SSI_STCR)....2965
52.8.7 SSI Receive Configuration Register (SSI*x*_SSI_SRCR)....2967
52.8.8 SSI Transmit Clock Control Register (SSI*x*_SSI_STCCR)....2969
52.8.9 SSI Receive Clock Control Register (SSI*x*_SRCCR)....2971
52.8.10 SSI FIFO Control/Status Register (SSI*x*_SSI_SFCSR)....2972
52.8.11 SSI AC97 Control Register (SSI*x*_SSI_SACNT)....2976
52.8.12 SSI AC97 Command Address Register (SSI*x*_SSI_SACADD)....2977
52.8.13 SSI AC97 Command Data Register (SSI*x*_SSI_SACDAT)....2978
52.8.14 SSI AC97 Tag Register (SSI*x*_SATAG)....2978

52.8.15 SSI Transmit Time Slot Mask Register (SSI*x*_SSI_STMSK)....2979
52.8.16 SSI Receive Time Slot Mask Register (SSI*x*_SSI_SRMSK)....2979
52.8.17 SSI AC97 Channel Status Register (SSI*x*_SSI_SACCST)....2980
52.8.18 SSI AC97 Channel Enable Register (SSI*x*_SSI_SACCEN)....2980
52.8.19 SSI AC97 Channel Disable Register (SSI*x*_SSI_SACCDIS)....2981

Chapter 53
Temperature Sensor (TempSensor)

53.1 Introduction....2983
53.2 Overview....2983
53.3 Features....2984
53.4 Digital-Count-to-Temperature Conversion Algorithm....2984
53.5 TempSensor Memory Map/Register Definition....2984
53.5.1 TempSensor Control Register (TEMPSENSOR_ANADIG_CTRL*n*)....2985

Chapter 54
Interrupt Controller (TZIC)

54.1 Overview....2987
54.2 Features....2988
54.3 External Signal Description....2989
54.4 Functional Description....2989
54.4.1 Security Configurability....2989
54.4.1.1 Interrupt Priority....2989
54.4.2 AXI Interface....2989
54.4.3 Interrupt Engine....2990
54.4.4 Auto-Vectored Interrupt Handling....2991
54.4.5 Reset....2992
54.5 Initialization Information....2992
54.6 Programmable Registers....2993
54.6.1 Control Register (TZIC_INTCTRL)....2997
54.6.2 Interrupt Controller Type Register (TZIC_INTTYPE)....2998

54.6.3 Priority Mask Register (TZIC_PRIOMASK)....2999
54.6.4 Synchronizer Control (TZIC_SYNCCTRL)....3000
54.6.5 DSM Interrupt Holdoff (TZIC_DSMINT)....3001
54.6.6 Interrupt Security (FIQ) Register: Irq 0 to 31 (TZIC_INTSEC*n*)....3002
54.6.7 Enable Set Register: Irq 0 to 31 (TZIC_ENSET*n*)....3003
54.6.8 Enable Clear Register: Irq 0 to 31 (TZIC_ENCLEAR*n*)....3004
54.6.9 Source Set Register: Irq 0 to 31 (TZIC_SRCSET*n*)....3005
54.6.10 Source Clear Register: Irq 0 to 31 (TZIC_SRCCLEAR*n*)....3006
54.6.11 Priority Register: Irq 0 to 3 (TZIC_PRIORITY*n*)....3006
54.6.12 Pending Register: Irq 0 to 31 (TZIC_PND*n*)....3008
54.6.13 High Priority Pending Register: Irq 0 to 31 (TZIC_HIPND*n*)....3008
54.6.14 Wakeup Config Register: Irq 0 to 31 (TZIC_WAKEUP*n*)....3009
54.6.15 Software Interrupt Trigger Register (TZIC_SWINT)....3010

Chapter 55
Universal Asynchronous Receiver/Transmitter (UART)

55.1 Overview....3013
55.1.1 Features....3014
55.1.2 Modes of Operation....3015
55.2 External Signals....3015
55.2.1 Detailed Signal Descriptions....3016
55.2.1.1 Serial/IrDA Signals....3016
55.2.1.1.1 RXD - Data Receive....3016
55.2.1.1.2 TXD - Data Transmit....3016
55.2.1.2 Modem Control Signals....3017
55.2.1.2.1 CTS - Clear To Send....3017
55.2.1.2.2 RTS - Request To Send....3017
55.2.1.2.3 DSR - Data Set Ready....3017
55.2.1.2.4 DCD - Data Carrier Detected....3017
55.2.1.2.5 DTR - Data Terminal Ready....3017

55.2.1.2.6 RI - Ring Indicator....3017
55.2.1.3 Interrupt Signals....3017
55.2.1.3.1 interrupt_uart - UART Interrupt....3018
55.2.1.4 DMA Request Signals....3018
55.2.1.4.1 dma_req_rx - Receiver DMA Request....3018
55.2.1.4.2 dma_req_tx - Transmitter DMA Request....3018
55.2.1.5 Clock Signals....3018
55.2.1.5.1 peripheral_clock - Peripheral Clock....3018
55.2.1.5.2 module_clock - Module Clock....3018
55.2.1.6 Special Signals....3018
55.2.1.6.1 stop_req - Stop Mode....3018
55.2.1.6.2 doze_req - Doze Mode....3018
55.2.1.6.3 debug_req - Debug Mode....3019
55.3 Functional Description....3019
55.3.1 Interrupts and DMA Requests....3019
55.3.2 Clocks....3020
55.3.2.1 Clock requirements....3020
55.3.2.2 Maximum Baud Rate....3021
55.3.2.3 Clocking in Low-Power Modes....3021
55.3.3 General UART Definitions....3022
55.3.3.1 RTS - UART Request To Send....3023
55.3.3.2 RTS Edge Triggered Interrupt....3023
55.3.3.3 DTR - Data Terminal Ready....3024
55.3.3.4 DSR - Data Set Ready....3024
55.3.3.5 DTR/DSR Edge Triggered Interrupt....3025
55.3.3.6 DCD - Data Carrier Detect....3025
55.3.3.7 RI - Ring Indicator....3026
55.3.3.8 CTS - Clear To Send....3026
55.3.3.9 Programmable CTS Deassertion....3026

55.3.3.10 TXD - UART Transmit....3026
55.3.3.11 RXD - UART Receive....3027
55.3.4 Transmitter....3028
55.3.4.1 Transmitter FIFO Empty Interrupt Suppression....3029
55.3.4.2 Transmitting a Break Condition....3031
55.3.5 Receiver....3031
55.3.5.1 Idle Line Detect....3032
55.3.5.2 Aging Character Detect....3033
55.3.5.3 Receiver Wake....3034
55.3.5.4 Receiving a BREAK Condition....3035
55.3.5.5 Vote Logic....3035
55.3.5.6 Baud Rate Automatic Detection Logic....3037
55.3.5.6.1 Baud Rate Automatic Detection Protocol....3038
55.3.5.6.2 New Baud Rate Determination....3038
55.3.5.6.2.1 New Autobaud Counter Stopped bit and Interrupt....3039
55.3.6 Escape Sequence Detection....3039
55.4 Binary Rate Multiplier (BRM)....3041
55.5 Infrared Interface....3043
55.5.1 Generalities-Infrared....3043
55.5.2 Inverted Transmission and Reception bits (INVT & INVR)....3044
55.5.3 InfraRed Special Case (IRSC) Bit....3044
55.5.4 IrDA interrupt....3045
55.5.5 Conclusion about IrDA....3046
55.5.6 Programming IrDA Interface....3047
55.5.6.1 High Speed....3047
55.5.6.2 Low Speed....3047
55.6 Low Power Modes....3048
55.6.1 UART Operation in System Doze Mode....3049
55.6.2 UART Operation in System Stop Mode....3049

55.6.3 Power Saving Method in UART....3049
55.7 UART Operation in System Debug State....3050
55.8 Reset....3050
55.8.1 Hardware reset....3050
55.8.2 Software reset....3051
55.9 Transfer Error....3051
55.10 Functional Timing....3051
55.10.1 IrDA Mode....3051
55.11 Initialization....3052
55.11.1 Programming the UART in RS-232 mode....3052
55.12 References....3054
55.13 Programmable Registers....3054
55.13.1 UART Receiver Register (UART*x*_URXD)....3059
55.13.2 UART Transmitter Register (UART*x*_UTXD)....3061
55.13.3 UART Control Register 1 (UART*x*_UCR1)....3062
55.13.4 UART Control Register 2 (UART*x*_UCR2)....3064
55.13.5 UART Control Register 3 (UART*x*_UCR3)....3067
55.13.6 UART Control Register 4 (UART*x*_UCR4)....3069
55.13.7 UART FIFO Control Register (UART*x*_UFCR)....3071
55.13.8 UART Status Register 1 (UART*x*_USR1)....3073
55.13.9 UART Status Register 2 (UART*x*_USR2)....3075
55.13.10 UART Escape Character Register (UART*x*_UESC)....3078
55.13.11 UART Escape Timer Register (UART*x*_UTIM)....3078
55.13.12 UART BRM Incremental Register (UART*x*_UBIR)....3079
55.13.13 UART BRM Modulator Register (UART*x*_UBMR)....3079
55.13.14 UART Baud Rate Count Register (UART*x*_UBRC)....3080
55.13.15 UART One Millisecond Register (UART*x*_ONEMS)....3081
55.13.16 UART Test Register (UART*x*_UTS)....3082

Chapter 56
Universal Serial Bus OTG HOST1 (USBOH1)

56.1 Introduction....3085
56.1.1 Overview....3086
56.1.2 Features....3086
56.1.3 Modes of Operation....3086
56.1.3.1 Operational Modes....3086
56.1.3.1.1 Normal Mode....3086
56.1.3.1.2 Low Power Mode....3087
56.2 External Signal Description....3087
56.2.1 External Signal Overview....3087
56.2.2 Detailed Signal Descriptions....3091
56.3 Functional Description....3092
56.3.1 USB HOST Controller 1....3092
56.3.2 USB OTG Controller....3092
56.3.2.1 Host Mode....3092
56.3.2.2 Peripheral (Device) Mode....3092
56.3.3 USB Power Control Module....3092
56.3.3.1 Entering Suspend Mode....3093
56.3.3.2 Wake-up Events....3093
56.3.3.2.1 Host Mode Events....3093
56.3.3.2.2 Device Mode Events....3094
56.3.4 USB Interrupts....3094
56.3.4.1 USB Core Interrupts....3094
56.3.4.2 USB Wake-up Interrupts....3094
56.4 Initialization/Application Information....3095
56.4.1 Software Model....3095
56.4.1.1 Device Data Structure....3096
56.4.1.2 Host Data Structure....3097

56.4.2 Register Interface....3098
56.4.2.1 Configuration, Control and Status register set....3099
56.4.2.2 Summary of register layouts....3102
56.4.2.3 Identification Registers....3105
56.4.2.4 Device/Host Capability Registers....3105
56.4.2.5 Device/Host Timer Registers (Non-EHCI)....3105
56.4.2.6 Device/Host Operational Registers....3105
56.4.2.7 OTG Operations....3105
56.4.2.7.1 Register Bits....3105
56.4.2.7.2 Hardware Assist....3106
56.4.2.7.2.1 Auto-Reset....3107
56.4.2.7.2.2 Data-Pulse....3107
56.4.2.7.2.3 B-Disconnect to A-Connect....3107
56.4.3 Host Data Structures....3108
56.4.3.1 Periodic Frame List....3109
56.4.3.2 Asynchronous List Queue Head Pointer....3110
56.4.3.3 Isochronous (High-Speed) Transfer Descriptor (iTD)....3111
56.4.3.3.1 iTD Next Link Pointer....3113
56.4.3.3.2 iTD Transaction Status and Control List....3114
56.4.3.3.3 iTD Buffer Page Pointer List (Plus)....3115
56.4.3.4 Split Transaction Isochronous Transfer Descriptor (siTD)....3116
56.4.3.4.1 siTD Next Link Pointer....3118
56.4.3.4.2 siTD Endpoint Capabilities/Characteristics....3118
56.4.3.4.3 siTD Transfer State....3119
56.4.3.4.4 siTD Buffer Pointer List (plus)....3120
56.4.3.4.5 siTD Back Link Pointer....3121
56.4.3.5 Queue Element Transfer Descriptor (qTD)....3121
56.4.3.5.1 Next qTD Pointer....3123
56.4.3.5.2 Alternate Next qTD Pointer....3123

56.4.3.5.3 qTD Token....3123
56.4.3.5.4 qTD Buffer Page Pointer List....3126
56.4.3.6 Queue Head....3127
56.4.3.6.1 Queue Head Endpoint Capabilities/Characteristics....3129
56.4.3.6.2 Queue Head Transfer Overlay....3131
56.4.3.7 Periodic Frame Span Traversal Node (FSTN) 3132
56.4.3.7.1 FSTN Normal Path Pointer 3133
56.4.3.7.2 FSTN Back Path Link Pointer 3133
56.4.4 Host Operational Model 3134
56.4.4.1 Host Controller Initialization 3134
56.4.4.2 Port Routing and Control 3136
56.4.4.2.1 Port Routing Control via EHCI Configured (CF) Bit 3137
56.4.4.2.2 Port Routing Control via PortOwner and Disconnect Event 3138
56.4.4.2.3 Example Port Routing State Machine 3140
56.4.4.2.3.1 EHCI HC Owner 3140
56.4.4.2.3.2 Companion HC Owner 3141
56.4.4.2.4 Port Power 3141
56.4.4.2.5 Port Reporting Over-Current 3142
56.4.4.3 Host Operational Model Suspend/Resume 3143
56.4.4.3.1 Port Suspend/Resume 3144
56.4.4.4 Schedule Traversal Rules 3146
56.4.4.4.1 Example - Preserving Micro-Frame Integrity 3148
56.4.4.4.1.1 Transaction Fit - A Best-Fit Approximation Algorithm 3149
56.4.4.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries 3151
56.4.4.6 Periodic Schedule 3154
56.4.4.7 Managing Isochronous Transfers Using iTDs 3155
56.4.4.7.1 Host Controller Operational Model for iTDs 3156
56.4.4.7.2 Software Operational Model for iTDs 3158
56.4.4.7.2.1 Periodic Scheduling Threshold....3160

56.4.4.8 Asynchronous Schedule 3161
56.4.4.8.1 Adding Queue Heads to Asynchronous Schedule....3162
56.4.4.8.2 Removing Queue Heads from Asynchronous Schedule 3163
56.4.4.8.3 Empty Asynchronous Schedule Detection 3165
56.4.4.8.4 Restarting Asynchronous Schedule Before EOF 3166
56.4.4.8.4.1 Example Method for Restarting Asynchronous Schedule Traversal 3167
56.4.4.8.4.2 Async Sched Not Active 3168
56.4.4.8.4.3 Async Sched Active 3168
56.4.4.8.4.4 Async Sched Sleeping 3169
56.4.4.8.4.5 Example Derivation for AsyncSchedSleepTime....3169
56.4.4.8.5 Asynchronous Schedule Traversal : Start Event....3169
56.4.4.8.6 Reclamation Status Bit (USBSTS Register) 3170
56.4.4.9 Operational Model for Nak Counter....3170
56.4.4.9.1 Nak Count Reload Control 3172
56.4.4.9.1.1 Wait for List Head 3173
56.4.4.9.1.2 Do Reload 3173
56.4.4.9.1.3 Wait for Start Event 3173
56.4.4.10 Managing Control/Bulk/Interrupt Transfers via Queue Heads....3174
56.4.4.10.1 Fetch Queue Head 3176
56.4.4.10.2 Advance Queue 3176
56.4.4.10.3 Execute Transaction 3177
56.4.4.10.3.1 Interrupt Transfer Pre-condition Criteria 3178
56.4.4.10.3.2 Asynchronous Transfer Pre-operations and Pre-condition Criteria 3178
56.4.4.10.3.3 Transfer Type Independent Pre-operations....3178
56.4.4.10.3.4 Halting a Queue Head 3181
56.4.4.10.3.5 Asynchronous Schedule Park Mode 3182
56.4.4.10.4 Write Back qTD 3184

56.4.4.10.5 Follow Queue Head Horizontal Pointer 3184
56.4.4.10.6 Buffer Pointer List Use for Data Streaming with qTDs 3184
56.4.4.10.7 Adding Interrupt Queue Heads to the Periodic Schedule 3187
56.4.4.10.8 Managing Transfer Complete Interrupts from Queue Heads 3187
56.4.4.11 Ping Control 3188
56.4.4.12 Split Transactions 3189
56.4.4.12.1 Split Transactions for Asynchronous Transfers 3189
56.4.4.12.1.1 Asynchronous-Do Start Split.....3190
56.4.4.12.1.2 Asynchronous-Do Complete Split 3191
56.4.4.12.2 Split Transaction Interrupt 3192
56.4.4.12.2.1 Split Transaction Scheduling Mechanisms for Interrupt 3192
56.4.4.12.2.2 Host Controller Operational Model for FSTNs 3195
56.4.4.12.2.3 Software Operational Model for FSTNs.....3198
56.4.4.12.2.4 Tracking Split Transaction Progress for Interrupt Transfers ...3198
56.4.4.12.2.5 Split Transaction Execution State Machine for Interrupt 3199
56.4.4.12.2.6 Periodic Interrupt-Do Start Split 3201
56.4.4.12.2.7 Periodic Interrupt- Do Complete Split 3202
56.4.4.12.2.8 Managing QH.FrameTag Field 3205
56.4.4.12.2.9 Rebalancing the Periodic Schedule 3206
56.4.4.12.3 Split Transaction Isochronous 3207
56.4.4.12.3.1 Split Transaction Scheduling Mechanisms for Isochronous …3207
56.4.4.12.3.2 Tracking Split Transaction Progress for Isochronous Transfers 3211
56.4.4.12.3.3 Split Transaction Execution State Machine for Isochronous . .3213
56.4.4.12.3.4 Periodic Isochronous-Do Start Split 3214
56.4.4.12.3.5 Periodic Isochronous-Do Complete Split 3217
56.4.4.12.3.6 Complete-Split for Scheduling Boundary Cases 2a, 2b 3220
56.4.4.12.3.7 Split Transaction for Isochronous - Processing Examples 3221
56.4.4.13 Host Controller Pause 3223

56.4.4.14 Port Test Modes 3224
56.4.4.15 EHCI Interrupts 3225
56.4.4.15.1 Transfer/Transaction Based Interrupts 3226
56.4.4.15.1.1 Transaction Error 3226
56.4.4.15.1.2 Serial Bus Babble....3227
56.4.4.15.1.3 Data Buffer Error 3228
56.4.4.15.1.4 USB Interrupt (Interrupt on Completion (IOC)) 3228
56.4.4.15.1.5 Short Packet....3229
56.4.4.15.2 Host Controller Event Interrupts 3229
56.4.4.15.2.1 Port Change Events 3229
56.4.4.15.2.2 Frame List Rollover 3229
56.4.4.15.2.3 Interrupt on Async Advance 3229
56.4.4.15.2.4 Host System Error 3230
56.4.5 EHCI Deviation....3231
56.4.5.1 Embedded Transaction Translator Function....3231
56.4.5.1.1 Capability Registers....3232
56.4.5.1.2 Operational Registers....3232
56.4.5.1.3 Discovery of FS or LS....3232
56.4.5.1.4 Data Structures....3233
56.4.5.1.5 Operational Model....3233
56.4.5.1.5.1 Micro- frame Pipeline....3234
56.4.5.1.5.2 Split State Machines....3234
56.4.5.1.5.3 Asynchronous Transaction Scheduling and Buffer Management....3235
56.4.5.1.5.4 Periodic Transaction Scheduling and Buffer Management....3235
56.4.5.1.5.5 Multiple Transaction Translators....3236
56.4.5.2 Device Operation....3236
56.4.5.3 USBMODE Register....3236
56.4.5.3.1 Non-Zero Fields the Register File....3237

56.4.5.3.2 SOF Interrupt....3237
56.4.5.4 Embedded Design Interface....3237
56.4.5.4.1 Frame Adjust Register....3237
56.4.5.5 Miscellaneous Variations from EHCI....3237
56.4.5.5.1 Programmable Physical Interface Behaviour....3237
56.4.5.5.2 Discovery....3238
56.4.5.5.2.1 Port Reset....3238
56.4.5.5.2.2 Port Speed Detection....3238
56.4.5.5.3 Port Test Mode....3238
56.4.6 Device Data Structures....3239
56.4.6.1 Endpoint Queue Head (dQH)....3240
56.4.6.1.1 dQH Endpoint Capabilities/Characteristics....3241
56.4.6.1.2 dQH Transfer Overlay....3242
56.4.6.1.3 dQH Current dTD Pointer....3242
56.4.6.1.4 dQH Set-up Buffer....3243
56.4.6.2 Endpoint Transfer Descriptor (dTD)....3243
56.4.7 Device Operational Model....3245
56.4.7.1 Device Controller Initialization....3246
56.4.7.2 Port State and Control....3247
56.4.7.2.1 Bus Reset....3249
56.4.7.2.2 Port State and Control Suspend/Resume....3250
56.4.7.2.3 Managing Endpoints....3251
56.4.7.2.4 Endpoint Initialization....3252
56.4.7.2.5 Data Toggle....3253
56.4.7.3 Operational Model For Packet Transfers....3254
56.4.7.3.1 Interrupt/Bulk Endpoint Operational Model....3255
56.4.7.3.2 Control Endpoint Operation Model....3257
56.4.7.3.3 Isochronous Endpoint Operational Model....3260

56.4.7.4 Managing Queue Heads....3263
56.4.7.4.1 Queue Head Initialization....3264
56.4.7.4.2 Operational Model For Setup Transfers....3264
56.4.7.5 Managing Transfers with Transfer Descriptors....3265
56.4.7.5.1 Software Link Pointers....3265
56.4.7.5.2 Building a Transfer Descriptor....3266
56.4.7.5.3 Executing A Transfer Descriptor....3266
56.4.7.5.4 Transfer Completion....3267
56.4.7.5.5 Flushing/De-priming an Endpoint....3267
56.4.7.5.6 Device Error Matrix....3268
56.4.7.6 Servicing Interrupts....3269
56.4.7.6.1 High-Frequency Interrupts....3269
56.4.7.6.2 Low-Frequency Interrupts....3269
56.4.7.6.3 Error Interrupts....3270
56.5 Programmable Registers....3270
56.5.1 USB Control Register (USBOH1_USB_CTRL)....3271
56.5.2 UTMI PHY Output Clock Valid Register (USBOH1_UTMI_CLK_VLD)....3273
56.5.3 OTG UTMI PHY Control 0 Register (USBOH1_OTG_PHY_CTRL_0)....3274
56.5.4 OTG UTMI PHY Control 1 Register (USBOH1_OTG_PHY_CTRL_1)....3276
56.5.5 USB Control Register 1 (USBOH1_USB_CTRL_1)....3279
56.5.6 USB Control Register 2 (USBOH1_USB_CTRL_2)....3280
56.5.7 Host1 UTMI PHY Control 0 Register (USBOH1_UH1_PHY_CTRL_0)....3281
56.5.8 Host1 UTMI PHY Control 1 Register (USBOH1_UH1_PHY_CTRL_1)....3283
56.5.9 USB Clock on/off control Register (USBOH1_USB_CLKONOFF_CTRL)....3285
56.6 Core Registers....3286
56.6.1 Identification register (USB_ID)....3289
56.6.2 Hardware General (USB_HWGENERAL)....3289
56.6.3 Host Hardware Parameters (USB_HWHOST)....3290
56.6.4 Device Hardware Parameters (USB_HWDEVICE)....3290

56.6.5 TX Buffer Hardware Parameters (USB_HWTXBUF).....3291
56.6.6 RX Buffer Hardware Parameters (USB_HWRXBUF).....3292
56.6.7 General Purpose Timer #0 Load (USB_GPTIMER0LD - (non-EHCI)).....3292
56.6.8 General Purpose Timer #0 Controller (USB_GPTIMER0CTRL - (non-EHCI)).....3293
56.6.9 General Purpose Timer #1 Load (USB_GPTIMER1LD - (non-EHCI)).....3294
56.6.10 General Purpose Timer #1 Controller (USB_GPTIMER1CTRL - (non-EHCI)).....3295
56.6.11 System Bus Interface Control Register (USB_SBUSCFG).....3296
56.6.12 Capability Register Length (USB_CAPLENGTH - EHCI Compliant).....3297
56.6.13 Host Controller Interface Version (USB_HCIVERSION - EHCI Compliant).....3297
56.6.14 Host Controller Structural Parameters (USB_HCSPARAMS - EHCI Compliant).....3298
56.6.15 Host Controller Capability Parameters (USB_HCCPARAMS - EHCI Compliant).....3300
56.6.16 Device Controller Interface Version (USB_DCIVERSION - (non-EHCI)).....3301
56.6.17 Device Controller Capability Parameters (USB_DCCPARAMS - (non-EHCI)).....3302
56.6.18 USB Command Register (USB_USBCMD).....3303
56.6.19 USB Status Register (USB_USBSTS).....3307
56.6.20 Interrupt Enable Register (USB_USBINTR).....3310
56.6.21 USB Frame Index (USB_FRINDEX).....3312
56.6.22 Frame List Base Address (USB_PERIODICLISTBASE).....3313
56.6.23 Device Address (USB_DEVICEADDR).....3314
56.6.24 Next Asynch. Address (USB_ASYNCLISTADDR).....3314
56.6.25 Endpoint List Address (USB_ENDPTLISTADDR).....3315
56.6.26 Programmable Burst Size (USB_BURSTSIZE).....3316
56.6.27 TX FIFO Fill Tuning (USB_TXFILLTUNING).....3316
56.6.28 IC_USB enable and voltage negotiation (USB_IC_USB).....3318
56.6.29 ULPI Viewport (USB_ULPIVIEW).....3319
56.6.30 Port Status and Control (USB_PORTSC).....3322
56.6.31 On-The-Go Status & control (USB_OTGSC).....3329
56.6.32 USB Device Mode (USB_USBMODE).....3332
56.6.33 Endpoint Setup Status (USB_ENDPTSETUPSTAT).....3333

56.6.34 Endpoint Initialization (USB_ENDPTPRIME)....3334
56.6.35 Endpoint De-Initialize (USB_ENDPTFLUSH)....3334
56.6.36 Endpoint Status (USB_ENDPTSTAT)....3335
56.6.37 Endpoint Complete (USB_ENDPTCOMPLETE)....3336
56.6.38 Endpoint Control0 (USB_ENDPTCTRL0)....3337
56.6.39 Endpoint Controln (USB_ENDPTCTRL*n*)....3338

Chapter 57
Watchdog Timer (WDOG-1)

57.1 Introduction....3343
57.2 Overview....3343
57.2.1 Features....3344
57.2.2 Modes and Operations....3345
57.3 External Signals....3345
57.4 Functional Description....3345
57.4.1 Time-Out Event....3345
57.4.1.1 Servicing WDOG-1 To Reload The Counter....3346
57.4.2 Interrupt Event....3346
57.4.3 Power-Down Counter Event....3347
57.4.4 Low Power Modes....3347
57.4.4.1 STOP and DOZE Mode....3347
57.4.4.2 WAIT Mode....3347
57.4.5 Debug Mode....3348
57.4.6 Operations....3348
57.4.6.1 Watchdog Reset Generation....3348
57.4.6.2 WDOG_B Generation....3349
57.4.7 Clocks....3351
57.4.8 Reset....3351
57.4.9 Interrupt....3352
57.4.10 Flow Diagrams....3352

57.5 Initialization....3356
57.6 Programmable Registers....3357
57.6.1 Watchdog Control Register (WDOG_WCR)....3357
57.6.2 Watchdog Service Register (WDOG_WSR)....3359
57.6.3 Watchdog Reset Status Register (WDOG_WRSR)....3360
57.6.4 Watchdog Interrupt Control Register (WDOG_WICR)....3361
57.6.5 Watchdog Miscellaneous Control Register (WDOG_WMCR)....3362

Chapter 58
Crystal Oscillator (XTALOSC)

58.1 Introduction....3363
58.1.1 Interface Specification....3363
58.1.2 Crystal Operating Frequency....3365
58.1.3 Power Supply....3365
58.1.4 Input Clock in Bypass Mode....3365
58.1.5 Input Control Signal....3365
58.1.6 Output Clock....3365
58.2 Operation Modes....3365
58.2.1 Crystal Osc Mode....3365
58.2.2 Bypass Mode....3366
58.2.3 Low Power Mode....3366
58.2.4 Oscillator Safety Margin Requirements....3366

Chapter 1
Introduction

1.1 About This Document

This reference manual describes the functionality of the i.MX50 multimedia applications processor, Freescale Semiconductor's latest addition to a growing family of multimedia-focused products offering high performance processing optimized for lowest power consumption. The i.MX50 is a system on a chip (SoC) which integrates on one circuit a complete microcomputer platform, and features Freescale's advanced implementation of the ARM®Cortex™-A8 core.

1.1.1 Audience

This manual is intended for use by board-level product designers and product software developers. Readers should have a background in computer engineering and/or software engineering, and understand concepts of digital system design, microprocessor architecture, input/output (I/O) devices, and industry standard communication and device interface protocols.

1.1.2 Organization

The manual is parced in to two main sections titled "Book I" and "Book II".

Book I examines the chip at a system level and provides an architectural overview. Topics you'll find include the system memory map, system-level interrupt events, external pins and pin multiplexing, external memory, system debug, system boot, multimedia subsystem, power management, and system security.

Book II describes the ARM Platform, ARM Platform debug, and an array of internal functional blocks.

1.1.3 Conventions

This document uses the following notational conventions:

Table 1-1. Notational Conventions

Convention	Description
cleared / set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
italics	Italics indicate variable command parameters, for example, **bcctr***x* Book titles in text are set in italics.
15	An integer in decimal
0x	Prefix to denote hexadecimal number
0b	Prefix to denote binary number. Binary 0 and 1 are written without the prefix.
n'H4000CA00	n-bit hexadecimal number
BLK_REG_NAME	Register names are all uppercase. The block mnemonic is prepended with an underscore delimiter (_).
BLK_REG[FIELD]	Fields within registers appear in brackets. For example, ESR[RLS] refers to the Receive Last Slot field of the ESAI Status Register.
BLK_REG[*n*]	Bit number *n* within register BLK.REG. Bit numbering is little endian.
BLK_REG[*l:r*]	Register bit ranges. Ranges are indicated by the left-most bit number *l* and the right-most bit number *r* separated by a colon (:). For example, ESR[15:0] refers to the lower half word in the ESAI Status Register.
x, U	In some contexts, such as signal encodings, an unitalicized x indicates a don't care or uninitialized. The binary value could be 1 or 0.
x	An italicized *x* indicates an alphanumeric variable.
n, m	Italicized *n or m* represent integer variables.
!	Binary logic operator NOT
&&	Binary logic operator AND
||	Binary logic operator OR
^ or <O+>	Binary logic operator XOR
|	Bit-wise OR. For example, 0b0001 | 0b1000 yields the value 0b1001.
&	Bit-wise AND. For example, 0b0001 & 0b1000 yields the value 0b0000.

Table continues on the next page...

Table 1-1. Notational Conventions (continued)

Convention	Description
{A,B}	Concatenation, where the *n*-bit value A is prepended to the *m*-bit value B to form an (*n*+*m*)-bit value. For example, {0, REG*m* [14:0]} yeilds a 16-bit value with 0 in the most significant bit.
- or grey fill	Indicates a reserved bit field in an register. Although these bits can be written to as ones or zeros, they are always read as zeros.
>>	Shift right logical one position
<<	Shift left logical one position
= <left arrow>	Assignment
==	Compare equal
!=	Compare not equal
>	Greater than
<	Less than

The following signal conventions are also used:

Table 1-2. Signal Conventions

Convention	Description
_b, _B	When appended to a signal name, this indicates that a signal is active-low.
$\overline{\text{NEG\_ACTIVE}}$	Overbar also denotes a negative active signal.
UPPERCASE	Package pin names, Block I/O signals
lowercase	Lowercase is used to indicate internal signals.

1.1.4 Register Access

1.1.4.1 Register Diagram Field Access Type Legend

The following figure provides an interpretation of the notation which is used in register diagrams for a number of common field access types.

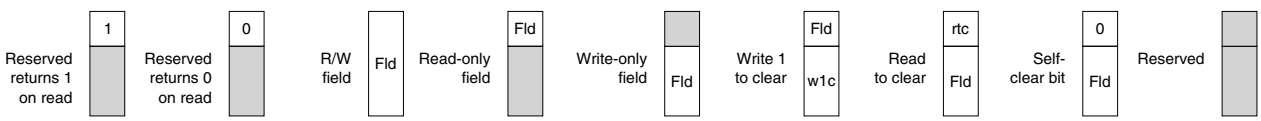


Figure 1-1. Register Field Conventions

NOTE

For reserved register fields, software should mask off the data in the field after read (software can't rely on the contents of data read from a reserved field) and always write all zeros.

1.1.4.2 Register Macro Usage

A common operation is to update one field without disturbing the contents of the remaining fields in the register. Normally, this requires a read-modify-write (RMW) operation, where the CPU reads the register, modifies the target field, then writes the results back to the register. This is an expensive operation in terms of CPU cycles, because of the initial register read.

To address this issue, some hardware registers are implemented as a group, including registers that can be used to either set, clear, or toggle (SCT) individual bits of the primary register. When writing to an SCT register, all bits set to 1 perform the associated operation on the primary register, while all bits set to 0 are not affected. The SCT registers always read back 0, and should be considered write-only. The SCT registers are not implemented if the primary register is read-only.

With this architecture, it is possible to update one or more fields using only register writes. First, all bits of the target fields are cleared by a write to the associated clear register, then the desired value of the target fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have one potential drawback. Whenever a field is modified, the hardware sees a value of 0 before the final value is written. For most fields, passing through the 0 state is not a problem. Nonetheless, this behavior is something to consider when using a CS operation.

Also, a CS operation is not required for fields that are one bit wide. While the CS operation works in this case, it is more efficient to simply set or clear the target bit (that is, one write instead of two). A simple set or clear operation is also atomic, while a CS operation is not.

Note that not all macros for set, clear, or toggle (SCT) are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of read/modify/write operations. When atomic operation is required, the developer should pay attention to this detail, because unexpected behavior might result if an interrupt occurs in the middle of the critical section comprising the update sequence.

1.1.5 Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document:

Table 1-3. Acronyms and Abbreviated Terms

Term	Meaning
BIST	Built-in self test
DDR	Double data rate (of Dynamic RAM)
FIFO	First In / First Out (of a queue)
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
EHCI	Enhanced host controller interface
EPROM	Erasable programmable read-only memory
GPIO	General-purpose I/O
GPR	General-purpose register
GPU	Graphicd Processing Unit
I2C or I2C	Inter-integrated circuit
IEEE	Institute of Electrical and Electronics Engineers
IrDA	Infrared Data Association
JTAG	Joint Test Action Group (a serial bus protocol usually used for test purposes)
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
MSB	Most-significant byte
msb	Most-significant bit
PCMCIA	Personal Computer Memory Card International Association
PIC	Programmable interrupt controller
POR	Power-on reset
RISC	Reduced instruction set computing
RTOS	Real-time operating system
Rx	Receive
SDLC	Synchronous data link control
SDMA	Serial DMA
SPDIF	Sony Phillips Digital Interface
SPI	Serial peripheral interface
SRAM	Static random access memory

Table continues on the next page...

Table 1-3. Acronyms and Abbreviated Terms (continued)

Term	Meaning
Tx	Transmit
UART	Universal asynchronous receiver/transmitter
USB	Universal serial bus

1.2 Target Applications

The i.MX50 processor provides designers with the power and flexibility necessary to support a wide variety of system designs including:

- eReaders
- Smartphones
- Gaming consoles
- PDAs
- Portable media players (PMPs)
- Web tablets
- Other media devices

1.3 Features

The i.MX50 Application Processor (AP) is based on the ARM Cortex A8™ platform, which has the following features:

- Memory Management Unit (MMU), L1 Instruction Cache, and L1 Data Cache
- Unified L2 cache
- 800 MHz target frequency of the core (including Neon, VFP Version 3, and L1 cache)
- Neon coprocessor (SIMD Media Processing Architecture) and Vector Floating Point (VFP-Lite) coprocessor supporting VFPv3

The memory system consists of the following components:

- Level 1 Cache:
  - Instruction (32 Kbyte)
  - Data (32 Kbyte)
- Level 2 Cache:
  - Unified instruction and data (256 Kbyte)
- Level 2 (internal) memory:

    - Boot ROM, including High Assurance Boot (96 Kbyte)
    - Internal multimedia/shared, fast access RAM (128 Kbyte)
- External memory interfaces:
    - 16/32-bit DDR2-533, LPDDR2-533, or LPDDR1-400 up to a total of 2 Gbyte
    - 8-bit NAND SLC/MLC Flash, up to 100 MHz, with up to 32-bit hardware ECC with up to 1 Kbyte block size.
    - 16/32-bit NOR Flash, with a dedicated 16-bit muxed-mode interface. I/O muxing logic selects EIM port as primary muxing at boot.
    - 16-bit PSRAM, Cellular RAM
    - Managed NAND including eMMC up to rev 4.4

The i.MX50 introduces a next generation system bus fabric architecture which aggregates various subsystem buses and masters for access to system peripherals and memories. The various bus systems and components are as follows:

- 64-bit AXI Fabric (266 MHz)—This bus fabric is the SoC's central bus aggregation point.
    - Provides access to all slave targets in the SoC:
        - ROM (ROMCP)
        - On-chip RAM (OCRAM)
        - DRAM Memory Controller (DRAMMC)
        - External static RAM (EIM)
        - Interrupt controller (TZIC)
        - Decode into the AHB MAX crossbar second level AHB fabric.
    - Provides arbitration to the following masters in the system:
        - ARM platform complex
        - Pixel processing pipeline (ePXP)
        - Electrophoretic display controller (EPDC)
        - LCD display controller (eLCDIF)
        - Data Co-processor (DCP) Crypto engine
        - Error Correcting Accelerator (BCH)
        - 2D GPU
        - Smart Direct Memory Access (SDMA)
        - USB On the Go and host controller complex (USB)
        - Fast Ethernet controller (FEC)
- MAX AHB crossbar (133 MHz)—This connects the various AHB bus sub-segments in the system and provides decode into the following slaves:
    - IP-Bus 1 (66 MHz)—This bus segment contains peripherals accessible by the ARM core and without DMA capability

    - IP-Bus 2 (66 MHz)—This bus segment contains peripherals accessible by the ARM core and without DMA capability
    - APBH DMA bridge (133 MHz)—The APBH DMA bridge is a master to the MAX for its memory-side DMA operations. The APBH bus is an AMBA APB slave bus providing peripheral access to many of the high-speed IP blocks on the i.MX50.
- IP-Bus 3 (66 MHz): This third peripheral bus segment contains peripherals accessible by the ARM core and SDMA and as such houses peripherals with DMA capability. The IP-Bus 3 can be accessed by the ARM platform through IP-Bus 1 and SPBA.
- Quality of Service Controller (QoSC): This provides both soft and dynamic arbitration/priority control. The QoSC works in conjunction with the critical display modules such as the eLCDIF and EPDC to provide dynamic priority control, based on real-time metrics.

The i.MX50 makes use of dedicated hardware accelerators to achieve state-of-the-art multimedia performance. The use of hardware accelerators provides both high performance and low power consumption, while freeing up the ARM core for other tasks.

The i.MX50 incorporates the following hardware accelerators:

- GPU2D —2D Graphics accelerator, OpenVG 1.1, version 1, 200 Mpix/s performance
- ePXP—enhanced PiXel Processing Pipeline off loading key pixel processing operations required to support both LCD and EPD display applications.

The i.MX50 includes the following interfaces to external devices:

NOTE

Not all the interfaces are available simultaneously depending on I/O multiplexer configuration.

- Displays:
    - EPDC—Supporting direct-driver TFT backplanes beyond 2048 × 1536 at 106 Hz refresh (or 4096 × 4096 at 20 Hz)
    - eLCDIF—Supporting beyond SXGA + (1400 × 1050) at 60 Hz resolutions with up to a 32-bit display interface
    - Both displays can be active simultaneously. If both displays are active, the eLCDIF only provides a 16-bit interface due to pin multiplexing.
- Expansion cards:
    - Four SD/MMC cards
- USB:

    - One High Speed (HS) USB 2.0 OTG-capable port with integrated HS USB PHY
    - One High Speed (HS) USB 2.0 host port with integrated HS USB PHY
- Miscellaneous interfaces:
    - One-Wire (OWIRE) port
    - Two internal I2S/SSI/AC97 ports, supporting up to 1.4 Mbps each connected to the Digital Audio Multiplexer (AUDMUX) with four external ports
    - Five UART RS232 ports, up to 4.0 Mbps each.
    - Two eCSPI (Enhanced CSPI) ports, up to 66 Mbps each plus a CSPI port, up to 16.6 Mbps
    - Three I^2C ports, supporting 400 kbps
    - Fast Ethernet controller IEEE 802.3, 10/100 Mbps
    - Sony Memory Stick Pro Host Controller
    - Key Pad Port (KPP)
    - Two Pulse Width Modulators (PWM)
    - GPIO with interrupt capabilities
    - System JTAG Controller (SJC)

The system supports efficient and smart power control and clocking:

- Suppors DVFS techniques for low power modes, including autos-low from the STMP architecture
- Power gating-SRPG (State Retention Power Gating) for ARM core and Neon
- Support for various levels of system power modes
- Flexible clock gating control scheme
- On-chip temperature monitor
- On-chip 32 KHz and 24 MHz oscillators
- A total of four PLLs with the fourth PLL providing up to eight independently controllable outputs, improving the ease of clocking control especially for display and connectivity modules

Security functions are enabled and accelerated by the following hardware:

- System JTAG Controller (SJC)—Protecting JTAG from debug port attacks by regulating or blocking the access to the system debug features
- Secure Real-Time Clock (SRTC)—Tamper resistant RTC with dedicated power domain and mechanism to detect voltage and clock glitches
- Advanced High Assurance Boot (A-HAB)—HAB with the next embedded enhancements: SHA-256, 2048-bit RSA key, version control mechanism, warm boot, CSU, and TZ initialization

1.4 Architectural Overview

This section contains a simplified block diagram of the i.MX50.

1.4.1 Block Diagram

Figure 1-2 shows a high-level block diagram of the i.MX50, providing a view of the major subsystems (processor domains, shared peripherals domain, memories, and so on) and logical connectivity.

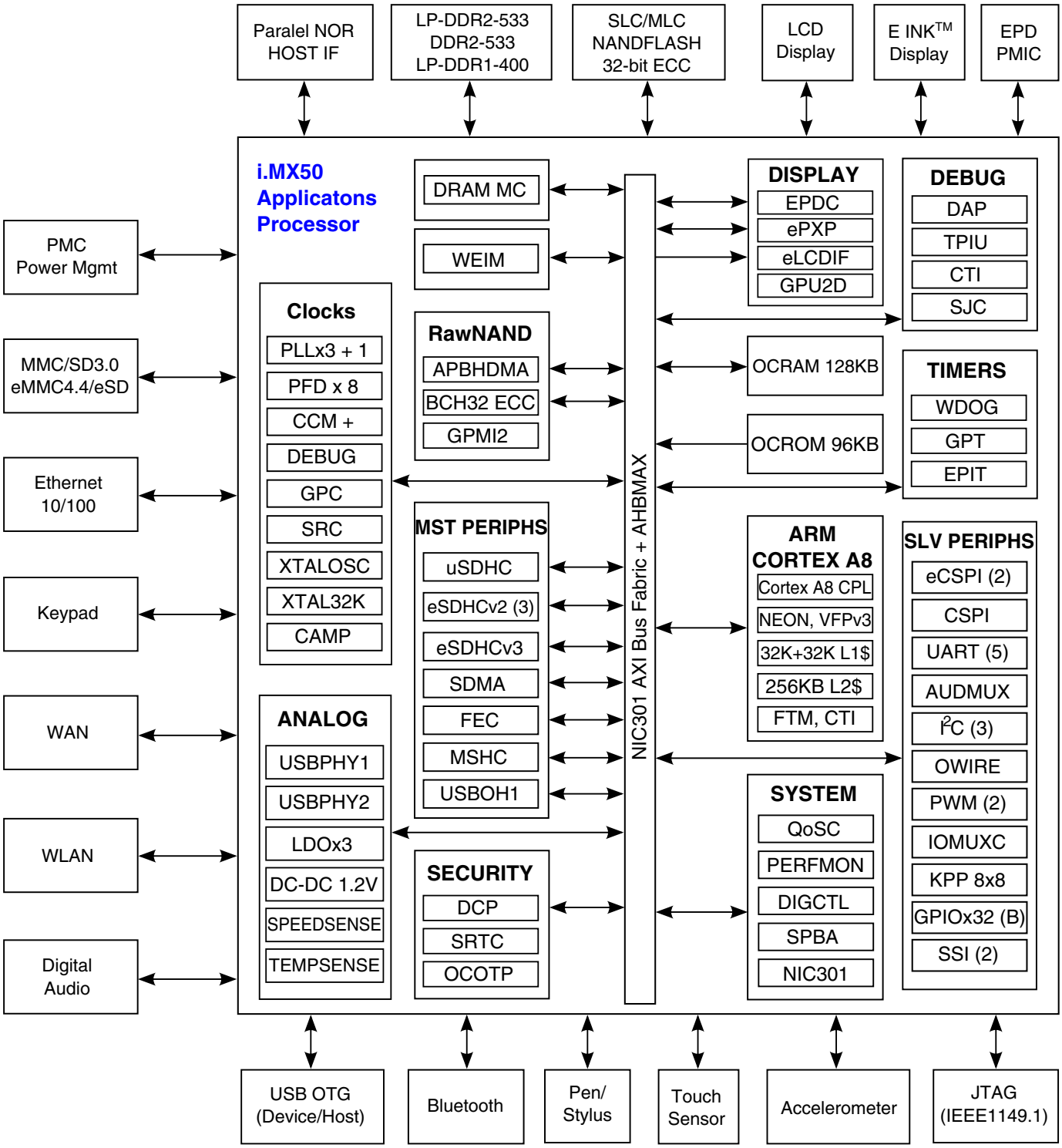


Figure 1-2. i.MX50 System Block Diagram

1.4.2 Major Subsystems

The i.MX50 consists of the following major subsystems:

- Core (ARM Cortex A8™) Platform, L1/L2 memories
- SDMA controller and the shared peripheral domain
- System Control—Boot Flow control, Clocks distribution, **Reset** control and Low Power logic
- Multimedia
  - Video Processor
  - TV-Encoder (TVe-) for PAL/NTSC output
  - Image Processor
  - Graphics Processors
  - Audio—connectivity interfaces in hardware, while codecs are performed in SW by ARM core.
- Security
- Connectivity peripherals
- External Memory Interface

1.4.3 Architectural Partitioning

This paragraph defines how architecture supports the processing intensive tasks:

- ARM Cortex A8™ Platform is responsible for:
  - Operating System
  - User applications (including control over hardware accelerators and non-accelerated functions)
- Smart DMA enables data transfer between non-mastering peripherals and external or internal memories
- System Control is supported through:
  - Clock Control Block (CCM)
  - Three (3×) PLLs
  - XTALOSC—Crystal oscillator source support
  - Frequency Pre-multiplier (FPM)
  - System Reset Controller (SRC)
  - Global Power Controller (GPC)
  - Two (2×) CAMPs—Clock Amplifier modules on the CKIH1 and CKIH2 inputs
- Multimedia is supported with:
  - LCD Interface (eLCDIF) ,
    - Pixel processing pipeline (ePXP), supports
- YUV bi-linear scaling, upto 8 overlays, rotation function, in-place rendering, and high resolutions upto 2048x2048.
- Electrophoretic Display Controller (EPDC)
- Graphics Processing Unit (GPU2D), 2D graphics processing
- Audio

    - Audio codecs are provided by software, which runs on ARM core
    - 2 × SSIs
    - Audio Mux
- Security is supported by:
    - High Assurance Boot (HAB) System
    - On-Chip OTP Controller (OCOTP) for the One-Time programmable electrical fuse array (E-Fusebox)
    - Data co-processor (DCP)
    - SJC—System JTAG controller
    - Secure Real Time Clock (SRTC)
    - Tamper Detection
- Connectivity peripherals, timers, and External Memory Interface (EMI)
    - Low-level communication protocols
    - Embedded DMAs
    - 3.3-V IO voltage for seamless integration
    - USB 2.0 with integrated PHY/Transceiver
    - DDR, Nand Flash (MLC 4/8-bit ECC) memory interface
    - Timers: 2 × EPIT, GPT, and Watch Dog timer (WDOG)
    - Miscellaneous connectivity support—I2C, SPI, UART, PWM, and Keypad interface

1.5 i.MX50 Modules List

Table 1-4 lists the modules used by the various subsystems of the i.MX50.

Table 1-4. i.MX50 Digital and Analog Modules

Block Mnemonic	Block Name	Subsystem	Brief Description
ARM Cortex A8™	ARM Cortex A8™ Platform	ARM	The ARM Cortex A8™ Core Platform consists of the ARM Cortex A8™ processor and its essential sub-blocks. It contains the 32 Kbyte L1 instruction cache, 32 Kbyte L1 data cache, Level 2 cache controller and a 256 Kbyte L2 cache. The platform also contains an event monitor and debug modules. It also has a NEON co-processor with SIMD media processing architecture, register file with 32 × 64-bit general-purpose registers, an Integer execute pipeline (ALU, Shift, MAC), dual, single-precision floating point execute pipeline (FADD, FMUL), load/store and permute pipeline, and a non-pipelined vector floating point (VFP Lite) co-processor supporting VFPv3.

Table continues on the next page...

Table 1-4. i.MX50 Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Subsystem	Brief Description
EPDC	Electrophoretic Display Controller	Display	The EPDC is a feature-rich, low power, and high-performance direct-drive active matrix EPD controller. It is specifically designed to drive E-INK™ EPD panels supporting a wide variety of TFT backplanes. The goal of the EPDC is to provide an efficient SoC integration of this functionality.
ePXP	enhanced PiXel Processing Pipeline	Display	A high-performance pixel processor is capable of 1 pixel/clock performance for combined operations such as color-space conversion, alpha blending, gamma-mapping, and rotation. The ePXP is enhanced with features specifically for grayscale applications. In addition, the ePXP supports traditional pixel/frame processing paths for still-image and video processing applications allowing it to interface with the integrated LCD controller (eLCDIF).
eLCDIF	enhanced LCD Interface	Display	The eLCDIF is a high-performance LCD controller interface supporting a rich set of modes allowing inter operability with a wide variety of LCD panels (including smart-panels, DOTCK/RGB, and so on). The module also supports a synchronous operation with the ePXP to allow the processed frames to be passed from the ePXP to the eLCDIF through an on-chip SRAM buffer. The eLCDIF can support up to 32-bit interfaces.
AUDMUX	Digital Audio Mux	Multimedia Peripherals	The AUDMUX is a programmable interconnect for voice, audio, and synchronous data routing between host serial interfaces (for example, SSI1, SSI2) and peripheral serial interfaces (audio and voice codecs). The AUDMUX has seven ports (three internal and four external) with identical functionality and programming models. A desired connectivity is achieved by configuring two or more AUDMUX ports.
CAMP-1	Clock Amplifier	Clocks, Resets, and Power Control	Clock Amplifier
CCM GPC SRC	Clock Control Module General Power Controller System Reset Controller	Clocks, Resets, and Power Control	These modules are responsible for clock and reset distribution in the system, and also for system power management. The system includes four PLLs.
CSPI eCSPI-1 eCSPI-2	Configurable SPI, Enhanced CSPI	Connectivity Peripherals	Full-duplex enhanced synchronous serial interface, with data rate up to 66.5 Mbit/s (for eCSPI, master mode). It is configurable to support Master/Slave modes, four chip selects to support multiple peripherals.

Table continues on the next page...

Table 1-4. i.MX50 Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Subsystem	Brief Description
Debug System	Debug System	System Control	The Debug System provides real-time trace debug capability of both instructions and data. It supports a trace protocol that is an integral part of the ARM Real Time Debug solution (RealView). Real-time tracing is controlled by specifying a set of triggering and filtering resources, which include address and data comparators, three cross-system triggers (CTI), counters, and sequencers. Debug Access Port (DAP)—The DAP provides real-time access for the debugger without halting the core to System memory and peripheral registers. All debug configuration registers and Debugger access to JTAG scan chains.
DRAM MC	DRAM Memory Controller	Connectivity Peripherals	The DRAM Memory Controller multiple external memory types, including: Standard 1.8V DDR2, 1.8V Mobile DDR1 (LPDDR1), 1.2V Mobile DDR2 (LPDDR2).
EIM	Wireless External Interface Module (EIM)	Connectivity Peripherals	The Wireless External Interface Module (EIM) handles the interface to devices external to the chip, including generation of chip selects, clock and control for external peripherals and memory. It provides asynchronous access to devices with SRAM-like interface and synchronous access to devices with Nor-Flash like or PSRAM like interface.
BCH32/GPMI	Raw NAND System with ECC	Connectivity Peripherals	The General Purpose Media Interface(GPMI) is a flexible interface to up to eight NAND flash with configurable address and command behavior, providing support for ONFI2.1 standard and Samsung Toggle-DDR NAND devices. A hardware BCH ECC engine is built in allowing for up to 32-bit correction with programmable NAND page layout.
APBHDMA	AHB-to-APBH Bridge with DMA	System Control Peripherals	The AHB-to-APBH bridge includes the AHB-to-APB PIO bridge for memory-mapped I/O to the APB devices, as well as a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM core.
DCP	Data Co-Processor	Security	It includes sections on memory copy functionality, Advanced Encryption Standard (AES), hashing, and arbitration
System Fabric and QoS	System Fabric and QoS	System	The Quality of Service Controller APB Slave provides the following features: • Provide software programmability of the master ports of the AXI fabric(NIC301). • Provide translation and pass through of AXI QOS values to the AWPRIORITY and ARPRIORITY sideband signals of the DRAM Memory Controller (DRAM MC). Alternatively allow independent programmability of the DRAM MC priority sideband signals. • Automatic qos/priority boost for eEPDC and eLCDIF masters. • Per port disabling of AXI master port traffic.

Table continues on the next page...

Table 1-4. i.MX50 Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Subsystem	Brief Description
EPIT	Enhanced Periodic Interrupt Timer	Timer Peripherals	Each EPIT is a 32-bit set and forget timer that starts counting after the EPIT is enabled by software. It is capable of providing precise interrupts at regular intervals with minimal processor intervention. It has a 12-bit prescaler for division of input clock frequency to get the required time setting for the interrupts to occur, and counter values can be programmed on the fly.
eSDHCv3-3 (eMMC 4.4)	Ultra-High- Speed Multi-Media Card/ Secure Digital card host controller, ver. 3	Connectivity Peripherals	Ultra High-Speed eSDHC, enhanced to support eMMC 4.4 standard specification, for 832 MBps. Port 3 is specifically enhanced to support eMMC 4.4 specification, for double data rate (832 Mbps, 8-bit port). eSDHCv3 is backward compatible to eSDHCv2 IP and supports all the features of eSDHCv2 as described below.
eSDHCv2-1 eSDHCv2-2 eSDHCv2-4	Enhanced Multi-Media Card/ Secure Digital Host Controller, ver. 2		Enhanced Multi-Media Card/Secure Digital Host Controller: Ports 1, 2, and 4 are compatible with the MMC System Specification version 4.3, full support Port 4 supports up to 8-bit data width, ports 1 and 2 are limited to 4-bit data width interface The generic features of the eSDHCv2 module, when serving as SD/MMC host, include the following: Can be configured either as SD/MMC controller Supports eSD and eMMC standard, for SD/MMC embedded type cards Conforms to *SD Host Controller Standard Specification* version 2.0, full support. Compatible with the SD Memory Card Specification version 1.1 Compatible with the SDIO Card Specification version 1.2 Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC and MMC RS cards Configurable to work in one of the following modes: - SD/SDIO 1-bit, 4-bit - MMC 1-bit, 4-bit, 8-bit (8-bit supported only on port 4) Full/High speed mode Host clock frequency variable between 32 kHz to 52 MHz Up to 200 Mbps data transfer for SD/SDIO cards using four parallel data lines Up to 416 Mbps data transfer for MMC cards using eight parallel data lines (port 4)

Table continues on the next page...

Table 1-4. i.MX50 Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Subsystem	Brief Description
FEC	Fast Ethernet Controller	Connectivity Peripherals	The Ethernet Media Access Controller (MAC) is designed to support both 10 Mbps and 100 Mbps Ethernet/IEEE Std 802.3™ networks. An external transceiver interface and transceiver function are required to complete the interface to the media.
GPIO-1 GPIO-2 GPIO-3 GPIO-4 GPIO-5 GPIO-6	General Purpose I/O Modules	System Control Peripherals	These modules are used for general purpose input/output to external ICs. Each GPIO module supports up to 32 bits of I/O.
GPT	General Purpose Timer	Timer Peripherals	Each GPT is a 32-bit *free-running* or *set* and *forget* mode timer with a programmable prescaler and compare and capture register. A timer counter value can be captured using an external event, and can be configured to trigger a capture event on either the leading or trailing edges of an input pulse. When the timer is configured to operate in "set and forget" mode, it is capable of providing precise interrupts at regular intervals with minimal processor intervention. The counter has output compare logic to provide the status and interrupt at comparison. This timer can be configured to run either on an external clock or on an internal clock.
GPU2Dv1	Graphics Processing Unit-2D, ver. 1	Multimedia Peripherals	The GPU2Dv1 provides hardware acceleration for 2D graphic algorithms with sufficient processor power to run desk-top quality interactive graphics applications on displays up to HD1080 resolution.
I^2C-1 I^2C-2 I^2C-3	I^2C Interface	Connectivity Peripherals	I^2C provides serial interface for controlling peripheral devices. Data rates of up to 400 kbps are supported.
OCOTP	On-chip OTP controller	Security	The on-chip one-time -programmable (OCOTP) ROM serves the functions of hardware and software capability bits, Freescale operations and unique-ID, the customer-programmable cryptography key, and storage of various ROM configuration bits.
IOMUXC	IOMUX Control	System Control Peripherals	This module enables flexible I/O multiplexing. Each I/O pad has default as well as several alternate functions. The alternate functions are software configurable.
KPP	Keypad Port	Connectivity Peripherals	The KPP supports an 8 × 8 external keypad matrix. The KPP features are as follows: Open drain design Glitch suppression circuit design Multiple keys detection Standby key press detection

Table continues on the next page...

Table 1-4. i.MX50 Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Subsystem	Brief Description
OWIRE	One-Wire Interface	Connectivity Peripherals	One-Wire support provided for interfacing with an on-board EEPROM, and smart battery interfaces, for example, Dallas DS2502.
PWM-1 PWM-2	Pulse Width Modulation	Connectivity Peripherals	The pulse-width modulator (PWM) has a 16-bit counter and is optimized to generate sound from stored sample audio images. It can also generate tones. The PWM uses 16-bit resolution and a 4 x 16 data FIFO to generate sound.
OC RAM 128 Kbytes	On-Chip RAM	Internal Memory	Internal RAM The On-Chip Memory controller (OCRAM) module, is an interface between the system's AXI bus, to the internal (on-chip) SRAM memory module. It is used for controlling the 128 Kbyte multimedia RAM, through a 64-bit AXI bus.
OC ROM 96 Kbytes	On-Chip ROM	Internal Memory	Supports secure and regular Boot Modes. The On-Chip ROM Controller supports ROM Patching.
SDMA	Smart Direct Memory Access	System Control Peripherals	The SDMA is multi-channel flexible DMA engine. It helps in maximizing system performance by offloading various cores in dynamic data routing. The SDMA features list is as follows: Powered by a 16-bit instruction-set micro-RISC engine Multi-channel DMA supports up to 32 time-division multiplexed DMA channels 48 events with total flexibility to trigger any combination of channels Memory accesses including linear, FIFO, and 2D addressing Shared peripherals between ARM Cortex A8™ and SDMA Very fast context-switching with two-level priority-based preemptive multi-tasking DMA units with auto-flush and prefetch capability Flexible address management for DMA transfers (increment, decrement, and no address changes on source and destination address) DMA ports can handle uni-directional and bi-directional flows (copy mode) Up to 8-word buffer for configurable burst transfers for PL301 Support of byte-swapping and CRC calculations A library of scripts and API is available

Table continues on the next page...

Table 1-4. i.MX50 Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Subsystem	Brief Description
SJC	System JTAG Interface	System Control Peripherals	The System JTAG controller provides a mechanism for regulating JTAG access, preventing unauthorized JTAG usage while allowing JTAG access for manufacturing tests and software debugging. The i.MX50 JTAG port provides debug access to several hardware blocks including the ARM processor and the system bus, therefore, it must be accessible for initial laboratory bring-up,manufacturing tests and troubleshooting, and for software debugging by authorized entities. However, if the JTAG port is left unsecured it provides a method for executing unauthorized program code, getting control over secure applications, and running code in privileged modes. The System JTAG controller provides three different security modes that can be selected through an e-fuse configuration to prevent unauthorized JTAG access.
SPBA	Shared Peripheral Bus Arbiter	System Control Peripherals	SPBA (Shared Peripheral Bus Arbiter) is a two-to-one IP bus interface (IP bus) arbiter.
SRTC	Secure Real Time Clock	Security	The SRTC incorporates a special System State Retention Register (SSRR) that stores system parameters during system shutdown modes. This register and all SRTC counters are powered by dedicated supply rail NVCC_SRTC_POW. The NVCC_SRTC_POW can be energized separately even if all other supply rails are shut down. This register is helpful for storing warm boot parameters. The SSRR also stores the system security state. In case of a security violation, the SSRR marks the event (security violation indication).
SSI-1 SSI-2	I2S/SSI/AC97 Interface	Connectivity Peripherals	The SSI is a full-duplex synchronous interface used on the i.MX50 processor to provide connectivity with off-chip audio peripherals. The SSI interfaces connect internally to the AUDMUX for mapping to external ports. The SSI supports a wide variety of protocols (SSI normal, SSI network, I2S, and AC-97), bit depths (up to 24 bits per word), and clock/frame sync options. Each SSI has two pairs of 15 x 24 FIFOs and hardware support for an external DMA controller in order to minimize its impact on system performance. The second pair of FIFOs provides hardware interleaving of a second audio stream, which reduces ARM platform overhead in use cases where two time slots are being used simultaneously.
Temperature Monitor	(Part of SATA IP)	System Control Peripherals	The temperature sensor is an internal module to the i.MX50 that monitors the die temperature. The monitor is capable of generating a SW interrupt, or triggering the CCM, to reduce the core operating frequency.

Table continues on the next page...

Table 1-4. i.MX50 Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Subsystem	Brief Description
UART-1 UART-2 UART-3 UART-4 UART-5	UART Interface, ver. 2	Connectivity Peripherals	Each of the UARTv2 modules supports the following serial data transmit/receive protocols and configurations: 7 or 8-bit data words, 1 or 2 stop bits, programmable parity (even, odd, or none) Programmable bit-rates up to 4 Mbps. This is a higher max baud rate relative to the 1.875 Mbps, which is specified by the TIA/EIA-232-F standard. 32-byte FIFO on Tx and 32 half-word FIFO on Rx supporting auto-baud IrDA 1.0 support (up to SIR speed of 115200 bps) Option to operate as 8-pins full UART, DCE, or DTE
USB-OH-1	USB 2.0 High-Speed OTG-capable and Host ports	Connectivity Peripherals	USB-OH-1 supports USB2.0 HS/FS/LS, and contains: One high-speed OTG-capable module with integrated HS USB PHY One high-speed Host module with integrated HS USB PHY.
WDOG-1	Watch Dog	Timer Peripherals	The Watchdog (WDOG) timer module protects against system failures by providing a method of escaping from unexpected events or programming errors. The WDOG Timer supports two comparison points during each counting period. Each of the comparison points is configurable to invoke an interrupt to the ARM core, and a second point invokes an external event on the WDOG line.
XTALOSC	Crystal Oscillator I/F	Clocking	The XTALOSC module allows connectivity to an external crystal.

Chapter 2
Memory Map

2.1 Introduction

This chapter introduces the memory architecture of the i.MX50 device. The system memory high-level partition is defined below:

- Level 1 Cache
  - Instruction (32 Kbyte)
  - Data (32 Kbyte)
- Level 2 Cache
  - Unified instruction and data (256 Kbyte)
- Level 2 (internal) Memory
  - Boot ROM, including HAB (96 Kbyte)
  - Internal multimedia/shared, fast access RAM (128 Kbyte).
- External memory interfaces
  - 16/32-bit DDR2-533, LP-DDR2-533, or LP-DDR1-400 up to a total of 2 Gbyte
  - 8-bit NAND SLC/MLC Flash, up to 100 MHz, with up to 32-bit hardware ECC with up to 1 Kbyte block size.
  - 16/32-bit NOR Flash, with a dedicated 16-bit muxed-mode interface. I/O muxing logic selects EIM port as primary muxing at system boot.
  - 16-bit PSRAM, Cellular RAM
  - Managed NAND including eMMC up to rev 4.4

2.2 ARM Platform Memory Map

Table 2-1 shows the system memory map.

Table 2-1. System Memory Map

AP		Size (Bytes)	Region
Start Address	End Address		
On Chip Memories			
0000_0000	0001_7FFF	96 K	Boot ROM
0001_8000	0001_FFFF	32 K	Reserved (Boot ROM Hole)
0002_0000	00FF_FFFF	16 M-128 K	Reserved
0100_0000	0FFF_BFFF	240 M-16 K	Reserved
0FFF_C000	0FFF_FFFF	16 K	TZIC
1000_0000	13FF_FFFF	64 M	Reserved
1400_0000	17FF_FFFF	64 M	DRAM MC
1800_0000	1FFF_FFFF	128 M	Reserved
2000_0000	2FFF_FFFF	256 M	GPU2D
3000_0000	3GFF_FFFF	256 M	Reserved
On Chip AHB Accessed IPs-Debug APB			
4000_0000	4000_0FFF	4 K	Debug ROM
4000_1000	4000_1FFF	4 K	ETB
4000_2000	4000_2FFF	4 K	ETM
4000_3000	4000_3FFF	4 K	TPIU
4000_4000	4000_4FFF	4 K	CTI0
4000_5000	4000_5FFF	4 K	CTI1
4000_6000	4000_6FFF	4 K	CTI2
4000_7000	4000_7FFF	4 K	CTI3
4000_8000	4000_8FFF	4 K	Cortex Debug Unit
4000_9000	40FF_FFFF	16M - 36K	Reserved
4100_0000	4100_1FFF	8 K	ABPHDMA
4100_2000	4100_3FFF	8 K	OCOTP
4100_4000	4100_5FFF	8 K	DIGCTL
4100_6000	4100_7FFF	8 K	GPMI
4100_8000	4100_9FFF	8 K	BCH
4100_A000	4100_BFFF	8 K	eLCDIF
4100_C000	4100_DFFF	8 K	ePXP
4100_E000	4100_FFFF	8 K	DCP
4101_0000	4101_1FFF	8 K	EPDC
4101_2000	4101_3FFF	8 K	QoSC
4101_4000	4101_5FFF	8 K	PERFMON
4101_6000	47FF_7FFF	8 K	Reserved

Table continues on the next page...

Table 2-1. System Memory Map (continued)

AP		Size (Bytes)	Region
Start Address	End Address		
4101_8000	4101_9FFF	8 K	CCM ANALOG
4101_8080	4101_808F	16	TEMPSENSOR
4101_A000	47FF_FFFF	112M - 104K	Reserved
4800_0000	480F_FFFF	1 M	NIC301
4810_0000	4FFF_FFFF	127 M	Reserved
AIPS_TZ#1			
AIPS_TZ#1-SPBA IPs, Mapped to global module enable 0			
5000_0000	5000_3FFF	16 K	Reserved
5000_4000	5000_7FFF	16 K	ESDHC1
5000_8000	5000_BFFF	16 K	ESDHC2
5000_C000	5000_FFFF	16 K	UART3
5001_0000	5001_3FFF	16 K	eCSPI1
5001_4000	5001_7FFF	16 K	SSI2
5001_8000	5001_BFFF	16 K	Reserved
5001_C000	5001_FFFF	16 K	Reserved for SDMA internal registers
5002_0000	5002_3FFF	16 K	ESDHC3
5002_4000	5002_7FFF	16 K	ESDHC4
5002_8000	5002_bFFF	16 K	Reserved
5002_C000	5002_FFFF	16 K	Reserved
5003_0000	5003_3FFF	16 K	Reserved
5003_4000	5003_7FFF	16 K	Reserved
5003_8000	5003_BFFF	16 K	Reserved
5003_C000	5003_FFFF	16 K	SPBA
AIPS_TZ#1-Global Module Enables			
5004_0000	51FF_FFFF	32 M (minus 256 K)	Reserved AIPS_TZ #1 off platform global module enable #0
5200_0000	53EF_FFFF	31 M	Reserved AIPS_TZ #1 off platform global module enable #1
AIPS_TZ#1-On Platform			
53F0_0000	53F7_FFFF	512 K	Reserved AIPS_TZ #1 on platform slots
AIPS_TZ#1-Off Platform			
53F8_0000	53F8_3FFF	16 K	USBOH1 (PORT USB)
53F8_4000	53F8_7FFF	16 K	GPIO1
53F8_8000	53F8_BFFF	16 K	GPIO2
53F8_C000	53F8_FFFF	16 K	GPIO3

Table continues on the next page...

Table 2-1. System Memory Map (continued)

AP		Size (Bytes)	Region
Start Address	End Address		
53F9_0000	53F9_3FFF	16 K	GPIO4
53F9_4000	53F9_7FFF	16 K	KPP
53F9_8000	53F9_BFFF	16 K	WDOG1
53F9_C000	53F9_FFFF	16 K	Reserved
53FA_0000	53FA_3FFF	16 K	GPT
53FA_4000	53FA_7FFF	16K	SRTC
53FA_8000	53FA_BFFF	16 K	IOMUXC
53FA_C000	53FA_FFFF	16 K	EPIT1
53FB_0000	53FB_FFFF	16 K	Reserved
53FB_4000	53FB_7FFF	16 K	PWM1
53FB_8000	53FB_BFFF	16 K	PWM2
53FB_C000	53FB_FFFF	16 K	UART1
53FC_0000	53FC_3FFF	16 K	UART2
53FC_4000	53FC_7FFF	16 K	USBOH1 (PORT PL301)
53FC_8000	53FC_BFFF	16 K	Reserved
53FC_C000	53FC_FFFF	16 K	Reserved
53FD_0000	53FD_3FFF	16 K	SRC
53FD_4000	53FD_7FFF	16 K	CCM
53FD_8000	53FD_BFFF	16 K	GPC
53FD_C000	53FD_FFFF	16 K	GPIO5
53FE_0000	53FE_3FFF	16 K	GPIO6
53FE_4000	53FE_7FFF	16 K	Reserved
53FE_8000	53FE_BFFF	16 K	Reserved
53FE_C000	53FE_FFFF	16 K	I2C3
53FF_0000	53FF_3FFF	16 K	UART4
53FF_4000	53FF_7FFF	16 K	Reserved
53FF_8000	53FF_BFFF	16 K	RNGB_BLOCK
53FF_C000	53FF_FFFF	16 K	Reserved AIPS_TZ #1 off platform space.
5400_0000	5FFF_FFFF	448 M	Reserved (Aliased to AIPS_TZ#1 slots)
AIPS_TZ#2-Global Module Enables			
6000_0000	61FF_FFFF	32 M	Reserved AIPS_TZ #1 off platform global module enable #0
6200_0000	63EF_FFFF	31 M	Reserved AIPS_TZ #1 off platform global module enable #1
AIPS_TZ#2-On Platform			

Table continues on the next page...

Table 2-1. System Memory Map (continued)

AP		Size (Bytes)	Region
Start Address	End Address		
63F0_0000	63F7_FFFF	512 K	Reserved AIPS_TZ #2 on platform slots
AIPS_TZ#2-Off Platform			
63F8_0000	63F8_3FFF	16 K	DPLLC-1
63F8_4000	63F8_7FFF	16 K	DPLLC-2
63F8_8000	63F8_BFFF	16 K	DPLLC-3
63F8_C000	63F8_FFFF	16 K	Reserved
63F9_0000	63F9_3FFF	16 K	UART5
63F9_4000	63F9_7FFF	16 K	AHBMAX
63F9_8000	63F9_BFFF	16 K	Reserved
63F9_C000	63F9_FFFF	16 K	Reserved
63FA_0000	63FA_3FFF	16 K	TIGERP_PLATFORM_NE_32K_256K
63FA_4000	63FA_7FFF	16 K	OWIRE
63FA_8000	63FA_BFFF	16 K	Reserved
63FA_C000	63FA_FFFF	16 K	eCSPI2
63FB_0000	63FB_3FFF	16 K	SDMA
63FB_4000	63FB_FFFF	16 K	Reserved
63FB_8000	63FB_BFFF	16 K	ROMCP
63FB_C000	63FB_FFFF	16 K	Reserved
63FC_0000	63FC_3FFF	16 K	CSPI
63FC_4000	63FC_7FFF	16K	I2C2
63FC_8000	63FC_BFFF	16 K	I2C1
63FC_C000	63FC_FFFF	16 K	SSI1
63FD_0000	63FD_3FFF	16 K	AUDMUX
63FD_4000	63FD_7FFF	16 K	Reserved
63FD_8000	63FD_BFFF	16 K	EIM
63FD_C000	63FD_FFFF	16 K	Reserved
63FE_0000	63FE_3FFF	16 K	Reserved
63FE_4000	63FE_7FFF	16 K	Reserved
63FE_8000	63FE_BFFF	16 K	Reserved
63FE_C000	63FE_FFFF	16 K	FEC
63FF_0000	63FF_3FFF	16 K	Reserved
63FF_4000	63FF_7FFF	16 K	Reserved
63FF_8000	63FF_BFFF	16 K	Reserved
63FF_C000	63FF_FFFF	16 K	Reserved

Table continues on the next page...

Table 2-1. System Memory Map (continued)

AP		Size (Bytes)	Region
Start Address	**End Address**		
6400_0000	6FFF_BFFF	256 M (minus 512 K)	Reserved (aliased AIPS_TZ #2 slots)
On Chip AHB Accessed IPs			
6FFF_C000	6FFF_FFFF	16 K	Reserved
Off Chip Memories			
7000_0000	EFFF_FFFF	2 G	DRAM
F000_0000 F000_0000 F400_0000	F7FE_FFFF F3FF_FFFF F7FE_FFFF	128 M-64 K 64 M 64 M-64 K	CS0 (128M NOR/SRAM)-Default Configuration CS1, CS2, CS3 - Not Active. In case of 2 active CS. CS0(64M) CS1(64M) Via IOMUXC (GPRR1) register, 4CS of 32M are supported. The sum of all CS spaces must equal 128M, and can be divided across a maximum of 4 CSs. The biggest region is CS0. No holes are supported. The supported configurations are: • CS0(128M), CS1 (0M), CS2 (0M), CS3(0M) • CS0(64M), CS1(64M), CS2(0M), CS3(0M) • CS0(64M), CS1(32M), CS2(32M), CS3(0M) • CS0(32M), CS1(32M), CS2(32M), CS3(32M)
On Chip Memories			
F7FF_0000	F7FF_FFFF	64 K	Reserved
F800_0000	F801_FFFF	128 K	iRAM (OCRAM)
F802_0000	FFFF_FFFF	128 M-128 K	Reserved

NOTE

User should not address reserved memory regions. Access to reserved memory regions can cause unpredictable behavior.

2.3 DMA Memory Map

The Smart DMA's memory map is defined in Table 2-2.

Table 2-2. SDMA Peripheral Memory Map

Peripheral	Base Address	Size	Comments
Reserved for SDMA internal memory	0x0000	4 Kbyte	Reserved
ESDHCv2-1	0x1000	4 Kbyte	-
ESDHCv2-2	0x2000	4 Kbyte	-
UARTv2 3	0x3000	4 Kbyte	-
eCSPI1	0x4000	4 Kbyte	-
SSI2	0x5000	4 Kbyte	-
Reserved	0x6000	4 Kbyte	Reserved
Reserved for SDMA internal registers	0x7000	4 Kbyte	Reserved
ESDHCv2-3 (CE-ATA)	0x8000	4 Kbyte	-
ESDHCv3-4	0x9000	4 Kbyte	-
Reserved	0xA000	4 Kbyte	Reserved
Reserved	0xB000	4 Kbyte	Reserved
Reserved	0xC000	4 Kbyte	Reserved
Reserved	0xD000	4 Kbyte	Reserved
Reserved	0xE000	4 Kbyte	Reserved
SPBA Registers	0xF000	4 Kbyte	-

NOTE

User should not address reserved memory regions. Access to reserved memory regions can cause unpredictable behavior.

Chapter 3
Interrupts and DMA Events

3.1 Overview

The Interrupts and DMA Events chapter provides information on the assigments of interrupts of the AP domain in AP Interrupts and of the DMA events in SDMA Event Mapping.

3.2 AP Interrupts

The Interrupt Controller (TZIC) collects up to 128 interrupt requests from all i.MX50 sources and provides interface to the core. Software force registers and software priority masking are also supported. Table 3-1 details the ARM interrupt sources.

Table 3-1. ARM Domain Interrupt Summary

IRQ	Interrupt Source	Interrupt Description
0	Reserved	Reserved
1	eSDHC1	Enhanced SDHC1 Interrupt Request
2	eSDHC2	Enhanced SDHC2 Interrupt Request
3	eSDHC3	Enhanced SDHC3 Interrupt Request
4	uSDHC	Enhanced SDHC4 Interrupt Request
5	DAP	DAP Interrupt Request
6	SDMA	"AND" of all 48 interrupts from all the channels
7	IOMUX	External Interrupt Request, usually used as POWER FAIL interrupt
8	Reserved	Reserved
9	Reserved	Reserved
10	Reserved	Reserved
11	Reserved	Reserved
12	Reserved	Reserved

Table continues on the next page...

Table 3-1. ARM Domain Interrupt Summary (continued)

IRQ	Interrupt Source	Interrupt Description
13	UART4	UART4 ORed Interrupt Request
14	USBOH	USB Host Interrupt Request
15	Reserved	Reserved
16	Reserved	Reserved
17	Reserved	Reserved
18	USBOH	USB OTG Interrupt Request
19	DRAM MC	DRAM MCInterrupt Request
20	eLCDIF	eLCDIF Interrupt Request
21	ePXP	ePXP Interrupt Request
22	Reserved	Reserved
23	Reserved	Reserved
24	SRTC	SRTC Consolidated Interrupt Request (Non TZ)
25	SRTC	SRTC Security Interrupt Request (TZ)
26	Reserved	Reserved
27	EPDC	EPDC Interrupt Request
28	NIC	Perfmon Interrupt Request
29	SSI1	SSI1 Interrupt Request
30	SSI2	SSI2 Interrupt Request
31	UART1	UART1 ORed Interrupt Request
32	UART2	UART2 ORed Interrupt Request
33	UART3	UART3 ORed Interrupt Request
34	Reserved	Reserved
35	Reserved	Reserved
36	eCSPI1	eCSPI1 Interrupt Request
37	eCSPI2	eCSPI2 Interrupt Request
38	CSPI	CSPI Interrupt Request
39	GPT	GPT Interrupt Request
40	EPIT	EPIT output compare Interrupt Request
41	Reserved	Reserved
42	GPIO1	Interrupt Request for GPIO1 signal 7
43	GPIO1	Interrupt Request for GPIO1 signal 6
44	GPIO1	Interrupt Request for GPIO1 signal 5
45	GPIO1	Interrupt Request for GPIO1 signal 4
46	GPIO1	Interrupt Request for GPIO1 signal 3
47	GPIO1	Interrupt Request for GPIO1 signal 2

Table continues on the next page...

Table 3-1. ARM Domain Interrupt Summary (continued)

IRQ	Interrupt Source	Interrupt Description
48	GPIO1	Interrupt Request for GPIO1 signal 1
49	GPIO1	Interrupt Request for GPIO1 signal 0
50	GPIO1	Combined interrupt indication for GPIO1 signal 0 throughout 15
51	GPIO1	Combined interrupt indication for GPIO1 signal 16 throughout 31
52	GPIO2	Combined interrupt indication for GPIO2 signal 0 throughout 15
53	GPIO2	Combined interrupt indication for GPIO2 signal 16 throughout 31
54	GPIO3	Combined interrupt indication for GPIO3 signal 0 throughout 15
55	GPIO3	Combined interrupt indication for GPIO3 signal 16 throughout 31
56	GPIO4	Combined interrupt indication for GPIO4 signal 0 throughout 15
57	GPIO4	Combined interrupt indication for GPIO4 signal 16 throughout 31
58	WDOG1	Watchdog Timer reset Interrupt Request
59	Reserved	Reserved
60	KPP	Keypad Interrupt Request
61	PWM1	PWM1 Interrupt Request
62	I2C1	I2C1 Interrupt Request
63	I2C2	I2C2 Interrupt Request
64	I2C3	I2C3 Interrupt Request
65	Reserved	Reserved
66	Analog	Interrupt Request to indicate DC-DC OK
67	Analog	Interrupt Request of thermal alarm
68	Analog	Interrupt Request for signal irq_ana3
69	Analog	Interrupt Request for signal irq_ana4
70	Reserved	Reserved
71	CCM	CCM, Interrupt Request 1
72	CCM	CCM, Interrupt Request 2
73	GPC	GPC, Interrupt Request 1
74	GPC	GPC, Interrupt Request 2
75	SRC	SRC Interrupt Request
76	TIGERP PLATFORM	Neon Monitor Interrupt Request
77	TIGERP PLATFORM	Performance Unit Interrupt Request
78	TIGERP PLATFORM	CTI Interrupt Request
79	TIGERP PLATFORM	Debug Interrupt Request0, from Cross-Trigger Interface
80	TIGERP PLATFORM	Debug Interrupt Request1, from Cross-Trigger Interface
81	Reserved	Reserved
82	Reserved	Reserved

Table continues on the next page...

Table 3-1. ARM Domain Interrupt Summary (continued)

IRQ	Interrupt Source	Interrupt Description
83	Reserved	Reserved
84	GPU2D	General Interrupt Request
85	GPU2D	Interrupt Request for Busy (for S/W power gating feasibility)
86	UART5	UART5 ORed Interrupt Request
87	FEC	Fast Interrupt Request (OR of 13 interrupt sources)
88	OWIRE	1-Wire Interrupt Request
89	TIGERP PLATFORM	Debug Interrupt Request2, from Cross-Trigger Interface
90	SJC	SJC Interrupt Request
91	DCP	Interrupt Request for channels 1-3
92	DCP	Interrupt Request for channel 0
93	DCP	secure Interrupt Request for channels 0-3
94	PWM2	PWM Interrupt Request
95	Reserved	Reserved
96	Reserved	Reserved
97	RNGB	RNGB Interrupt Request
98	TIGERP PLATFORM	Debug Interrupt Request 3, from Cross-Trigger Interface
99	Reserved	Reserved
100	RAWNAND	BCH Interrupt Request
101	Reserved	Reserved
102	RAWNAND	GPMI Interrupt Request
103	GPIO5	Combined interrupt indication for GPIO5 signal 0 throughout 15
104	GPIO5	Combined interrupt indication for GPIO5 signal 16 throughout 31
105	GPIO6	Combined interrupt indication for GPIO6 signal 0 throughout 15
106	GPIO6	Combined interrupt indication for GPIO6 signal 16 throughout 31
107	Reserved	Reserved
108	Reserved	Reserved
109	Reserved	Reserved
110	APBHDMA	Interrupt Request for channel 0
111	APBHDMA	Interrupt Request for channel 1
112	APBHDMA	Interrupt Request for channel 2
113	APBHDMA	Interrupt Request for channel 3
114	APBHDMA	Interrupt Request for channel 4
115	APBHDMA	Interrupt Request for channel 5
116	APBHDMA	Interrupt Request for channel 6
117	APBHDMA	Interrupt Request for channel 7

Table continues on the next page...

Table 3-1. ARM Domain Interrupt Summary (continued)

IRQ	Interrupt Source	Interrupt Description
118-128	Reserved	Reserved

3.3 SDMA Event Mapping

Table 3-2 shows the DMA request signals for peripherals in the i.MX50.

Table 3-2. SDMA Event Mapping

Event Number	DMA Source	Description
0	Reserved	-
1	GPC	Will be used for power management.
2	UART4	UART4 Rx request
3	UART4	UART4 Tx request
4	Reserved	-
5	Reserved	-
6	eCSPI1	DMA Rx request
7	eCSPI1	DMA Tx request
8	eCSPI2	DMA Rx request
9	eCSPI2	DMA Tx request
10	I^2C3/esdhc3	I^2C3 request muxed with eSDHC3
11	esdhc4/cti2	eSDHC4 request muxed with CTI2 (SDMA_CTI) trigger_out[0] connected to SDMA event
12	UART2	UART2 Rx request
13	UART2	UART2 Tx request
14	IOMUX	External DMA request 0
15	IOMUX	External DMA request 1
16	UART5	UART5 Rx request
17	UART5	UART5 Tx request
18	UART1	UART1 Rx request
19	UART1	UART1 Tx request
20	I^2C1/esdhc1	I^2C1 muxed with eSDHC1
21	I^2C2/esdhc2	I^2C2 muxed with eSDHC2
22	SSI2	SSI2 Rx 2 DMA request
23	SSI2	SSI2 Tx 2 DMA request
24	SSI2	SSI2 Rx 1 DMA request

Table continues on the next page...

Table 3-2. SDMA Event Mapping (continued)

Event Number	DMA Source	Description
25	SSI2	SSI2 Tx 1 DMA request
26	SSI1	SSI1 Rx 2 DMA request
27	SSI1	SSI1 Tx 2 DMA request
28	SSI1	SSI1 Rx 1 DMA request
29	SSI1	SSI1 Tx 1 DMA request
30	Reserved	-
31	Reserved	-
32	Reserved	-
33	Reserved	-
34	Reserved	-
35	Reserved	-
36	Reserved	-
37	Reserved	-
38	CSPI	CSPI DMA Rx request
39	CSPI	CSPI DMA Tx request
40	Reserved	-
41	Reserved	-
42	UART3	UART3 Rx request
43	UART3	UART3 Tx request
44	Reserved	-
45	Reserved	-
46	Reserved	-
47	Reserved	-

As shown in the table, some of the events are an output of a mux of two signals or triggers. The select of this mux is controlled by general purpose register in IOMUXC.

For other shared connectivity peripherals that do not have dedicated DMA request signals, the AP interrupt service routines have the option to program the SDMA to move data between the peripheral and memory.

Chapter 4
External Signals and Pin Multiplexing

4.1 Introduction

The i.MX50 IC contains a limited number of pins, therefore, most pins have multiple signal options. The input-output multiplexer (IOMUX) controls the pin multiplexing. The IOMUX also configures other pin characteristics, such as voltage level, drive strength, and hysteresis.

4.2 External Signals

- Table 4-1 shows the chip's external signals.
- Table 4-2 shows an additional view of the muxing of external signals. It presents the muxing options per module/instance.
- Table 4-3 shows the IOMUX daisy chain.

Table 4-1. Pin Assignments Report

Pad Name	Mode	Instance	Signal	Pad Settings
KEY_COL0	ALT0	KPP	COL[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[0]	
	ALT2	rawnand	CLE	
	ALT6	ARM Platform	CTI_TRIGIN7	
	ALT7	USBPHY1	TXREADY	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
KEY_ROW0	ALT0	KPP	ROW[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[1]	
	ALT2	rawnand	ALE	
KEY_ROW0	ALT6	ARM Platform	CTI_TRIGIN_ACK7	
	ALT7	USBPHY1	RXVALID	
KEY_COL1	ALT0	KPP	COL[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[2]	
	ALT2	rawnand	CEN[0]	
	ALT6	ARM Platform	CTI_TRIGOUT_ACK6	
	ALT7	USBPHY1	RXACTIVE	
KEY_ROW1	ALT0	KPP	ROW[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
KEY_ROW1	ALT1	GPIO4	GPIO[3]	
	ALT2	rawnand	CEN[1]	
	ALT6	ARM Platform	CTI_TRIGOUT_ACK7	
	ALT7	USBPHY1	RXERROR	
KEY_COL2	ALT0	KPP	COL[2]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[4]	
	ALT2	rawnand	CEN[2]	
	ALT6	ARM Platform	CTI_TRIGOUT6	
	ALT7	USBPHY1	SIECLOCK	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
KEY_ROW2	ALT0	KPP	ROW[2]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[5]	
	ALT2	rawnand	CEN[3]	
	ALT6	ARM Platform	CTI_TRIGOUT7	
	ALT7	USBPHY1	LINESTATE[0]	
KEY_COL3	ALT0	KPP	COL[3]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[6]	
	ALT2	rawnand	READY0	
KEY_COL3	ALT6	SDMA	SDMA_EXT_EVENT[0]	
	ALT7	USBPHY1	LINESTATE[1]	
KEY_ROW3	ALT0	KPP	ROW[3]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[7]	
	ALT2	rawnand	DQS	
	ALT6	SDMA	SDMA_EXT_EVENT[1]	
	ALT7	USBPHY1	VBUSVALID	
I2C1_SCL	ALT0	I2C1	SCL	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Enabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
I2C1_SCL	ALT1	GPIO6	GPIO[18]	
	ALT2	UART2	TXD_MUX	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
I2C1_SDA	ALT0	I2C1	SDA	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Enabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[19]	
	ALT2	UART2	RXD_MUX	
I2C2_SCL	ALT0	I2C2	SCL	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Enabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[20]	
	ALT2	UART2	CTS	
I2C2_SDA	ALT0	I2C2	SDA	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Enabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[21]	
	ALT2	UART2	RTS	
I2C3_SCL	ALT0	I2C3	SCL	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Enabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[22]	
	ALT2	FEC	MDC	
I2C3_SCL	ALT3	GPC	PMIC_RDY	
	ALT5	GPT	CAPIN1	
	ALT6	observe_mux	OBSRV_INT_OUT0	
	ALT7	USB1	USBOTG_OC	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
I2C3_SDA	ALT0	I2C3	SDA	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Enabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[23]	
	ALT2	FEC	MDIO	
	ALT3	TZIC	PWRFAIL_INT	
	ALT4	SRTC	ALARM_DEB	
	ALT5	GPT	CAPIN2	
	ALT6	observe_mux	OBSRV_INT_OUT1	
	ALT7	USB1	USBOTG_PWR	
PWM1	ALT0	PWM1	PWMO	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Disabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[24]	
	ALT2	USB1	USBOTG_OC	
	ALT5	GPT	CMPOUT1	
	ALT6	observe_mux	OBSRV_INT_OUT2	
	ALT7	SJC	FAIL	
PWM2	ALT0	PWM2	PWMO	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Disabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[25]	
	ALT2	USB1	USBOTG_PWR	
	ALT5	GPT	CMPOUT2	
	ALT6	observe_mux	OBSRV_INT_OUT3	
	ALT7	SRC	ANY_PU_RST	
OWIRE	ALT0	owire	LINE	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Disabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[26]	
	ALT2	USB1	USBH1_OC	
	ALT3	CCM	SSI_EXT1_CLK	
	ALT4	EPDC	PWRIRQ	
	ALT5	GPT	CMPOUT3	
	ALT6	observe_mux	OBSRV_INT_OUT4	
	ALT7	SJC	JTAG_ACT	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPITO	ALT0	EPIT1	EPITO	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Disabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[27]	
	ALT2	USB1	USBH1_PWR	
	ALT3	CCM	SSI_EXT2_CLK	
	ALT4	DPLLC1	TOG_EN	
	ALT5	GPT	CLKIN	
	ALT6	ARM Platform	PMU_IRQ_B	
	ALT7	SJC	DE_B	
WDOG	ALT0	WDOG1	WDOG_B	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Disabled Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[28]	
	ALT2	WDOG1	WDOG_RST_B_DEB	
	ALT6	CCM	XTAL32K	
	ALT7	SJC	DONE	
SSI_TXFS	ALT0	AUDMUX	AUD3_TXFS	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[0]	
	ALT6	SRC	BT_FUSE_RSV[1]	
SSI_TXFS	ALT7	USBPHY1	DATAOUT[8]	
SSI_TXC	ALT0	AUDMUX	AUD3_TXC	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[1]	
	ALT6	SRC	BT_FUSE_RSV[0]	
	ALT7	USBPHY1	DATAOUT[9]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SSI_TXD	ALT0	AUDMUX	AUD3_TXD	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[2]	
	ALT4	CSPI	RDY	
SSI_TXD	ALT7	USBPHY1	DATAOUT[10]	
SSI_RXD	ALT0	AUDMUX	AUD3_RXD	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[3]	
	ALT4	CSPI	SS3	
	ALT7	USBPHY1	DATAOUT[11]	
SSI_RXFS	ALT0	AUDMUX	AUD3_RXFS	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[4]	
	ALT2	UART5	TXD_MUX	
SSI_RXFS	ALT3	EIM	D[6]	
	ALT4	CSPI	SS2	
	ALT5	FEC	COL	
	ALT6	FEC	MDC	
	ALT7	USBPHY1	DATAOUT[12]	
SSI_RXC	ALT0	AUDMUX	AUD3_RXC	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[5]	
	ALT2	UART5	RXD_MUX	
	ALT3	EIM	D[7]	
	ALT4	CSPI	SS1	
	ALT5	FEC	RX_CLK	
	ALT6	FEC	MDIO	
	ALT7	USBPHY1	DATAOUT[13]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
UART1_TXD	ALT0	UART1	TXD_MUX	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[6]	
	ALT7	USBPHY1	DATAOUT[14]	
UART1_RXD	ALT0	UART1	RXD_MUX	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[7]	
	ALT7	USBPHY1	DATAOUT[15]	
UART1_CTS	ALT0	UART1	CTS	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[8]	
	ALT2	UART5	TXD_MUX	
	ALT4	ESDHC4	DAT4	
	ALT5	ESDHC4	CMD	
	ALT7	USBPHY2	DATAOUT[8]	
UART1_RTS	ALT0	UART1	RTS	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[9]	
UART1_RTS	ALT2	UART5	RXD_MUX	
	ALT4	ESDHC4	DAT5	
	ALT5	ESDHC4	CLK	
	ALT7	USBPHY2	DATAOUT[9]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
UART2_TXD	ALT0	UART2	TXD_MUX	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[10]	
	ALT4	ESDHC4	DAT6	
	ALT5	ESDHC4	DAT4	
	ALT7	USBPHY2	DATAOUT[10]	
UART2_RXD	ALT0	UART2	RXD_MUX	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[11]	
	ALT4	ESDHC4	DAT7	
	ALT5	ESDHC4	DAT5	
	ALT7	USBPHY2	DATAOUT[11]	
UART2_CTS	ALT0	UART2	CTS	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[12]	
	ALT4	ESDHC4	CMD	
UART2_CTS	ALT5	ESDHC4	DAT6	
	ALT7	USBPHY2	DATAOUT[12]	
UART2_RTS	ALT0	UART2	RTS	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[13]	
	ALT4	ESDHC4	CLK	
	ALT5	ESDHC4	DAT7	
	ALT7	USBPHY2	DATAOUT[13]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
UART3_TXD	ALT0	UART3	TXD_MUX	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
UART3_TXD	ALT1	GPIO6	GPIO[14]	
	ALT3	ESDHC1	DAT4	
	ALT4	ESDHC4	DAT0	
	ALT5	ESDHC2	WP	
	ALT6	EIM	D[12]	
	ALT7	USBPHY2	DATAOUT[14]	
UART3_RXD	ALT0	UART3	RXD_MUX	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[15]	
	ALT3	ESDHC1	DAT5	
	ALT4	ESDHC4	DAT1	
	ALT5	ESDHC2	CD	
	ALT6	EIM	D[13]	
	ALT7	USBPHY2	DATAOUT[15]	
UART4_TXD	ALT0	UART4	TXD_MUX	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO6	GPIO[16]	
	ALT2	UART3	CTS	
	ALT3	ESDHC1	DAT6	
	ALT4	ESDHC4	DAT2	
	ALT5	ESDHC2	LCTL	
	ALT6	EIM	D[14]	
UART4_RXD	ALT0	UART4	RXD_MUX	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
UART4_RXD	ALT1	GPIO6	GPIO[17]	
	ALT2	UART3	RTS	
	ALT3	ESDHC1	DAT7	
	ALT4	ESDHC4	DAT3	
	ALT5	ESDHC1	LCTL	
	ALT6	EIM	D[15]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
CSPI_SCLK	ALT0	CSPI	SCLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[8]	
CSPI_MOSI	ALT0	CSPI	MOSI	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[9]	
CSPI_MISO	ALT0	CSPI	MISO	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[10]	
CSPI_SS0	ALT0	CSPI	SS0	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[11]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
ECSPI1_SCLK	ALT0	ECSPI1	SCLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[12]	
	ALT2	CSPI	RDY	
	ALT3	ECSPI2	RDY	
	ALT4	UART3	RTS	
	ALT5	EPDC	SDCE[6]	
ECSPI1_SCLK	ALT7	EIM	D[8]	
ECSPI1_MOSI	ALT0	ECSPI1	MOSI	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[13]	
	ALT2	CSPI	SS1	
	ALT3	ECSPI2	SS1	
	ALT4	UART3	CTS	
	ALT5	EPDC	SDCE[7]	
	ALT7	EIM	D[9]	
ECSPI1_MISO	ALT0	ECSPI1	MISO	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[14]	
	ALT2	CSPI	SS2	
	ALT3	ECSPI2	SS2	
	ALT4	UART4	RTS	
	ALT5	EPDC	SDCE[8]	
	ALT7	EIM	D[10]	
ECSPI1_SS0	ALT0	ECSPI1	SS0	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
ECSPI1_SS0	ALT1	GPIO4	GPIO[15]	
	ALT2	CSPI	SS3	
	ALT3	ECSPI2	SS3	
	ALT4	UART4	CTS	
	ALT5	EPDC	SDCE[9]	
	ALT7	EIM	D[11]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
ECSPI2_SCLK	ALT0	ECSPI2	SCLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[16]	
	ALT2	ELCDIF	WR_RWN	
	ALT3	ECSPI1	RDY	
	ALT4	UART5	RTS	
	ALT5	ELCDIF	DOTCLK	
	ALT6	rawnand	CEN[4]	
	ALT7	EIM	D[8]	
ECSPI2_MOSI	ALT0	ECSPI2	MOSI	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[17]	
	ALT2	ELCDIF	RD_E	
	ALT3	ECSPI1	SS1	
	ALT4	UART5	CTS	
	ALT5	ELCDIF	ENABLE	
	ALT6	rawnand	CEN[5]	
	ALT7	EIM	D[9]	
ECSPI2_MISO	ALT0	ECSPI2	MISO	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[18]	
	ALT2	ELCDIF	RS	
	ALT3	ECSPI1	SS2	
	ALT4	UART5	TXD_MUX	
	ALT5	ELCDIF	VSYNC	
	ALT6	rawnand	CEN[6]	
	ALT7	EIM	D[10]	
ECSPI2_SS0	ALT0	ECSPI2	SS0	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[19]	
	ALT2	ELCDIF	CS	
	ALT3	ECSPI1	SS3	
	ALT4	UART5	RXD_MUX	
	ALT5	ELCDIF	HSYNC	
	ALT6	rawnand	CEN[7]	
	ALT7	EIM	D[11]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SD1_CLK	ALT0	ESDHC1	CLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[0]	
	ALT7	CCM	CLKO	
SD1_CMD	ALT0	ESDHC1	CMD	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[1]	
	ALT7	CCM	CLKO2	
SD1_D0	ALT0	ESDHC1	DAT0	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[2]	
	ALT7	CCM	PLL1_BYP	
SD1_D1	ALT0	ESDHC1	DAT1	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[3]	
	ALT7	CCM	PLL2_BYP	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SD1_D2	ALT0	ESDHC1	DAT2	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[4]	
	ALT7	CCM	PLL3_BYP	
SD1_D3	ALT0	ESDHC1	DAT3	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[5]	
SD2_CLK	ALT0	ESDHC2	CLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[6]	
	ALT2	mshc	SCLK	
SD2_CMD	ALT0	ESDHC2	CMD	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[7]	
	ALT2	mshc	BS	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SD2_D0	ALT0	ESDHC2	DAT0	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[8]	
	ALT2	mshc	DATA[0]	
	ALT3	KPP	COL[4]	
SD2_D1	ALT0	ESDHC2	DAT1	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[9]	
	ALT2	mshc	DATA[1]	
	ALT3	KPP	ROW[4]	
SD2_D2	ALT0	ESDHC2	DAT2	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[10]	
	ALT2	mshc	DATA[2]	
	ALT3	KPP	COL[5]	
SD2_D3	ALT0	ESDHC2	DAT3	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[11]	
	ALT2	mshc	DATA[3]	
	ALT3	KPP	ROW[5]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SD2_D4	ALT0	ESDHC2	DAT4	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[12]	
	ALT2	AUDMUX	AUD4_RXFS	
	ALT3	KPP	COL[6]	
	ALT4	EIM	D[0]	
	ALT7	CCM	CCM_OUT_0	
SD2_D5	ALT0	ESDHC2	DAT5	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[13]	
SD2_D5	ALT2	AUDMUX	AUD4_RXC	
	ALT3	KPP	ROW[6]	
	ALT4	EIM	D[1]	
	ALT7	CCM	CCM_OUT_1	
SD2_D6	ALT0	ESDHC2	DAT6	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[14]	
	ALT2	AUDMUX	AUD4_RXD	
	ALT3	KPP	COL[7]	
	ALT4	EIM	D[2]	
	ALT7	CCM	CCM_OUT_2	
SD2_D7	ALT0	ESDHC2	DAT7	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[15]	
	ALT2	AUDMUX	AUD4_TXFS	
	ALT3	KPP	ROW[7]	
	ALT4	EIM	D[3]	
	ALT7	CCM	STOP	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SD2_WP	ALT0	ESDHC2	WP	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[16]	
SD2_WP	ALT2	AUDMUX	AUD4_TXD	
	ALT4	EIM	D[4]	
	ALT7	CCM	WAIT	
SD2_CD	ALT0	ESDHC2	CD	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[17]	
	ALT2	AUDMUX	AUD4_TXC	
	ALT4	EIM	D[5]	
	ALT7	CCM	REF_EN_B	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
PMIC_ON_REQ	No Muxing (ALT0)	SRTC	PMIC_ON_REQ	Drive Strength - High Hyst. Enable - NA Pull / Keep Select - NA Pull Up / Down Config. - NA strength mode - 4_LEVEL dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - Disabled Slew Rate - CFG(FAST) test_ts - Disabled
PMIC_STBY_REQ		CCM	PMIC_VSTBY_REQ	Drive Strength - CFG(High) Hyst. Enable - NA Pull / Keep Select - NA Pull Up / Down Config. - NA strength mode - 4_LEVEL dse test - regular Open Drain Enable - Disabled Pull / Keep Enable - Disabled Slew Rate - CFG(FAST) test_ts - Disabled
POR_B	No Muxing (ALT0)	SRC	POR_B	Drive Strength - NA Hyst. Enable - CFG(Enabled) Pull / Keep Select - Pull Pull Up / Down Config. - 100KOhm PU strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - CFG(Enabled) Slew Rate - NA test_ts - Disabled
BOOT_MODE1		SRC	BOOT_MODE[1]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
RESET_IN_B	No Muxing (ALT0)	SRC	RESET_B	Drive Strength - NA Hyst. Enable - CFG(Enabled) Pull / Keep Select - Pull Pull Up / Down Config. - 100KOhm PU strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - Enabled Slew Rate - NA test_ts - Disabled
BOOT_MODE0		SRC	BOOT_MODE[0]	Drive Strength - NA Hyst. Enable - CFG(Enabled) Pull / Keep Select - Pull Pull Up / Down Config. - 100KOhm PU strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - CFG(Enabled) Slew Rate - NA test_ts - Disabled

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
TEST_MODE	No Muxing (ALT0)	tcu	TEST_MODE	Drive Strength - NA Hyst. Enable - CFG(Disabled) Pull / Keep Select - Pull Pull Up / Down Config. - 100KOhm PD strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - CFG(Enabled) Slew Rate - NA test_ts - Disabled
JTAG_TMS		SJC	TMS	Drive Strength - NA Hyst. Enable - CFG(Enabled) Pull / Keep Select - Pull Pull Up / Down Config. - 47KOhm PU strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - CFG(Enabled) Slew Rate - NA test_ts - Disabled

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
JTAG_MOD	No Muxing (ALT0)	SJC	MOD	Drive Strength - NA Hyst. Enable - CFG(Disabled) Pull / Keep Select - Pull Pull Up / Down Config. - 100KOhm PU strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - CFG(Enabled) Slew Rate - NA test_ts - Disabled
JTAG_TRSTB		SJC	TRSTB	Drive Strength - NA Hyst. Enable - CFG(Disabled) Pull / Keep Select - Pull Pull Up / Down Config. - 47KOhm PU strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - CFG(Enabled) Slew Rate - NA test_ts - Disabled

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
JTAG_TDI	No Muxing (ALT0)	SJC	TDI	Drive Strength - NA Hyst. Enable - CFG(Enabled) Pull / Keep Select - Pull Pull Up / Down Config. - 47KOhm PU strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - CFG(Enabled) Slew Rate - NA test_ts - Disabled
JTAG_TCK		SJC	TCK	Drive Strength - NA Hyst. Enable - CFG(Enabled) Pull / Keep Select - Pull Pull Up / Down Config. - 100KOhm PD strength mode - 4_LEVEL dse test - regular Open Drain Enable - NA Pull / Keep Enable - CFG(Enabled) Slew Rate - NA test_ts - Disabled
JTAG_TDO	No Muxing (ALT0)	SJC	TDO	Drive Strength - CFG(High) Hyst. Enable - CFG(Disabled) Pull / Keep Select - Keep Pull Up / Down Config. - NA strength mode - 4_LEVEL dse test - regular Open Drain Enable - Disabled Pull / Keep Enable - CFG(Enabled) Slew Rate - CFG(FAST) test_ts - Disabled

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DISP_D0	ALT0	ELCDIF	DAT[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[0]	
	ALT2	FEC	TX_CLK	
	ALT3	EIM	A[16]	
	ALT6	SDMA	DEBUG_PC[0]	
	ALT7	USBPHY1	VSTATUS[0]	
DISP_D1	ALT0	ELCDIF	DAT[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[1]	
	ALT2	FEC	RX_ER	
	ALT3	EIM	A[17]	
	ALT6	SDMA	DEBUG_PC[1]	
	ALT7	USBPHY1	VSTATUS[1]	
DISP_D2	ALT0	ELCDIF	DAT[2]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[2]	
DISP_D2	ALT2	FEC	RX_DV	
	ALT3	EIM	A[18]	
	ALT6	SDMA	DEBUG_PC[2]	
	ALT7	USBPHY1	VSTATUS[2]	
DISP_D3	ALT0	ELCDIF	DAT[3]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[3]	
	ALT2	FEC	RDATA[1]	
	ALT3	EIM	A[19]	
	ALT4	FEC	COL	
	ALT6	SDMA	DEBUG_PC[3]	
	ALT7	USBPHY1	VSTATUS[3]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DISP_D4	ALT0	ELCDIF	DAT[4]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[4]	
	ALT2	FEC	RDATA[0]	
	ALT3	EIM	A[20]	
	ALT6	SDMA	DEBUG_PC[4]	
	ALT7	USBPHY1	VSTATUS[4]	
DISP_D5	ALT0	ELCDIF	DAT[5]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[5]	
DISP_D5	ALT2	FEC	TX_EN	
	ALT3	EIM	A[21]	
	ALT6	SDMA	DEBUG_PC[5]	
	ALT7	USBPHY1	VSTATUS[5]	
DISP_D6	ALT0	ELCDIF	DAT[6]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[6]	
	ALT2	FEC	TDATA[1]	
	ALT3	EIM	A[22]	
	ALT4	FEC	RX_CLK	
	ALT6	SDMA	DEBUG_PC[6]	
	ALT7	USBPHY1	VSTATUS[6]	
DISP_D7	ALT0	ELCDIF	DAT[7]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[7]	
	ALT2	FEC	TDATA[0]	
	ALT3	EIM	A[23]	
	ALT6	SDMA	DEBUG_PC[7]	
	ALT7	USBPHY1	VSTATUS[7]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DISP_WR	ALT0	ELCDIF	WR_RWN	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[16]	
DISP_WR	ALT2	ELCDIF	DOTCLK	
	ALT3	EIM	A[24]	
	ALT6	SDMA	DEBUG_PC[8]	
	ALT7	USBPHY1	AVALID	
DISP_RD	ALT0	ELCDIF	RD_E	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[19]	
	ALT2	ELCDIF	ENABLE	
	ALT3	EIM	A[25]	
	ALT6	SDMA	DEBUG_PC[9]	
	ALT7	USBPHY1	BVALID	
DISP_RS	ALT0	ELCDIF	RS	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[17]	
	ALT2	ELCDIF	VSYNC	
	ALT3	EIM	A[26]	
	ALT6	SDMA	DEBUG_PC[10]	
	ALT7	USBPHY1	ENDSESSION	
DISP_CS	ALT0	ELCDIF	CS	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[21]	
	ALT2	ELCDIF	HSYNC	
	ALT3	EIM	A[27]	
	ALT4	EIM	CS[3]	
	ALT6	SDMA	DEBUG_PC[11]	
	ALT7	USBPHY1	IDDIG	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DISP_BUSY	ALT0	ELCDIF	HSYNC	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[18]	
	ALT4	EIM	CS[3]	
	ALT6	SDMA	DEBUG_PC[12]	
	ALT7	USBPHY1	HOSTDISCONNECT	
DISP_RESET	ALT0	ELCDIF	RESET	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[20]	
	ALT4	EIM	CS[3]	
	ALT6	SDMA	DEBUG_PC[13]	
	ALT7	USBPHY1	BISTOK	
SD3_CMD	ALT0	ESDHC3	CMD	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[18]	
	ALT2	rawnand	WRN	
SD3_CLK	ALT0	ESDHC3	CLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[19]	
	ALT2	rawnand	RDN	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SD3_D0	ALT0	ESDHC3	DAT0	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[20]	
	ALT2	rawnand	D[4]	
SD3_D1	ALT0	ESDHC3	DAT1	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[21]	
	ALT2	rawnand	D[5]	
	ALT7	CCM	PLL2_BYP	
SD3_D2	ALT0	ESDHC3	DAT2	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[22]	
	ALT2	rawnand	D[6]	
	ALT7	CCM	PLL3_BYP	
SD3_D3	ALT0	ESDHC3	DAT3	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[23]	
	ALT2	rawnand	D[7]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SD3_D4	ALT0	ESDHC3	DAT4	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[24]	
	ALT2	rawnand	D[0]	
SD3_D5	ALT0	ESDHC3	DAT5	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[25]	
	ALT2	rawnand	D[1]	
SD3_D6	ALT0	ESDHC3	DAT6	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[26]	
	ALT2	rawnand	D[2]	
SD3_D7	ALT0	ESDHC3	DAT7	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[27]	
	ALT2	rawnand	D[3]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
SD3_WP	ALT0	ESDHC3	WP	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO5	GPIO[28]	
	ALT2	rawnand	RESETN	
	ALT4	ESDHC4	LCTL	
	ALT5	EIM	CS[3]	
DISP_D8	ALT0	ELCDIF	DAT[8]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[8]	
DISP_D8	ALT2	rawnand	CLE	
	ALT3	ESDHC1	LCTL	
	ALT4	ESDHC4	CMD	
	ALT5	KPP	COL[4]	
	ALT6	FEC	TX_CLK	
	ALT7	USBPHY1	DATAOUT[0]	
DISP_D9	ALT0	ELCDIF	DAT[9]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[9]	
	ALT2	rawnand	ALE	
	ALT3	ESDHC2	LCTL	
	ALT4	ESDHC4	CLK	
	ALT5	KPP	ROW[4]	
	ALT6	FEC	RX_ER	
	ALT7	USBPHY1	DATAOUT[1]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DISP_D10	ALT0	ELCDIF	DAT[10]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[10]	
	ALT2	rawnand	CEN[0]	
	ALT3	ESDHC3	LCTL	
	ALT4	ESDHC4	DAT0	
	ALT5	KPP	COL[5]	
	ALT6	FEC	RX_DV	
	ALT7	USBPHY1	DATAOUT[2]	
DISP_D11	ALT0	ELCDIF	DAT[11]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[11]	
	ALT2	rawnand	CEN[1]	
	ALT4	ESDHC4	DAT1	
	ALT5	KPP	ROW[5]	
	ALT6	FEC	RDATA[1]	
	ALT7	USBPHY1	DATAOUT[3]	
DISP_D12	ALT0	ELCDIF	DAT[12]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[12]	
	ALT2	rawnand	CEN[2]	
	ALT3	ESDHC1	CD	
	ALT4	ESDHC4	DAT2	
	ALT5	KPP	COL[6]	
	ALT6	FEC	RDATA[0]	
	ALT7	USBPHY1	DATAOUT[4]	
DISP_D13	ALT0	ELCDIF	DAT[13]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[13]	
	ALT2	rawnand	CEN[3]	
	ALT3	ESDHC3	CD	
	ALT4	ESDHC4	DAT3	
	ALT5	KPP	ROW[6]	
	ALT6	FEC	TX_EN	
	ALT7	USBPHY1	DATAOUT[5]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DISP_D14	ALT0	ELCDIF	DAT[14]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[14]	
	ALT2	rawnand	READY0	
	ALT3	ESDHC1	WP	
	ALT4	ESDHC4	WP	
	ALT5	kpp	COL[7]	
	ALT6	FEC	TDATA[1]	
	ALT7	USBPHY1	DATAOUT[6]	
DISP_D15	ALT0	ELCDIF	DAT[15]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO2	GPIO[15]	
	ALT2	rawnand	DQS	
	ALT3	ESDHC3	RST	
	ALT4	ESDHC4	CD	
	ALT5	kpp	ROW[7]	
	ALT6	FEC	TDATA[0]	
	ALT7	USBPHY1	DATAOUT[7]	
DRAM_CAS	No Muxing (ALT0)	DRAM MC	CASN	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_RAS		DRAM MC	RASN	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_SDBA2	No Muxing (ALT0)	DRAM MC	BA[2]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_SDBA1		DRAM MC	BA[1]	
DRAM_SDBA0	No Muxing (ALT0)	DRAM MC	BA[0]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_SDWE		DRAM MC	WEN	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_OPEN	No Muxing (ALT0)	DRAM MC	OPEN	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_OPENFB		DRAM MC	OPENFB	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_A14	No Muxing (ALT0)	DRAM MC	A[14]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_A13		DRAM MC	A[13]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_A12	No Muxing (ALT0)	DRAM MC	A[12]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_A11		DRAM MC	A[11]	
DRAM_A10	No Muxing (ALT0)	DRAM MC	A[10]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_A9		DRAM MC	A[9]	
DRAM_A8	No Muxing (ALT0)	DRAM MC	A[8]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_A7		DRAM MC	A[7]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_A6	No Muxing (ALT0)	DRAM MC	A[6]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_A5		DRAM MC	A[5]	
DRAM_SDCLK_1_B	No Muxing (ALT0)	DRAM MC	CLKN[1]	Drive Strength - NA Hyst. Enable - NA receiver_type - NA Pull / Keep Select - NA Pull Up / Down Config. - NA DDR / CMOS Input Mode - NA ddr_sel - NA Pull / Keep Enable - NA do_trim - NA test_ts - NA On Die Termination - NA
DRAM_SDCLK_1	No Muxing (ALT0)	DRAM MC	CLK[1]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - DDR2 ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - NA

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_SDCLK_0_B	No Muxing (ALT0)	DRAM MC	CLKN[0]	Drive Strength - NA Hyst. Enable - NA receiver_type - NA Pull / Keep Select - NA Pull Up / Down Config. - NA DDR / CMOS Input Mode - NA ddr_sel - NA Pull / Keep Enable - NA do_trim - NA test_ts - NA On Die Termination - NA
DRAM_SDCLK_0	No Muxing (ALT0)	DRAM MC	CLK[0]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - DDR2 ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - NA
DRAM_CS0	No Muxing (ALT0)	DRAM MC	CSN[0]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_CS1		DRAM MC	CSN[1]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_SDCKE	No Muxing (ALT0)	DRAM MC	CKE	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_SDODT0		DRAM MC	ODT[0]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - NA
DRAM_A0	No Muxing (ALT0)	DRAM MC	A[0]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_A1		DRAM MC	A[1]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_A2	No Muxing (ALT0)	DRAM MC	A[2]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_A3		DRAM MC	A[3]	
DRAM_A4	No Muxing (ALT0)	DRAM MC	A[4]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - NA test_ts - Disabled On Die Termination - NA
DRAM_D16		DRAM MC	D[16]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_D17	No Muxing (ALT0)	DRAM MC	D[17]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D18		DRAM MC	D[18]	
DRAM_D19	No Muxing (ALT0)	DRAM MC	D[19]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D20		DRAM MC	D[20]	
DRAM_D21	No Muxing (ALT0)	DRAM MC	D[21]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D22		DRAM MC	D[22]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_D23	No Muxing (ALT0)	DRAM MC	D[23]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_DQM2		DRAM MC	DQM[2]	Drive Strength - CFG(OI_240) Hyst. Enable - NA receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_SDQS2	No Muxing (ALT0)	DRAM MC	DQS[2]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(DDR2) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Disabled) do_trim - NA test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_SDQS2_B		DRAM MC	DQSN[2]	Drive Strength - NA Hyst. Enable - NA receiver_type - NA Pull / Keep Select - NA Pull Up / Down Config. - NA DDR / CMOS Input Mode - NA ddr_sel - NA Pull / Keep Enable - NA do_trim - NA test_ts - NA On Die Termination - NA
DRAM_D0	No Muxing (ALT0)	DRAM MC	D[0]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D1		DRAM MC	D[1]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_D2	No Muxing (ALT0)	DRAM MC	D[2]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D3		DRAM MC	D[3]	
DRAM_D4	No Muxing (ALT0)	DRAM MC	D[4]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D5		DRAM MC	D[5]	
DRAM_D6	No Muxing (ALT0)	DRAM MC	D[6]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D7		DRAM MC	D[7]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_DQM0	No Muxing (ALT0)	DRAM MC	DQM[0]	Drive Strength - CFG(OI_240) Hyst. Enable - NA receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_SDQS0		DRAM MC	DQS[0]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(DDR2) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Disabled) do_trim - NA test_ts - Disabled On Die Termination - CFG(DISABLE)

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_SDQS0_B	No Muxing (ALT0)	DRAM MC	DQSN[0]	Drive Strength - NA Hyst. Enable - NA receiver_type - NA Pull / Keep Select - NA Pull Up / Down Config. - NA DDR / CMOS Input Mode - NA ddr_sel - NA Pull / Keep Enable - NA do_trim - NA test_ts - NA On Die Termination - NA
DRAM_SDODT1		DRAM MC	ODT[1]	Drive Strength - CFG(OI_240) Hyst. Enable - Disabled receiver_type - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - NA

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_SDQS1_B	No Muxing (ALT0)	DRAM MC	DQSN[1]	Drive Strength - NA Hyst. Enable - NA receiver_type - NA Pull / Keep Select - NA Pull Up / Down Config. - NA DDR / CMOS Input Mode - NA ddr_sel - NA Pull / Keep Enable - NA do_trim - NA test_ts - NA On Die Termination - NA
DRAM_SDQS1		DRAM MC	DQS[1]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(DDR2) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Disabled) do_trim - NA test_ts - Disabled On Die Termination - CFG(DISABLE)

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_DQM1	No Muxing (ALT0)	DRAM MC	DQM[1]	Drive Strength - CFG(OI_240) Hyst. Enable - NA receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D8		DRAM MC	D[8]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D9	No Muxing (ALT0)	DRAM MC	D[9]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D10		DRAM MC	D[10]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_D11	No Muxing (ALT0)	DRAM MC	D[11]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D12		DRAM MC	D[12]	
DRAM_D13	No Muxing (ALT0)	DRAM MC	D[13]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D14		DRAM MC	D[14]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_D15	No Muxing (ALT0)	DRAM MC	D[15]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_SDQS3_B		DRAM MC	DQSN[3]	Drive Strength - NA Hyst. Enable - NA receiver_type - NA Pull / Keep Select - NA Pull Up / Down Config. - NA DDR / CMOS Input Mode - NA ddr_sel - NA Pull / Keep Enable - NA do_trim - NA test_ts - NA On Die Termination - NA

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_SDQS3	No Muxing (ALT0)	DRAM MC	DQS[3]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(DDR2) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Disabled) do_trim - NA test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_DQM3		DRAM MC	DQM[3]	Drive Strength - CFG(OI_240) Hyst. Enable - NA receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CMOS ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D24	No Muxing (ALT0)	DRAM MC	D[24]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D25		DRAM MC	D[25]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
DRAM_D26	No Muxing (ALT0)	DRAM MC	D[26]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D27		DRAM MC	D[27]	
DRAM_D28	No Muxing (ALT0)	DRAM MC	D[28]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D29		DRAM MC	D[29]	
DRAM_D30	No Muxing (ALT0)	DRAM MC	D[30]	Drive Strength - CFG(OI_240) Hyst. Enable - CFG(Disabled) receiver_type - CFG(default) Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - NA DDR / CMOS Input Mode - CFG(CMOS) ddr_sel - CFG(LPDDR2) Pull / Keep Enable - CFG(Enabled) do_trim - CFG(MIN) test_ts - Disabled On Die Termination - CFG(DISABLE)
DRAM_D31		DRAM MC	D[31]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_D0	ALT0	EPDC	SDDO[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[0]	
	ALT2	EIM	D[0]	
	ALT3	ELCDIF	RS	
	ALT4	ELCDIF	DOTCLK	
	ALT6	SDMA	DEBUG_EVT_CHN_LINES[0]	
	ALT7	USBPHY2	DATAOUT[0]	
EPDC_D1	ALT0	EPDC	SDDO[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EPDC_D1	ALT1	GPIO3	GPIO[1]	
	ALT2	EIM	D[1]	
	ALT3	ELCDIF	CS	
	ALT4	ELCDIF	ENABLE	
	ALT6	SDMA	DEBUG_EVT_CHN_LINES[1]	
	ALT7	USBPHY2	DATAOUT[1]	
EPDC_D2	ALT0	EPDC	SDDO[2]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[2]	
	ALT2	EIM	D[2]	
	ALT3	ELCDIF	WR_RWN	
	ALT4	ELCDIF	VSYNC	
	ALT6	SDMA	DEBUG_EVT_CHN_LINES[2]	
	ALT7	USBPHY2	DATAOUT[2]	
EPDC_D3	ALT0	EPDC	SDDO[3]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[3]	
	ALT2	EIM	D[3]	
	ALT3	ELCDIF	RD_E	
	ALT4	ELCDIF	HSYNC	
	ALT6	SDMA	DEBUG_EVT_CHN_LINES[3]	
	ALT7	USBPHY2	DATAOUT[3]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_D4	ALT0	EPDC	SDDO[4]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[4]	
	ALT2	EIM	D[4]	
	ALT6	SDMA	DEBUG_EVT_CHN_LINES[4]	
	ALT7	USBPHY2	DATAOUT[4]	
EPDC_D5	ALT0	EPDC	SDDO[5]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[5]	
	ALT2	EIM	D[5]	
	ALT6	SDMA	DEBUG_EVT_CHN_LINES[5]	
	ALT7	USBPHY2	DATAOUT[5]	
EPDC_D6	ALT0	EPDC	SDDO[6]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[6]	
	ALT2	EIM	D[6]	
	ALT6	SDMA	DEBUG_EVT_CHN_LINES[6]	
	ALT7	USBPHY2	DATAOUT[6]	
EPDC_D7	ALT0	EPDC	SDDO[7]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[7]	
	ALT2	EIM	D[7]	
EPDC_D7	ALT6	SDMA	DEBUG_EVT_CHN_LINES[7]	
	ALT7	USBPHY2	DATAOUT[7]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_D8	ALT0	EPDC	SDDO[8]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[8]	
	ALT2	EIM	D[8]	
	ALT3	ELCDIF	DAT[24]	
	ALT6	SDMA	DEBUG_MATCHED_DMBUS	
	ALT7	USBPHY2	VSTATUS[0]	
EPDC_D9	ALT0	EPDC	SDDO[9]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[9]	
	ALT2	EIM	D[9]	
	ALT3	ELCDIF	DAT[25]	
	ALT6	SDMA	DEBUG_EVENT_CHANNEL_SEL	
	ALT7	USBPHY2	VSTATUS[1]	
EPDC_D10	ALT0	EPDC	SDDO[10]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[10]	
EPDC_D10	ALT2	EIM	D[10]	
	ALT3	ELCDIF	DAT[26]	
	ALT6	SDMA	DEBUG_EVENT_CHANNEL[0]	
	ALT7	USBPHY2	VSTATUS[2]	
EPDC_D11	ALT0	EPDC	SDDO[11]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[11]	
	ALT2	EIM	D[11]	
	ALT3	ELCDIF	DAT[27]	
	ALT6	SDMA	DEBUG_EVENT_CHANNEL[1]	
	ALT7	USBPHY2	VSTATUS[3]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_D12	ALT0	EPDC	SDDO[12]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[12]	
	ALT2	EIM	D[12]	
	ALT3	ELCDIF	DAT[28]	
	ALT6	SDMA	DEBUG_EVENT_CHANNEL[2]	
	ALT7	USBPHY2	VSTATUS[4]	
EPDC_D13	ALT0	EPDC	SDDO[13]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[13]	
EPDC_D13	ALT2	EIM	D[13]	
	ALT3	ELCDIF	DAT[29]	
	ALT6	SDMA	DEBUG_EVENT_CHANNEL[3]	
	ALT7	USBPHY2	VSTATUS[5]	
EPDC_D14	ALT0	EPDC	SDDO[14]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[14]	
	ALT2	EIM	D[14]	
	ALT3	ELCDIF	DAT[30]	
	ALT4	AUDMUX	AUD6_TXD	
	ALT6	SDMA	DEBUG_EVENT_CHANNEL[4]	
	ALT7	USBPHY2	VSTATUS[6]	
EPDC_D15	ALT0	EPDC	SDDO[15]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[15]	
	ALT2	EIM	D[15]	
	ALT3	ELCDIF	DAT[31]	
	ALT4	AUDMUX	AUD6_TXC	
	ALT6	SDMA	DEBUG_EVENT_CHANNEL[5]	
	ALT7	USBPHY2	VSTATUS[7]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_GDCLK	ALT0	EPDC	GDCLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EPDC_GDCLK	ALT1	GPIO3	GPIO[16]	
	ALT2	EIM	D[16]	
	ALT3	ELCDIF	DAT[16]	
	ALT4	AUDMUX	AUD6_TXFS	
	ALT6	SDMA	DEBUG_CORE_STATE[0]	
	ALT7	USBPHY2	BISTOK	
EPDC_GDSP	ALT0	EPDC	GDSP	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[17]	
	ALT2	EIM	D[17]	
	ALT3	ELCDIF	DAT[17]	
	ALT4	AUDMUX	AUD6_RXD	
	ALT6	SDMA	DEBUG_CORE_STATE[1]	
	ALT7	USBPHY2	BVALID	
EPDC_GDOE	ALT0	EPDC	GDOE	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[18]	
	ALT2	EIM	D[18]	
	ALT3	ELCDIF	DAT[18]	
	ALT4	AUDMUX	AUD6_RXC	
	ALT6	SDMA	DEBUG_CORE_STATE[2]	
	ALT7	USBPHY2	ENDSESSION	
EPDC_GDRL	ALT0	EPDC	GDRL	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EPDC_GDRL	ALT1	GPIO3	GPIO[19]	
	ALT2	EIM	D[19]	
	ALT3	ELCDIF	DAT[19]	
	ALT4	AUDMUX	AUD6_RXFS	
	ALT6	SDMA	DEBUG_CORE_STATE[3]	
	ALT7	USBPHY2	IDDIG	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_SDCLK	ALT0	EPDC	SDCLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[20]	
	ALT2	EIM	D[20]	
	ALT3	ELCDIF	DAT[20]	
	ALT4	AUDMUX	AUD5_TXD	
	ALT6	SDMA	DEBUG_BUS_DEVICE[0]	
	ALT7	USBPHY2	HOSTDISCONNECT	
EPDC_SDOEZ	ALT0	EPDC	SDOEZ	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[21]	
	ALT2	EIM	D[21]	
	ALT3	ELCDIF	DAT[21]	
	ALT4	AUDMUX	AUD5_TXC	
	ALT6	SDMA	DEBUG_BUS_DEVICE[1]	
	ALT7	USBPHY2	TXREADY	
EPDC_SDOED	ALT0	EPDC	SDOED	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EPDC_SDOED	ALT1	GPIO3	GPIO[22]	
	ALT2	EIM	D[22]	
	ALT3	ELCDIF	DAT[22]	
	ALT4	AUDMUX	AUD5_TXFS	
	ALT6	SDMA	DEBUG_BUS_DEVICE[2]	
	ALT7	USBPHY2	RXVALID	
EPDC_SDOE	ALT0	EPDC	SDOE	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[23]	
	ALT2	EIM	D[23]	
	ALT3	ELCDIF	DAT[23]	
	ALT4	AUDMUX	AUD5_RXD	
	ALT6	SDMA	DEBUG_BUS_DEVICE[3]	
	ALT7	USBPHY2	RXACTIVE	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_SDLE	ALT0	EPDC	SDLE	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[24]	
	ALT2	EIM	D[24]	
	ALT3	ELCDIF	DAT[8]	
	ALT4	AUDMUX	AUD5_RXC	
	ALT6	SDMA	DEBUG_BUS_DEVICE[4]	
	ALT7	USBPHY2	RXERROR	
EPDC_SDCLKN	ALT0	EPDC	SDCLKN	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EPDC_SDCLKN	ALT1	GPIO3	GPIO[25]	
	ALT2	EIM	D[25]	
	ALT3	ELCDIF	DAT[9]	
	ALT4	AUDMUX	AUD5_RXFS	
	ALT6	SDMA	DEBUG_BUS_ERROR	
	ALT7	USBPHY2	SIECLOCK	
EPDC_SDSHR	ALT0	EPDC	SDSHR	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[26]	
	ALT2	EIM	D[26]	
	ALT3	ELCDIF	DAT[10]	
	ALT4	AUDMUX	AUD4_TXD	
	ALT6	SDMA	DEBUG_BUS_RWB	
	ALT7	USBPHY2	LINESTATE[0]	
EPDC_PWRCOM	ALT0	EPDC	PWRCOM	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[27]	
	ALT2	EIM	D[27]	
	ALT3	ELCDIF	DAT[11]	
	ALT4	AUDMUX	AUD4_TXC	
	ALT6	SDMA	DEBUG_CORE_RUN	
	ALT7	USBPHY2	LINESTATE[1]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_PWRSTAT	ALT0	EPDC	PWRSTAT	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EPDC_PWRSTAT	ALT1	GPIO3	GPIO[28]	
	ALT2	EIM	D[28]	
	ALT3	ELCDIF	DAT[12]	
	ALT4	AUDMUX	AUD4_TXFS	
	ALT6	SDMA	DEBUG_MODE	
	ALT7	USBPHY2	VBUSVALID	
EPDC_PWRCTRL0	ALT0	EPDC	PWRCTRL[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[29]	
	ALT2	EIM	D[29]	
	ALT3	ELCDIF	DAT[13]	
	ALT4	AUDMUX	AUD4_RXD	
	ALT6	SDMA	DEBUG_RTBUFFER_WRITE	
	ALT7	USBPHY2	AVALID	
EPDC_PWRCTRL1	ALT0	EPDC	PWRCTRL[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO3	GPIO[30]	
	ALT2	EIM	D[30]	
	ALT3	ELCDIF	DAT[14]	
	ALT4	AUDMUX	AUD4_RXC	
	ALT6	SDMA	DEBUG_YIELD	
	ALT7	USBPHY1	ONBIST	
EPDC_PWRCTRL2	ALT0	EPDC	PWRCTRL[2]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EPDC_PWRCTRL2	ALT1	GPIO3	GPIO[31]	
	ALT2	EIM	D[31]	
	ALT3	ELCDIF	DAT[15]	
	ALT4	AUDMUX	AUD4_RXFS	
	ALT6	SDMA	SDMA_EXT_EVENT[0]	
	ALT7	USBPHY2	ONBIST	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_PWRCTRL3	ALT0	EPDC	PWRCTRL[3]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[20]	
	ALT2	EIM	EB[2]	
	ALT6	SDMA	SDMA_EXT_EVENT[1]	
	ALT7	USBPHY1	BISTOK	
EPDC_VCOM0	ALT0	EPDC	VCOM[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[21]	
	ALT2	EIM	EB[3]	
	ALT7	USBPHY2	BISTOK	
EPDC_VCOM1	ALT0	EPDC	VCOM[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[22]	
	ALT2	EIM	CS[3]	
EPDC_BDR0	ALT0	EPDC	BDR[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[23]	
	ALT3	ELCDIF	DAT[7]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_BDR1	ALT0	EPDC	BDR[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[24]	
	ALT3	ELCDIF	DAT[6]	
EPDC_SDCE0	ALT0	EPDC	SDCE[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[25]	
	ALT3	ELCDIF	DAT[5]	
EPDC_SDCE1	ALT0	EPDC	SDCE[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[26]	
	ALT3	ELCDIF	DAT[4]	
EPDC_SDCE2	ALT0	EPDC	SDCE[2]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[27]	
	ALT3	ELCDIF	DAT[3]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EPDC_SDCE3	ALT0	EPDC	SDCE[3]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[28]	
	ALT3	ELCDIF	DAT[2]	
EPDC_SDCE4	ALT0	EPDC	SDCE[4]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[29]	
	ALT3	ELCDIF	DAT[1]	
EPDC_SDCE5	ALT0	EPDC	SDCE[5]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Keep) Pull Up / Down Config. - CFG(NA) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO4	GPIO[30]	
	ALT3	ELCDIF	DAT[0]	
EIM_DA0	ALT0	EIM	A[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[0]	
	ALT3	kpp	COL[4]	
	ALT6	tpiu	TRACE[0]	
	ALT7	SRC	BT_CFG1[0]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EIM_DA1	ALT0	EIM	A[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[1]	
	ALT3	kpp	ROW[4]	
EIM_DA1	ALT6	tpiu	TRACE[1]	
	ALT7	SRC	BT_CFG1[1]	
EIM_DA2	ALT0	EIM	A[2]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[2]	
	ALT3	kpp	COL[5]	
	ALT6	tpiu	TRACE[2]	
	ALT7	SRC	BT_CFG1[2]	
EIM_DA3	ALT0	EIM	A[3]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EIM_DA3	ALT1	GPIO1	GPIO[3]	
	ALT3	kpp	ROW[5]	
	ALT6	tpiu	TRACE[3]	
	ALT7	SRC	BT_CFG1[3]	
EIM_DA4	ALT0	EIM	A[4]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[4]	
	ALT3	kpp	COL[6]	
	ALT6	tpiu	TRACE[4]	
	ALT7	SRC	BT_CFG1[4]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EIM_DA5	ALT0	EIM	A[5]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[5]	
	ALT3	kpp	ROW[6]	
	ALT6	tpiu	TRACE[5]	
	ALT7	SRC	BT_CFG1[5]	
EIM_DA6	ALT0	EIM	A[6]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[6]	
	ALT3	kpp	COL[7]	
EIM_DA6	ALT6	tpiu	TRACE[6]	
	ALT7	SRC	BT_CFG1[6]	
EIM_DA7	ALT0	EIM	A[7]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[7]	
	ALT3	kpp	ROW[7]	
	ALT6	tpiu	TRACE[7]	
	ALT7	SRC	BT_CFG1[7]	
EIM_DA8	ALT0	EIM	A[8]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
EIM_DA8	ALT1	GPIO1	GPIO[8]	
	ALT2	rawnand	CLE	
	ALT6	tpiu	TRACE[8]	
	ALT7	SRC	BT_CFG2[0]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EIM_DA9	ALT0	EIM	A[9]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[9]	
	ALT2	rawnand	ALE	
	ALT6	tpiu	TRACE[9]	
	ALT7	SRC	BT_CFG2[1]	
EIM_DA10	ALT0	EIM	A[10]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[10]	
	ALT2	rawnand	CEN[0]	
	ALT6	tpiu	TRACE[10]	
	ALT7	SRC	BT_CFG2[2]	
EIM_DA11	ALT0	EIM	A[11]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[11]	
	ALT2	rawnand	CEN[1]	
EIM_DA11	ALT6	tpiu	TRACE[11]	
	ALT7	SRC	BT_CFG2[3]	
EIM_DA12	ALT0	EIM	A[12]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[12]	
	ALT2	rawnand	CEN[2]	
	ALT3	EPDC	SDCE[6]	
	ALT6	tpiu	TRACE[12]	
	ALT7	SRC	BT_CFG2[4]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EIM_DA13	ALT0	EIM	A[13]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[13]	
	ALT2	rawnand	CEN[3]	
	ALT3	EPDC	SDCE[7]	
	ALT6	tpiu	TRACE[13]	
	ALT7	SRC	BT_CFG2[5]	
EIM_DA14	ALT0	EIM	A[14]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[14]	
EIM_DA14	ALT2	rawnand	READY0	
	ALT3	EPDC	SDCE[8]	
	ALT6	tpiu	TRACE[14]	
	ALT7	SRC	BT_CFG2[6]	
EIM_DA15	ALT0	EIM	A[15]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[15]	
	ALT2	rawnand	DQS	
	ALT3	EPDC	SDCE[9]	
	ALT6	tpiu	TRACE[15]	
	ALT7	SRC	BT_CFG2[7]	
EIM_CS2	ALT0	EIM	CS[2]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[16]	
	ALT2	EIM	A[27]	
	ALT6	tpiu	TRCLK	
	ALT7	SRC	BT_CFG3[0]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EIM_CS1	ALT0	EIM	CS[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[17]	
	ALT6	tpiu	TRCTL	
EIM_CS1	ALT7	SRC	BT_CFG3[1]	
EIM_CS0	ALT0	EIM	CS[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[18]	
	ALT7	SRC	BT_CFG3[2]	
EIM_EB0	ALT0	EIM	EB[0]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[19]	
	ALT7	SRC	BT_CFG3[3]	
EIM_EB1	ALT0	EIM	EB[1]	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[20]	
	ALT7	SRC	BT_CFG3[4]	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EIM_WAIT	ALT0	EIM	WAIT	Drive Strength - CFG(Low) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[21]	
	ALT2	EIM	DTACK_B	
	ALT7	SRC	BT_CFG3[5]	
EIM_BCLK	ALT0	EIM	BCLK	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[22]	
	ALT7	SRC	BT_CFG3[6]	
EIM_RDY	ALT0	EIM	RDY	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[23]	
	ALT7	SRC	BT_CFG3[7]	
EIM_OE	ALT0	EIM	OE	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Disabled Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[24]	
	ALT7	SRC	INT_BOOT	

Table continues on the next page...

Table 4-1. Pin Assignments Report (continued)

Pad Name	Mode	Instance	Signal	Pad Settings
EIM_RW	ALT0	EIM	RW	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - Disabled Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[25]	
	ALT7	SRC	SYSTEM_RST	
EIM_LBA	ALT0	EIM	LBA	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[26]	
	ALT7	SRC	TESTER_ACK	
EIM_CRE	ALT0	EIM	CRE	Drive Strength - CFG(High) low/high output voltage - CFG(High) Hyst. Enable - NA Pull / Keep Select - CFG(Pull) Pull Up / Down Config. - CFG(100KOhm PU) dse test - regular Open Drain Enable - CFG(Disabled) Pull / Keep Enable - CFG(Enabled) test_ts - Disabled
	ALT1	GPIO1	GPIO[27]	

Table 4-2. Muxing Options Report

Instance	Signal	Pad	Mode
SJC	DE_B	EPITO	ALT7
	DONE	WDOG	ALT7
	FAIL	PWM1	ALT7
	JTAG_ACT	OWIRE	ALT7
	MOD	JTAG_MOD	No Muxing (ALT0)
	TCK	JTAG_TCK	No Muxing (ALT0)
	TDI	JTAG_TDI	No Muxing (ALT0)
	TDO	JTAG_TDO	No Muxing (ALT0)
	TMS	JTAG_TMS	No Muxing (ALT0)
	TRSTB	JTAG_TRSTB	No Muxing (ALT0)
WDOG1	WDOG_B	WDOG	ALT0
	WDOG_RST_B_DEB		ALT2
EIM	A[0]	EIM_DA0	ALT0
	A[10]	EIM_DA10	ALT0
	A[11]	EIM_DA11	ALT0
	A[12]	EIM_DA12	ALT0
	A[13]	EIM_DA13	ALT0
	A[14]	EIM_DA14	ALT0
	A[15]	EIM_DA15	ALT0
	A[16]	DISP_D0	ALT3
	A[17]	DISP_D1	ALT3
	A[18]	DISP_D2	ALT3
	A[19]	DISP_D3	ALT3
	A[1]	EIM_DA1	ALT0

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
EIM	A[20]	DISP_D4	ALT3
	A[21]	DISP_D5	ALT3
	A[22]	DISP_D6	ALT3
	A[23]	DISP_D7	ALT3
	A[24]	DISP_WR	ALT3
	A[25]	DISP_RD	ALT3
	A[26]	DISP_RS	ALT3
	A[27]	DISP_CS	ALT3
		EIM_CS2	ALT2
	A[2]	EIM_DA2	ALT0
	A[3]	EIM_DA3	ALT0
	A[4]	EIM_DA4	ALT0
	A[5]	EIM_DA5	ALT0
	A[6]	EIM_DA6	ALT0
	A[7]	EIM_DA7	ALT0
	A[8]	EIM_DA8	ALT0
	A[9]	EIM_DA9	ALT0
	BCLK	EIM_BCLK	ALT0
	CRE	EIM_CRE	ALT0
	CS[0]	EIM_CS0	ALT0
	CS[1]	EIM_CS1	ALT0
	CS[2]	EIM_CS2	ALT0
	CS[3]	DISP_BUSY	ALT4
		DISP_CS	ALT4

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
EIM	CS[3]	DISP_RESET	ALT4
		EPDC_VCOM1	ALT2
		SD3_WP	ALT5
	DTACK_B	EIM_WAIT	ALT2
	D[0]	EPDC_D0	ALT2
		SD2_D4	ALT4
	D[10]	ECSPI1_MISO	ALT7
		ECSPI2_MISO	ALT7
		EPDC_D10	ALT2
	D[11]	ECSPI1_SS0	ALT7
		ECSPI2_SS0	ALT7
		EPDC_D11	ALT2
	D[12]	EPDC_D12	ALT2
		UART3_TXD	ALT6
	D[13]	EPDC_D13	ALT2
		UART3_RXD	ALT6
	D[14]	EPDC_D14	ALT2
		UART4_TXD	ALT6
	D[15]	EPDC_D15	ALT2
		UART4_RXD	ALT6
	D[16]	EPDC_GDCLK	ALT2
	D[17]	EPDC_GDSP	ALT2
	D[18]	EPDC_GDOE	ALT2
	D[19]	EPDC_GDRL	ALT2

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
EIM	D[1]	EPDC_D1	ALT2
		SD2_D5	ALT4
	D[20]	EPDC_SDCLK	ALT2
	D[21]	EPDC_SDOEZ	ALT2
	D[22]	EPDC_SDOED	ALT2
	D[23]	EPDC_SDOE	ALT2
	D[24]	EPDC_SDLE	ALT2
	D[25]	EPDC_SDCLKN	ALT2
	D[26]	EPDC_SDSHR	ALT2
	D[27]	EPDC_PWRCOM	ALT2
	D[28]	EPDC_PWRSTAT	ALT2
	D[29]	EPDC_PWRCTRL0	ALT2
	D[2]	EPDC_D2	ALT2
		SD2_D6	ALT4
	D[30]	EPDC_PWRCTRL1	ALT2
	D[31]	EPDC_PWRCTRL2	ALT2
	D[3]	EPDC_D3	ALT2
		SD2_D7	ALT4
	D[4]	EPDC_D4	ALT2
		SD2_WP	ALT4
	D[5]	EPDC_D5	ALT2
		SD2_CD	ALT4
	D[6]	EPDC_D6	ALT2
		SSI_RXFS	ALT3

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
EIM	D[7]	EPDC_D7	ALT2
		SSI_RXC	ALT3
	D[8]	ECSPI1_SCLK	ALT7
		ECSPI2_SCLK	ALT7
		EPDC_D8	ALT2
	D[9]	ECSPI1_MOSI	ALT7
		ECSPI2_MOSI	ALT7
		EPDC_D9	ALT2
	EB[0]	EIM_EB0	ALT0
	EB[1]	EIM_EB1	ALT0
	EB[2]	EPDC_PWRCTRL3	ALT2
	EB[3]	EPDC_VCOM0	ALT2
	LBA	EIM_LBA	ALT0
	OE	EIM_OE	ALT0
	RDY	EIM_RDY	ALT0
	RW	EIM_RW	ALT0
	WAIT	EIM_WAIT	ALT0
TZIC	PWRFAIL_INT	I2C3_SDA	ALT3
ELCDIF	CS	DISP_CS	ALT0
		ECSPI2_SS0	ALT2
		EPDC_D1	ALT3
	DAT[0]	DISP_D0	ALT0
		EPDC_SDCE5	ALT3
	DAT[10]	DISP_D10	ALT0

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
ELCDIF	DAT[10]	EPDC_SDSHR	ALT3
	DAT[11]	DISP_D11	ALT0
		EPDC_PWRCOM	ALT3
	DAT[12]	DISP_D12	ALT0
		EPDC_PWRSTAT	ALT3
	DAT[13]	DISP_D13	ALT0
		EPDC_PWRCTRL0	ALT3
	DAT[14]	DISP_D14	ALT0
		EPDC_PWRCTRL1	ALT3
	DAT[15]	DISP_D15	ALT0
		EPDC_PWRCTRL2	ALT3
	DAT[16]	EPDC_GDCLK	ALT3
	DAT[17]	EPDC_GDSP	ALT3
	DAT[18]	EPDC_GDOE	ALT3
	DAT[19]	EPDC_GDRL	ALT3
	DAT[1]	DISP_D1	ALT0
		EPDC_SDCE4	ALT3
	DAT[20]	EPDC_SDCLK	ALT3
	DAT[21]	EPDC_SDOEZ	ALT3
	DAT[22]	EPDC_SDOED	ALT3
	DAT[23]	EPDC_SDOE	ALT3
	DAT[24]	EPDC_D8	ALT3
	DAT[25]	EPDC_D9	ALT3
	DAT[26]	EPDC_D10	ALT3

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
ELCDIF	DAT[27]	EPDC_D11	ALT3
	DAT[28]	EPDC_D12	ALT3
	DAT[29]	EPDC_D13	ALT3
	DAT[2]	DISP_D2	ALT0
		EPDC_SDCE3	ALT3
	DAT[30]	EPDC_D14	ALT3
	DAT[31]	EPDC_D15	ALT3
	DAT[3]	DISP_D3	ALT0
		EPDC_SDCE2	ALT3
	DAT[4]	DISP_D4	ALT0
		EPDC_SDCE1	ALT3
	DAT[5]	DISP_D5	ALT0
		EPDC_SDCE0	ALT3
	DAT[6]	DISP_D6	ALT0
		EPDC_BDR1	ALT3
	DAT[7]	DISP_D7	ALT0
		EPDC_BDR0	ALT3
	DAT[8]	DISP_D8	ALT0
		EPDC_SDLE	ALT3
	DAT[9]	DISP_D9	ALT0
		EPDC_SDCLKN	ALT3
	DOTCLK	DISP_WR	ALT2
		ECSPI2_SCLK	ALT5
		EPDC_D0	ALT4

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
ELCDIF	ENABLE	DISP_RD	ALT2
		ECSPI2_MOSI	ALT5
		EPDC_D1	ALT4
	HSYNC	DISP_BUSY	ALT0
		DISP_CS	ALT2
		ECSPI2_SS0	ALT5
		EPDC_D3	ALT4
	RD_E	DISP_RD	ALT0
		ECSPI2_MOSI	ALT2
		EPDC_D3	ALT3
	RESET	DISP_RESET	ALT0
	RS	DISP_RS	ALT0
		ECSPI2_MISO	ALT2
		EPDC_D0	ALT3
	VSYNC	DISP_RS	ALT2
		ECSPI2_MISO	ALT5
		EPDC_D2	ALT4
	WR_RWN	DISP_WR	ALT0
		ECSPI2_SCLK	ALT2
		EPDC_D2	ALT3
EPIT1	EPITO	EPITO	ALT0
USBPHY1	AVALID	DISP_WR	ALT7
	BISTOK	DISP_RESET	ALT7
		EPDC_PWRCTRL3	ALT7

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
USBPHY1	BVALID	DISP_RD	ALT7
	DATAOUT[0]	DISP_D8	ALT7
	DATAOUT[10]	SSI_TXD	ALT7
	DATAOUT[11]	SSI_RXD	ALT7
	DATAOUT[12]	SSI_RXFS	ALT7
	DATAOUT[13]	SSI_RXC	ALT7
	DATAOUT[14]	UART1_TXD	ALT7
	DATAOUT[15]	UART1_RXD	ALT7
	DATAOUT[1]	DISP_D9	ALT7
	DATAOUT[2]	DISP_D10	ALT7
	DATAOUT[3]	DISP_D11	ALT7
	DATAOUT[4]	DISP_D12	ALT7
	DATAOUT[5]	DISP_D13	ALT7
	DATAOUT[6]	DISP_D14	ALT7
	DATAOUT[7]	DISP_D15	ALT7
	DATAOUT[8]	SSI_TXFS	ALT7
	DATAOUT[9]	SSI_TXC	ALT7
	ENDSESSION	DISP_RS	ALT7
	HOSTDISCONNECT	DISP_BUSY	ALT7
	IDDIG	DISP_CS	ALT7
	LINESTATE[0]	KEY_ROW2	ALT7
	LINESTATE[1]	KEY_COL3	ALT7
	ONBIST	EPDC_PWRCTRL1	ALT7
	RXACTIVE	KEY_COL1	ALT7

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
USBPHY1	RXERROR	KEY_ROW1	ALT7
	RXVALID	KEY_ROW0	ALT7
	SIECLOCK	KEY_COL2	ALT7
	TXREADY	KEY_COL0	ALT7
	VBUSVALID	KEY_ROW3	ALT7
	VSTATUS[0]	DISP_D0	ALT7
	VSTATUS[1]	DISP_D1	ALT7
	VSTATUS[2]	DISP_D2	ALT7
	VSTATUS[3]	DISP_D3	ALT7
	VSTATUS[4]	DISP_D4	ALT7
	VSTATUS[5]	DISP_D5	ALT7
	VSTATUS[6]	DISP_D6	ALT7
	VSTATUS[7]	DISP_D7	ALT7
USBPHY2	AVALID	EPDC_PWRCTRL0	ALT7
	BISTOK	EPDC_GDCLK	ALT7
		EPDC_VCOM0	ALT7
	BVALID	EPDC_GDSP	ALT7
	DATAOUT[0]	EPDC_D0	ALT7
	DATAOUT[10]	UART2_TXD	ALT7
	DATAOUT[11]	UART2_RXD	ALT7
	DATAOUT[12]	UART2_CTS	ALT7
	DATAOUT[13]	UART2_RTS	ALT7
	DATAOUT[14]	UART3_TXD	ALT7
	DATAOUT[15]	UART3_RXD	ALT7

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
USBPHY2	DATAOUT[1]	EPDC_D1	ALT7
	DATAOUT[2]	EPDC_D2	ALT7
	DATAOUT[3]	EPDC_D3	ALT7
	DATAOUT[4]	EPDC_D4	ALT7
	DATAOUT[5]	EPDC_D5	ALT7
	DATAOUT[6]	EPDC_D6	ALT7
	DATAOUT[7]	EPDC_D7	ALT7
	DATAOUT[8]	UART1_CTS	ALT7
	DATAOUT[9]	UART1_RTS	ALT7
	ENDSESSION	EPDC_GDOE	ALT7
	HOSTDISCONNECT	EPDC_SDCLK	ALT7
	IDDIG	EPDC_GDRL	ALT7
	LINESTATE[0]	EPDC_SDSHR	ALT7
	LINESTATE[1]	EPDC_PWRCOM	ALT7
	ONBIST	EPDC_PWRCTRL2	ALT7
	RXACTIVE	EPDC_SDOE	ALT7
	RXERROR	EPDC_SDLE	ALT7
	RXVALID	EPDC_SDOED	ALT7
	SIECLOCK	EPDC_SDCLKN	ALT7
	TXREADY	EPDC_SDOEZ	ALT7
	VBUSVALID	EPDC_PWRSTAT	ALT7
	VSTATUS[0]	EPDC_D8	ALT7
	VSTATUS[1]	EPDC_D9	ALT7
	VSTATUS[2]	EPDC_D10	ALT7
USBPHY2	VSTATUS[3]	EPDC_D11	ALT7
	VSTATUS[4]	EPDC_D12	ALT7
	VSTATUS[5]	EPDC_D13	ALT7
	VSTATUS[6]	EPDC_D14	ALT7
	VSTATUS[7]	EPDC_D15	ALT7
mshc	BS	SD2_CMD	ALT2
	DATA[0]	SD2_D0	ALT2
	DATA[1]	SD2_D1	ALT2
	DATA[2]	SD2_D2	ALT2
	DATA[3]	SD2_D3	ALT2
	SCLK	SD2_CLK	ALT2

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
DRAM MC	A[0]	DRAM_A0	No Muxing (ALT0)
	A[10]	DRAM_A10	No Muxing (ALT0)
	A[11]	DRAM_A11	No Muxing (ALT0)
	A[12]	DRAM_A12	No Muxing (ALT0)
	A[13]	DRAM_A13	No Muxing (ALT0)
	A[14]	DRAM_A14	No Muxing (ALT0)
	A[1]	DRAM_A1	No Muxing (ALT0)
	A[2]	DRAM_A2	No Muxing (ALT0)
	A[3]	DRAM_A3	No Muxing (ALT0)
	A[4]	DRAM_A4	No Muxing (ALT0)
	A[5]	DRAM_A5	No Muxing (ALT0)
	A[6]	DRAM_A6	No Muxing (ALT0)
	A[7]	DRAM_A7	No Muxing (ALT0)

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
DRAM MC	A[8]	DRAM_A8	No Muxing (ALT0)
	A[9]	DRAM_A9	No Muxing (ALT0)
	BA[0]	DRAM_SDBA0	No Muxing (ALT0)
	BA[1]	DRAM_SDBA1	No Muxing (ALT0)
	BA[2]	DRAM_SDBA2	No Muxing (ALT0)
	CASN	DRAM_CAS	No Muxing (ALT0)
	CKE	DRAM_SDCKE	No Muxing (ALT0)
	CLKN[0]	DRAM_SDCLK_0_B	No Muxing (ALT0)
	CLKN[1]	DRAM_SDCLK_1_B	No Muxing (ALT0)
	CLK[0]	DRAM_SDCLK_0	No Muxing (ALT0)
	CLK[1]	DRAM_SDCLK_1	No Muxing (ALT0)
	CSN[0]	DRAM_CS0	No Muxing (ALT0)
	CSN[1]	DRAM_CS1	No Muxing (ALT0)
	DQM[0]	DRAM_DQM0	No Muxing (ALT0)
	DQM[1]	DRAM_DQM1	No Muxing (ALT0)
	DQM[2]	DRAM_DQM2	No Muxing (ALT0)
	DQM[3]	DRAM_DQM3	No Muxing (ALT0)
	DQSN[0]	DRAM_SDQS0_B	No Muxing (ALT0)
	DQSN[1]	DRAM_SDQS1_B	No Muxing (ALT0)
	DQSN[2]	DRAM_SDQS2_B	No Muxing (ALT0)
	DQSN[3]	DRAM_SDQS3_B	No Muxing (ALT0)
	DQS[0]	DRAM_SDQS0	No Muxing (ALT0)
	DQS[1]	DRAM_SDQS1	No Muxing (ALT0)
	DQS[2]	DRAM_SDQS2	No Muxing (ALT0)

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
DRAM MC	DQS[3]	DRAM_SDQS3	No Muxing (ALT0)
	D[0]	DRAM_D0	No Muxing (ALT0)
	D[10]	DRAM_D10	No Muxing (ALT0)
	D[11]	DRAM_D11	No Muxing (ALT0)
	D[12]	DRAM_D12	No Muxing (ALT0)
	D[13]	DRAM_D13	No Muxing (ALT0)
	D[14]	DRAM_D14	No Muxing (ALT0)
	D[15]	DRAM_D15	No Muxing (ALT0)
	D[16]	DRAM_D16	No Muxing (ALT0)
	D[17]	DRAM_D17	No Muxing (ALT0)
	D[18]	DRAM_D18	No Muxing (ALT0)
	D[19]	DRAM_D19	No Muxing (ALT0)
	D[1]	DRAM_D1	No Muxing (ALT0)
	D[20]	DRAM_D20	No Muxing (ALT0)
	D[21]	DRAM_D21	No Muxing (ALT0)
	D[22]	DRAM_D22	No Muxing (ALT0)
	D[23]	DRAM_D23	No Muxing (ALT0)
	D[24]	DRAM_D24	No Muxing (ALT0)
	D[25]	DRAM_D25	No Muxing (ALT0)
	D[26]	DRAM_D26	No Muxing (ALT0)
	D[27]	DRAM_D27	No Muxing (ALT0)
	D[28]	DRAM_D28	No Muxing (ALT0)
	D[29]	DRAM_D29	No Muxing (ALT0)
	D[2]	DRAM_D2	No Muxing (ALT0)

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
DRAM MC	D[30]	DRAM_D30	No Muxing (ALT0)
	D[31]	DRAM_D31	No Muxing (ALT0)
	D[3]	DRAM_D3	No Muxing (ALT0)
	D[4]	DRAM_D4	No Muxing (ALT0)
	D[5]	DRAM_D5	No Muxing (ALT0)
	D[6]	DRAM_D6	No Muxing (ALT0)
	D[7]	DRAM_D7	No Muxing (ALT0)
	D[8]	DRAM_D8	No Muxing (ALT0)
	D[9]	DRAM_D9	No Muxing (ALT0)
	ODT[0]	DRAM_SDODT0	No Muxing (ALT0)
	ODT[1]	DRAM_SDODT1	No Muxing (ALT0)
	OPEN	DRAM_OPEN	No Muxing (ALT0)
	OPENFB	DRAM_OPENFB	No Muxing (ALT0)
	RASN	DRAM_RAS	No Muxing (ALT0)
	WEN	DRAM_SDWE	No Muxing (ALT0)
tcu	TEST_MODE	TEST_MODE	No Muxing (ALT0)
SRTC	ALARM_DEB	I2C3_SDA	ALT4
	PMIC_ON_REQ	PMIC_ON_REQ	No Muxing (ALT0)

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
AUDMUX	AUD3_RXC	SSI_RXC	ALT0
	AUD3_RXD	SSI_RXD	ALT0
	AUD3_RXFS	SSI_RXFS	ALT0
	AUD3_TXC	SSI_TXC	ALT0
	AUD3_TXD	SSI_TXD	ALT0
	AUD3_TXFS	SSI_TXFS	ALT0
	AUD4_RXC	EPDC_PWRCTRL1	ALT4
AUDMUX	AUD4_RXC	SD2_D5	ALT2
	AUD4_RXD	EPDC_PWRCTRL0	ALT4
		SD2_D6	ALT2
	AUD4_RXFS	EPDC_PWRCTRL2	ALT4
		SD2_D4	ALT2
	AUD4_TXC	EPDC_PWRCOM	ALT4
		SD2_CD	ALT2
	AUD4_TXD	EPDC_SDSHR	ALT4
		SD2_WP	ALT2
	AUD4_TXFS	EPDC_PWRSTAT	ALT4
		SD2_D7	ALT2
	AUD5_RXC	EPDC_SDLE	ALT4
	AUD5_RXD	EPDC_SDOE	ALT4
	AUD5_RXFS	EPDC_SDCLKN	ALT4
	AUD5_TXC	EPDC_SDOEZ	ALT4
	AUD5_TXD	EPDC_SDCLK	ALT4
	AUD5_TXFS	EPDC_SDOED	ALT4
	AUD6_RXC	EPDC_GDOE	ALT4
	AUD6_RXD	EPDC_GDSP	ALT4
	AUD6_RXFS	EPDC_GDRL	ALT4
	AUD6_TXC	EPDC_D15	ALT4
	AUD6_TXD	EPDC_D14	ALT4
	AUD6_TXFS	EPDC_GDCLK	ALT4
EPDC	BDR[0]	EPDC_BDR0	ALT0

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
EPDC	BDR[1]	EPDC_BDR1	ALT0
	GDCLK	EPDC_GDCLK	ALT0
	GDOE	EPDC_GDOE	ALT0
	GDRL	EPDC_GDRL	ALT0
	GDSP	EPDC_GDSP	ALT0
	PWRCOM	EPDC_PWRCOM	ALT0
	PWRCTRL[0]	EPDC_PWRCTRL0	ALT0
	PWRCTRL[1]	EPDC_PWRCTRL1	ALT0
	PWRCTRL[2]	EPDC_PWRCTRL2	ALT0
	PWRCTRL[3]	EPDC_PWRCTRL3	ALT0
	PWRIRQ	OWIRE	ALT4
	PWRSTAT	EPDC_PWRSTAT	ALT0
	SDCE[0]	EPDC_SDCE0	ALT0
	SDCE[1]	EPDC_SDCE1	ALT0
	SDCE[2]	EPDC_SDCE2	ALT0
	SDCE[3]	EPDC_SDCE3	ALT0
	SDCE[4]	EPDC_SDCE4	ALT0
	SDCE[5]	EPDC_SDCE5	ALT0
	SDCE[6]	ECSPI1_SCLK	ALT5
		EIM_DA12	ALT3
	SDCE[7]	ECSPI1_MOSI	ALT5
		EIM_DA13	ALT3
	SDCE[8]	ECSPI1_MISO	ALT5
		EIM_DA14	ALT3

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
EPDC	SDCE[9]	ECSPI1_SS0	ALT5
		EIM_DA15	ALT3
	SDCLK	EPDC_SDCLK	ALT0
	SDCLKN	EPDC_SDCLKN	ALT0
	SDDO[0]	EPDC_D0	ALT0
	SDDO[10]	EPDC_D10	ALT0
	SDDO[11]	EPDC_D11	ALT0
	SDDO[12]	EPDC_D12	ALT0
	SDDO[13]	EPDC_D13	ALT0
	SDDO[14]	EPDC_D14	ALT0
	SDDO[15]	EPDC_D15	ALT0
	SDDO[1]	EPDC_D1	ALT0
	SDDO[2]	EPDC_D2	ALT0
	SDDO[3]	EPDC_D3	ALT0
	SDDO[4]	EPDC_D4	ALT0
	SDDO[5]	EPDC_D5	ALT0
	SDDO[6]	EPDC_D6	ALT0
	SDDO[7]	EPDC_D7	ALT0
	SDDO[8]	EPDC_D8	ALT0
	SDDO[9]	EPDC_D9	ALT0
	SDLE	EPDC_SDLE	ALT0
	SDOE	EPDC_SDOE	ALT0
	SDOED	EPDC_SDOED	ALT0
	SDOEZ	EPDC_SDOEZ	ALT0
EPDC	SDSHR	EPDC_SDSHR	ALT0
	VCOM[0]	EPDC_VCOM0	ALT0
	VCOM[1]	EPDC_VCOM1	ALT0

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
SDMA	DEBUG_BUS_DEVICE[0]	EPDC_SDCLK	ALT6
	DEBUG_BUS_DEVICE[1]	EPDC_SDOEZ	ALT6
	DEBUG_BUS_DEVICE[2]	EPDC_SDOED	ALT6
	DEBUG_BUS_DEVICE[3]	EPDC_SDOE	ALT6
	DEBUG_BUS_DEVICE[4]	EPDC_SDLE	ALT6
	DEBUG_BUS_ERROR	EPDC_SDCLKN	ALT6
	DEBUG_BUS_RWB	EPDC_SDSHR	ALT6
	DEBUG_CORE_RUN	EPDC_PWRCOM	ALT6
	DEBUG_CORE_STATE[0]	EPDC_GDCLK	ALT6
	DEBUG_CORE_STATE[1]	EPDC_GDSP	ALT6
	DEBUG_CORE_STATE[2]	EPDC_GDOE	ALT6
	DEBUG_CORE_STATE[3]	EPDC_GDRL	ALT6
	DEBUG_EVENT_CHANNEL[0]	EPDC_D10	ALT6
	DEBUG_EVENT_CHANNEL[1]	EPDC_D11	ALT6
	DEBUG_EVENT_CHANNEL[2]	EPDC_D12	ALT6
	DEBUG_EVENT_CHANNEL[3]	EPDC_D13	ALT6
	DEBUG_EVENT_CHANNEL[4]	EPDC_D14	ALT6
	DEBUG_EVENT_CHANNEL[5]	EPDC_D15	ALT6
	DEBUG_EVENT_CHANNEL_SEL	EPDC_D9	ALT6
	DEBUG_EVT_CHN_LINES[0]	EPDC_D0	ALT6
	DEBUG_EVT_CHN_LINES[1]	EPDC_D1	ALT6

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
SDMA	DEBUG_EVT_CHN_LINES[2]	EPDC_D2	ALT6
	DEBUG_EVT_CHN_LINES[3]	EPDC_D3	ALT6
	DEBUG_EVT_CHN_LINES[4]	EPDC_D4	ALT6
	DEBUG_EVT_CHN_LINES[5]	EPDC_D5	ALT6
	DEBUG_EVT_CHN_LINES[6]	EPDC_D6	ALT6
	DEBUG_EVT_CHN_LINES[7]	EPDC_D7	ALT6
	DEBUG_MATCHED_DMBUS	EPDC_D8	ALT6
	DEBUG_MODE	EPDC_PWRSTAT	ALT6
	DEBUG_PC[0]	DISP_D0	ALT6
	DEBUG_PC[10]	DISP_RS	ALT6
	DEBUG_PC[11]	DISP_CS	ALT6
	DEBUG_PC[12]	DISP_BUSY	ALT6
	DEBUG_PC[13]	DISP_RESET	ALT6
	DEBUG_PC[1]	DISP_D1	ALT6
	DEBUG_PC[2]	DISP_D2	ALT6
	DEBUG_PC[3]	DISP_D3	ALT6
	DEBUG_PC[4]	DISP_D4	ALT6
	DEBUG_PC[5]	DISP_D5	ALT6
	DEBUG_PC[6]	DISP_D6	ALT6
	DEBUG_PC[7]	DISP_D7	ALT6
	DEBUG_PC[8]	DISP_WR	ALT6
	DEBUG_PC[9]	DISP_RD	ALT6
	DEBUG_RTBUFFER_WRITE	EPDC_PWRCTRL0	ALT6
	DEBUG_YIELD	EPDC_PWRCTRL1	ALT6
SDMA	SDMA_EXT_EVENT[0]	EPDC_PWRCTRL2	ALT6
		KEY_COL3	ALT6
	SDMA_EXT_EVENT[1]	EPDC_PWRCTRL3	ALT6
		KEY_ROW3	ALT6

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
FEC	COL	DISP_D3	ALT4
		SSI_RXFS	ALT5
	MDC	I2C3_SCL	ALT2
		SSI_RXFS	ALT6
	MDIO	I2C3_SDA	ALT2
		SSI_RXC	ALT6
	RDATA[0]	DISP_D12	ALT6
		DISP_D4	ALT2
	RDATA[1]	DISP_D11	ALT6
		DISP_D3	ALT2
	RX_CLK	DISP_D6	ALT4
		SSI_RXC	ALT5
	RX_DV	DISP_D10	ALT6
		DISP_D2	ALT2
	RX_ER	DISP_D1	ALT2
		DISP_D9	ALT6
	TDATA[0]	DISP_D15	ALT6
		DISP_D7	ALT2
	TDATA[1]	DISP_D14	ALT6
		DISP_D6	ALT2
FEC	TX_CLK	DISP_D0	ALT2
		DISP_D8	ALT6
	TX_EN	DISP_D13	ALT6
		DISP_D5	ALT2
observe_mux	OBSRV_INT_OUT0	I2C3_SCL	ALT6
	OBSRV_INT_OUT1	I2C3_SDA	ALT6
	OBSRV_INT_OUT2	PWM1	ALT6
	OBSRV_INT_OUT3	PWM2	ALT6
	OBSRV_INT_OUT4	OWIRE	ALT6

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
CSPI	MISO	CSPI_MISO	ALT0
	MOSI	CSPI_MOSI	ALT0
	RDY	ECSPI1_SCLK	ALT2
		SSI_TXD	ALT4
	SCLK	CSPI_SCLK	ALT0
	SS0	CSPI_SS0	ALT0
	SS1	ECSPI1_MOSI	ALT2
		SSI_RXC	ALT4
	SS2	ECSPI1_MISO	ALT2
		SSI_RXFS	ALT4
	SS3	ECSPI1_SS0	ALT2
		SSI_RXD	ALT4
SRC	ANY_PU_RST	PWM2	ALT7
	BOOT_MODE[0]	BOOT_MODE0	No Muxing (ALT0)
	BOOT_MODE[1]	BOOT_MODE1	No Muxing (ALT0)

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
SRC	BT_CFG1[0]	EIM_DA0	ALT7
	BT_CFG1[1]	EIM_DA1	ALT7
	BT_CFG1[2]	EIM_DA2	ALT7
	BT_CFG1[3]	EIM_DA3	ALT7
	BT_CFG1[4]	EIM_DA4	ALT7
	BT_CFG1[5]	EIM_DA5	ALT7
	BT_CFG1[6]	EIM_DA6	ALT7
	BT_CFG1[7]	EIM_DA7	ALT7
	BT_CFG2[0]	EIM_DA8	ALT7
	BT_CFG2[1]	EIM_DA9	ALT7
	BT_CFG2[2]	EIM_DA10	ALT7
	BT_CFG2[3]	EIM_DA11	ALT7
	BT_CFG2[4]	EIM_DA12	ALT7
	BT_CFG2[5]	EIM_DA13	ALT7
	BT_CFG2[6]	EIM_DA14	ALT7
	BT_CFG2[7]	EIM_DA15	ALT7
	BT_CFG3[0]	EIM_CS2	ALT7
	BT_CFG3[1]	EIM_CS1	ALT7
	BT_CFG3[2]	EIM_CS0	ALT7
	BT_CFG3[3]	EIM_EB0	ALT7
	BT_CFG3[4]	EIM_EB1	ALT7
	BT_CFG3[5]	EIM_WAIT	ALT7
	BT_CFG3[6]	EIM_BCLK	ALT7
	BT_CFG3[7]	EIM_RDY	ALT7
SRC	BT_FUSE_RSV[0]	SSI_TXC	ALT6
	BT_FUSE_RSV[1]	SSI_TXFS	ALT6
	INT_BOOT	EIM_OE	ALT7
	POR_B	POR_B	No Muxing (ALT0)
	RESET_B	RESET_IN_B	No Muxing (ALT0)
	SYSTEM_RST	EIM_RW	ALT7
	TESTER_ACK	EIM_LBA	ALT7

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
USB1	USBH1_OC	OWIRE	ALT2
	USBH1_PWR	EPITO	ALT2
	USBOTG_OC	I2C3_SCL	ALT7
		PWM1	ALT2
	USBOTG_PWR	I2C3_SDA	ALT7
		PWM2	ALT2
tpiu	TRACE[0]	EIM_DA0	ALT6
	TRACE[10]	EIM_DA10	ALT6
	TRACE[11]	EIM_DA11	ALT6
	TRACE[12]	EIM_DA12	ALT6
	TRACE[13]	EIM_DA13	ALT6
	TRACE[14]	EIM_DA14	ALT6
	TRACE[15]	EIM_DA15	ALT6
	TRACE[1]	EIM_DA1	ALT6
	TRACE[2]	EIM_DA2	ALT6
	TRACE[3]	EIM_DA3	ALT6
	TRACE[4]	EIM_DA4	ALT6
tpiu	TRACE[5]	EIM_DA5	ALT6
	TRACE[6]	EIM_DA6	ALT6
	TRACE[7]	EIM_DA7	ALT6
	TRACE[8]	EIM_DA8	ALT6
	TRACE[9]	EIM_DA9	ALT6
	TRCLK	EIM_CS2	ALT6
	TRCTL	EIM_CS1	ALT6
GPC	PMIC_RDY	I2C3_SCL	ALT3

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
kpp	COL[0]	KEY_COL0	ALT0
	COL[1]	KEY_COL1	ALT0
	COL[2]	KEY_COL2	ALT0
	COL[3]	KEY_COL3	ALT0
	COL[4]	DISP_D8	ALT5
		EIM_DA0	ALT3
		SD2_D0	ALT3
	COL[5]	DISP_D10	ALT5
		EIM_DA2	ALT3
		SD2_D2	ALT3
	COL[6]	DISP_D12	ALT5
		EIM_DA4	ALT3
		SD2_D4	ALT3
	COL[7]	DISP_D14	ALT5
		EIM_DA6	ALT3
		SD2_D6	ALT3
kpp	ROW[0]	KEY_ROW0	ALT0
	ROW[1]	KEY_ROW1	ALT0
	ROW[2]	KEY_ROW2	ALT0
	ROW[3]	KEY_ROW3	ALT0
	ROW[4]	DISP_D9	ALT5
		EIM_DA1	ALT3
		SD2_D1	ALT3
	ROW[5]	DISP_D11	ALT5
		EIM_DA3	ALT3
		SD2_D3	ALT3
	ROW[6]	DISP_D13	ALT5
		EIM_DA5	ALT3
		SD2_D5	ALT3
	ROW[7]	DISP_D15	ALT5
		EIM_DA7	ALT3
		SD2_D7	ALT3

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
GPT	CAPIN1	I2C3_SCL	ALT5
	CAPIN2	I2C3_SDA	ALT5
	CLKIN	EPITO	ALT5
	CMPOUT1	PWM1	ALT5
	CMPOUT2	PWM2	ALT5
	CMPOUT3	OWIRE	ALT5
CCM	CCM_OUT_0	SD2_D4	ALT7
	CCM_OUT_1	SD2_D5	ALT7
CCM	CCM_OUT_2	SD2_D6	ALT7
	CLKO	SD1_CLK	ALT7
	CLKO2	SD1_CMD	ALT7
	PLL1_BYP	SD1_D0	ALT7
		SD3_D0	ALT7
	PLL2_BYP	SD1_D1	ALT7
		SD3_D1	ALT7
	PLL3_BYP	SD1_D2	ALT7
		SD3_D2	ALT7
	PMIC_VSTBY_REQ	PMIC_STBY_REQ	No Muxing (ALT0)
	REF_EN_B	SD2_CD	ALT7
	SSI_EXT1_CLK	OWIRE	ALT3
	SSI_EXT2_CLK	EPITO	ALT3
	STOP	SD2_D7	ALT7
	WAIT	SD2_WP	ALT7
	XTAL32K	WDOG	ALT6
UART1	CTS	UART1_CTS	ALT0
	RTS	UART1_RTS	ALT0
	RXD_MUX	UART1_RXD	ALT0
	TXD_MUX	UART1_TXD	ALT0

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
UART2	CTS	I2C2_SCL	ALT2
		UART2_CTS	ALT0
	RTS	I2C2_SDA	ALT2
		UART2_RTS	ALT0
	RXD_MUX	I2C1_SDA	ALT2
		UART2_RXD	ALT0
	TXD_MUX	I2C1_SCL	ALT2
		UART2_TXD	ALT0
UART3	CTS	ECSPI1_MOSI	ALT4
		UART4_TXD	ALT2
	RTS	ECSPI1_SCLK	ALT4
		UART4_RXD	ALT2
	RXD_MUX	UART3_RXD	ALT0
	TXD_MUX	UART3_TXD	ALT0
UART4	CTS	ECSPI1_SS0	ALT4
	RTS	ECSPI1_MISO	ALT4
	RXD_MUX	UART4_RXD	ALT0
	TXD_MUX	UART4_TXD	ALT0
UART5	CTS	ECSPI2_MOSI	ALT4
	RTS	ECSPI2_SCLK	ALT4
	RXD_MUX	ECSPI2_SS0	ALT4
		SSI_RXC	ALT2
		UART1_RTS	ALT2
	TXD_MUX	ECSPI2_MISO	ALT4
UART5	TXD_MUX	SSI_RXFS	ALT2
		UART1_CTS	ALT2
PWM1	PWMO	PWM1	ALT0
PWM2		PWM2	ALT0

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
ECSPI1	MISO	ECSPI1_MISO	ALT0
	MOSI	ECSPI1_MOSI	ALT0
	RDY	ECSPI2_SCLK	ALT3
	SCLK	ECSPI1_SCLK	ALT0
	SS0	ECSPI1_SS0	ALT0
	SS1	ECSPI2_MOSI	ALT3
	SS2	ECSPI2_MISO	ALT3
	SS3	ECSPI2_SS0	ALT3
ECSPI2	MISO	ECSPI2_MISO	ALT0
	MOSI	ECSPI2_MOSI	ALT0
	RDY	ECSPI1_SCLK	ALT3
	SCLK	ECSPI2_SCLK	ALT0
	SS0	ECSPI2_SS0	ALT0
	SS1	ECSPI1_MOSI	ALT3
	SS2	ECSPI1_MISO	ALT3
	SS3	ECSPI1_SS0	ALT3
rawnand	ALE	DISP_D9	ALT2
		EIM_DA9	ALT2
		KEY_ROW0	ALT2
	CEN[0]	DISP_D10	ALT2

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
rawnand	CEN[0]	EIM_DA10	ALT2
		KEY_COL1	ALT2
	CEN[1]	DISP_D11	ALT2
		EIM_DA11	ALT2
		KEY_ROW1	ALT2
	CEN[2]	DISP_D12	ALT2
		EIM_DA12	ALT2
		KEY_COL2	ALT2
	CEN[3]	DISP_D13	ALT2
		EIM_DA13	ALT2
		KEY_ROW2	ALT2
	CEN[4]	ECSPI2_SCLK	ALT6
	CEN[5]	ECSPI2_MOSI	ALT6
	CEN[6]	ECSPI2_MISO	ALT6
	CEN[7]	ECSPI2_SS0	ALT6
	CLE	DISP_D8	ALT2
		EIM_DA8	ALT2
		KEY_COL0	ALT2
	DQS	DISP_D15	ALT2
		EIM_DA15	ALT2
		KEY_ROW3	ALT2
	D[0]	SD3_D4	ALT2
	D[1]	SD3_D5	ALT2
	D[2]	SD3_D6	ALT2
rawnand	D[3]	SD3_D7	ALT2
	D[4]	SD3_D0	ALT2
	D[5]	SD3_D1	ALT2
	D[6]	SD3_D2	ALT2
	D[7]	SD3_D3	ALT2
	RDN	SD3_CLK	ALT2
	READY0	DISP_D14	ALT2
		EIM_DA14	ALT2
		KEY_COL3	ALT2
	RESETN	SD3_WP	ALT2
	WRN	SD3_CMD	ALT2

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
DPLLC1	TOG_EN	EPITO	ALT4
owire	LINE	OWIRE	ALT0
I2C1	SCL	I2C1_SCL	ALT0
	SDA	I2C1_SDA	ALT0
I2C2	SCL	I2C2_SCL	ALT0
	SDA	I2C2_SDA	ALT0
I2C3	SCL	I2C3_SCL	ALT0
	SDA	I2C3_SDA	ALT0
ARM Platform	CTI_TRIGIN7	KEY_COL0	ALT6
	CTI_TRIGIN_ACK7	KEY_ROW0	ALT6
	CTI_TRIGOUT6	KEY_COL2	ALT6
	CTI_TRIGOUT7	KEY_ROW2	ALT6
	CTI_TRIGOUT_ACK6	KEY_COL1	ALT6
	CTI_TRIGOUT_ACK7	KEY_ROW1	ALT6
	PMU_IRQ_B	EPITO	ALT6
GPIO1	GPIO[0]	EIM_DA0	ALT1
	GPIO[10]	EIM_DA10	ALT1
	GPIO[11]	EIM_DA11	ALT1
	GPIO[12]	EIM_DA12	ALT1
	GPIO[13]	EIM_DA13	ALT1
	GPIO[14]	EIM_DA14	ALT1
	GPIO[15]	EIM_DA15	ALT1
	GPIO[16]	EIM_CS2	ALT1
	GPIO[17]	EIM_CS1	ALT1
	GPIO[18]	EIM_CS0	ALT1
	GPIO[19]	EIM_EB0	ALT1

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
GPIO1	GPIO[1]	EIM_DA1	ALT1
	GPIO[20]	EIM_EB1	ALT1
	GPIO[21]	EIM_WAIT	ALT1
	GPIO[22]	EIM_BCLK	ALT1
	GPIO[23]	EIM_RDY	ALT1
	GPIO[24]	EIM_OE	ALT1
	GPIO[25]	EIM_RW	ALT1
	GPIO[26]	EIM_LBA	ALT1
	GPIO[27]	EIM_CRE	ALT1
	GPIO[2]	EIM_DA2	ALT1
	GPIO[3]	EIM_DA3	ALT1
	GPIO[4]	EIM_DA4	ALT1
	GPIO[5]	EIM_DA5	ALT1
	GPIO[6]	EIM_DA6	ALT1
	GPIO[7]	EIM_DA7	ALT1
	GPIO[8]	EIM_DA8	ALT1
	GPIO[9]	EIM_DA9	ALT1
GPIO2	GPIO[0]	DISP_D0	ALT1
	GPIO[10]	DISP_D10	ALT1
	GPIO[11]	DISP_D11	ALT1
	GPIO[12]	DISP_D12	ALT1
	GPIO[13]	DISP_D13	ALT1
	GPIO[14]	DISP_D14	ALT1
	GPIO[15]	DISP_D15	ALT1

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
GPIO2	GPIO[16]	DISP_WR	ALT1
	GPIO[17]	DISP_RS	ALT1
	GPIO[18]	DISP_BUSY	ALT1
	GPIO[19]	DISP_RD	ALT1
	GPIO[1]	DISP_D1	ALT1
	GPIO[20]	DISP_RESET	ALT1
	GPIO[21]	DISP_CS	ALT1
	GPIO[2]	DISP_D2	ALT1
	GPIO[3]	DISP_D3	ALT1
	GPIO[4]	DISP_D4	ALT1
	GPIO[5]	DISP_D5	ALT1
	GPIO[6]	DISP_D6	ALT1
	GPIO[7]	DISP_D7	ALT1
	GPIO[8]	DISP_D8	ALT1
	GPIO[9]	DISP_D9	ALT1
GPIO3	GPIO[0]	EPDC_D0	ALT1
	GPIO[10]	EPDC_D10	ALT1
	GPIO[11]	EPDC_D11	ALT1
	GPIO[12]	EPDC_D12	ALT1
	GPIO[13]	EPDC_D13	ALT1
	GPIO[14]	EPDC_D14	ALT1
	GPIO[15]	EPDC_D15	ALT1
	GPIO[16]	EPDC_GDCLK	ALT1
	GPIO[17]	EPDC_GDSP	ALT1

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
GPIO3	GPIO[18]	EPDC_GDOE	ALT1
	GPIO[19]	EPDC_GDRL	ALT1
	GPIO[1]	EPDC_D1	ALT1
	GPIO[20]	EPDC_SDCLK	ALT1
	GPIO[21]	EPDC_SDOEZ	ALT1
	GPIO[22]	EPDC_SDOED	ALT1
	GPIO[23]	EPDC_SDOE	ALT1
	GPIO[24]	EPDC_SDLE	ALT1
	GPIO[25]	EPDC_SDCLKN	ALT1
	GPIO[26]	EPDC_SDSHR	ALT1
	GPIO[27]	EPDC_PWRCOM	ALT1
	GPIO[28]	EPDC_PWRSTAT	ALT1
	GPIO[29]	EPDC_PWRCTRL0	ALT1
	GPIO[2]	EPDC_D2	ALT1
	GPIO[30]	EPDC_PWRCTRL1	ALT1
	GPIO[31]	EPDC_PWRCTRL2	ALT1
	GPIO[3]	EPDC_D3	ALT1
	GPIO[4]	EPDC_D4	ALT1
	GPIO[5]	EPDC_D5	ALT1
	GPIO[6]	EPDC_D6	ALT1
	GPIO[7]	EPDC_D7	ALT1
	GPIO[8]	EPDC_D8	ALT1
	GPIO[9]	EPDC_D9	ALT1
GPIO4	GPIO[0]	KEY_COL0	ALT1

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
GPIO4	GPIO[10]	CSPI_MISO	ALT1
	GPIO[11]	CSPI_SS0	ALT1
	GPIO[12]	ECSPI1_SCLK	ALT1
	GPIO[13]	ECSPI1_MOSI	ALT1
	GPIO[14]	ECSPI1_MISO	ALT1
	GPIO[15]	ECSPI1_SS0	ALT1
	GPIO[16]	ECSPI2_SCLK	ALT1
	GPIO[17]	ECSPI2_MOSI	ALT1
	GPIO[18]	ECSPI2_MISO	ALT1
	GPIO[19]	ECSPI2_SS0	ALT1
	GPIO[1]	KEY_ROW0	ALT1
	GPIO[20]	EPDC_PWRCTRL3	ALT1
	GPIO[21]	EPDC_VCOM0	ALT1
	GPIO[22]	EPDC_VCOM1	ALT1
	GPIO[23]	EPDC_BDR0	ALT1
	GPIO[24]	EPDC_BDR1	ALT1
	GPIO[25]	EPDC_SDCE0	ALT1
	GPIO[26]	EPDC_SDCE1	ALT1
	GPIO[27]	EPDC_SDCE2	ALT1
	GPIO[28]	EPDC_SDCE3	ALT1
	GPIO[29]	EPDC_SDCE4	ALT1
	GPIO[2]	KEY_COL1	ALT1
	GPIO[30]	EPDC_SDCE5	ALT1
	GPIO[3]	KEY_ROW1	ALT1
GPIO4	GPIO[4]	KEY_COL2	ALT1
	GPIO[5]	KEY_ROW2	ALT1
	GPIO[6]	KEY_COL3	ALT1
	GPIO[7]	KEY_ROW3	ALT1
	GPIO[8]	CSPI_SCLK	ALT1
	GPIO[9]	CSPI_MOSI	ALT1

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
GPIO5	GPIO[0]	SD1_CLK	ALT1
	GPIO[10]	SD2_D2	ALT1
	GPIO[11]	SD2_D3	ALT1
	GPIO[12]	SD2_D4	ALT1
	GPIO[13]	SD2_D5	ALT1
	GPIO[14]	SD2_D6	ALT1
	GPIO[15]	SD2_D7	ALT1
	GPIO[16]	SD2_WP	ALT1
	GPIO[17]	SD2_CD	ALT1
	GPIO[18]	SD3_CMD	ALT1
	GPIO[19]	SD3_CLK	ALT1
	GPIO[1]	SD1_CMD	ALT1
	GPIO[20]	SD3_D0	ALT1
	GPIO[21]	SD3_D1	ALT1
	GPIO[22]	SD3_D2	ALT1
	GPIO[23]	SD3_D3	ALT1
	GPIO[24]	SD3_D4	ALT1
	GPIO[25]	SD3_D5	ALT1
GPIO5	GPIO[26]	SD3_D6	ALT1
	GPIO[27]	SD3_D7	ALT1
	GPIO[28]	SD3_WP	ALT1
	GPIO[2]	SD1_D0	ALT1
	GPIO[3]	SD1_D1	ALT1
	GPIO[4]	SD1_D2	ALT1
	GPIO[5]	SD1_D3	ALT1
	GPIO[6]	SD2_CLK	ALT1
	GPIO[7]	SD2_CMD	ALT1
	GPIO[8]	SD2_D0	ALT1
	GPIO[9]	SD2_D1	ALT1

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
GPIO6	GPIO[0]	SSI_TXFS	ALT1
	GPIO[10]	UART2_TXD	ALT1
	GPIO[11]	UART2_RXD	ALT1
	GPIO[12]	UART2_CTS	ALT1
	GPIO[13]	UART2_RTS	ALT1
	GPIO[14]	UART3_TXD	ALT1
	GPIO[15]	UART3_RXD	ALT1
	GPIO[16]	UART4_TXD	ALT1
	GPIO[17]	UART4_RXD	ALT1
	GPIO[18]	I2C1_SCL	ALT1
	GPIO[19]	I2C1_SDA	ALT1
	GPIO[1]	SSI_TXC	ALT1
	GPIO[20]	I2C2_SCL	ALT1
GPIO6	GPIO[21]	I2C2_SDA	ALT1
	GPIO[22]	I2C3_SCL	ALT1
	GPIO[23]	I2C3_SDA	ALT1
	GPIO[24]	PWM1	ALT1
	GPIO[25]	PWM2	ALT1
	GPIO[26]	OWIRE	ALT1
	GPIO[27]	EPITO	ALT1
	GPIO[28]	WDOG	ALT1
	GPIO[2]	SSI_TXD	ALT1
	GPIO[3]	SSI_RXD	ALT1
	GPIO[4]	SSI_RXFS	ALT1
	GPIO[5]	SSI_RXC	ALT1
	GPIO[6]	UART1_TXD	ALT1
	GPIO[7]	UART1_RXD	ALT1
	GPIO[8]	UART1_CTS	ALT1
	GPIO[9]	UART1_RTS	ALT1

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
ESDHC1	CD	DISP_D12	ALT3
	CLK	SD1_CLK	ALT0
	CMD	SD1_CMD	ALT0
	DAT0	SD1_D0	ALT0
	DAT1	SD1_D1	ALT0
	DAT2	SD1_D2	ALT0
	DAT3	SD1_D3	ALT0
	DAT4	UART3_TXD	ALT3
ESDHC1	DAT5	UART3_RXD	ALT3
	DAT6	UART4_TXD	ALT3
	DAT7	UART4_RXD	ALT3
	LCTL	DISP_D8	ALT3
		UART4_RXD	ALT5
	WP	DISP_D14	ALT3
ESDHC2	CD	SD2_CD	ALT0
		UART3_RXD	ALT5
	CLK	SD2_CLK	ALT0
	CMD	SD2_CMD	ALT0
	DAT0	SD2_D0	ALT0
	DAT1	SD2_D1	ALT0
	DAT2	SD2_D2	ALT0
	DAT3	SD2_D3	ALT0
	DAT4	SD2_D4	ALT0
	DAT5	SD2_D5	ALT0
	DAT6	SD2_D6	ALT0
	DAT7	SD2_D7	ALT0
	LCTL	DISP_D9	ALT3
		UART4_TXD	ALT5
	WP	SD2_WP	ALT0
		UART3_TXD	ALT5
ESDHC3	CD	DISP_D13	ALT3
	CLK	SD3_CLK	ALT0

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
ESDHC3	CMD	SD3_CMD	ALT0
	DAT0	SD3_D0	ALT0
	DAT1	SD3_D1	ALT0
	DAT2	SD3_D2	ALT0
	DAT3	SD3_D3	ALT0
	DAT4	SD3_D4	ALT0
	DAT5	SD3_D5	ALT0
	DAT6	SD3_D6	ALT0
	DAT7	SD3_D7	ALT0
	LCTL	DISP_D10	ALT3
	RST	DISP_D15	ALT3
	WP	SD3_WP	ALT0
ESDHC4	CD	DISP_D15	ALT4
	CLK	DISP_D9	ALT4
		UART1_RTS	ALT5
		UART2_RTS	ALT4
	CMD	DISP_D8	ALT4
		UART1_CTS	ALT5
		UART2_CTS	ALT4
	DAT0	DISP_D10	ALT4
		UART3_TXD	ALT4
	DAT1	DISP_D11	ALT4
		UART3_RXD	ALT4
	DAT2	DISP_D12	ALT4

Table continues on the next page...

Table 4-2. Muxing Options Report (continued)

Instance	Signal	Pad	Mode
ESDHC4	DAT2	UART4_TXD	ALT4
	DAT3	DISP_D13	ALT4
		UART4_RXD	ALT4
	DAT4	UART1_CTS	ALT4
		UART2_TXD	ALT5
	DAT5	UART1_RTS	ALT4
		UART2_RXD	ALT5
	DAT6	UART2_CTS	ALT5
		UART2_TXD	ALT4
	DAT7	UART2_RTS	ALT5
		UART2_RXD	ALT4
	LCTL	SD3_WP	ALT4
	WP	DISP_D14	ALT4

Table 4-3. Daisy Chain Report

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
AUDMUX	p4_input_da_amx	Module: AUDMUX, Protocol: AUD4_4W, Port: AUD4_RXD	SD2_D6	ALT2
			EPDC_PWRCTRL0	ALT4
	p4_input_db_amx	Module: AUDMUX, Protocol: AUD4_4W, Port: AUD4_TXD	SD2_WP	ALT2
			EPDC_SDSHR	ALT4
	p4_input_rxclk_amx	Module: AUDMUX, Protocol: AUD4_6W, Port: AUD4_RXC	SD2_D5	ALT2
			EPDC_PWRCTRL1	ALT4
	p4_input_rxfs_amx	Module: AUDMUX, Protocol: AUD4_6W, Port: AUD4_RXFS	SD2_D4	ALT2
			EPDC_PWRCTRL2	ALT4
AUDMUX	p4_input_txclk_amx	Module: AUDMUX, Protocol: AUD4_4W, Port: AUD4_TXC	SD2_CD	ALT2
			EPDC_PWRCOM	ALT4
	p4_input_txfs_amx	Module: AUDMUX, Protocol: AUD4_4W, Port: AUD4_TXFS	SD2_D7	ALT2
			EPDC_PWRSTAT	ALT4

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
CCM	pll1_bypass_clk	Module: CCM, Protocol: PLL1_BYP, Port: PLL1_BYP	SD1_D0	ALT7
			SD3_D0	ALT7
	pll2_bypass_clk	Module: CCM, Protocol: PLL2_BYP, Port: PLL2_BYP	SD1_D1	ALT7
			SD3_D1	ALT7
CCM	pll3_bypass_clk	Module: CCM, Protocol: PLL3_BYP, Port: PLL3_BYP	SD1_D2	ALT7
			SD3_D2	ALT7
CSPI	ipp_ind_dataready_b	Module: CSPI, Protocol: MASTER, Port: RDY	SSI_TXD	ALT4
			ECSPI1_SCLK	ALT2
	ipp_ind_ss1_b	Module: CSPI, Protocol: MASTER, Port: SS1	SSI_RXC	ALT4
			ECSPI1_MOSI	ALT2
	ipp_ind_ss2_b	Module: CSPI, Protocol: MASTER, Port: SS2	SSI_RXFS	ALT4
			ECSPI1_MISO	ALT2
CSPI	ipp_ind_ss3_b	Module: CSPI, Protocol: MASTER, Port: SS3	SSI_RXD	ALT4
			ECSPI1_SS0	ALT2
ELCDIF	lcdif_busy	Module: LCDIF, Protocol: DOTCLK, Port: HSYNC	ECSPI2_SS0	ALT5
			DISP_CS	ALT2
			DISP_BUSY	ALT0
			EPDC_D3	ALT4
	lcdif_rxdata[0]	Module: LCDIF, Protocol: 16B, Port: DAT[0]	DISP_D0	ALT0
			EPDC_SDCE5	ALT3

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
ELCDIF	lcdif_rxdata[1]	Module: LCDIF, Protocol: 16B, Port: DAT[1]	DISP_D1	ALT0
			EPDC_SDCE4	ALT3
	lcdif_rxdata[2]	Module: LCDIF, Protocol: 16B, Port: DAT[2]	DISP_D2	ALT0
			EPDC_SDCE3	ALT3
	lcdif_rxdata[3]	Module: LCDIF, Protocol: 16B, Port: DAT[3]	DISP_D3	ALT0
			EPDC_SDCE2	ALT3
	lcdif_rxdata[4]	Module: LCDIF, Protocol: 16B, Port: DAT[4]	DISP_D4	ALT0
			EPDC_SDCE1	ALT3
ELCDIF	lcdif_rxdata[5]	Module: LCDIF, Protocol: 16B, Port: DAT[5]	DISP_D5	ALT0
			EPDC_SDCE0	ALT3
	lcdif_rxdata[6]	Module: LCDIF, Protocol: 16B, Port: DAT[6]	DISP_D6	ALT0
			EPDC_BDR1	ALT3
	lcdif_rxdata[7]	Module: LCDIF, Protocol: 16B, Port: DAT[7]	DISP_D7	ALT0
			EPDC_BDR0	ALT3
	lcdif_rxdata[8]	Module: LCDIF, Protocol: 16B, Port: DAT[8]	DISP_D8	ALT0
			EPDC_SDLE	ALT3
ELCDIF	lcdif_rxdata[9]	Module: LCDIF, Protocol: 16B, Port: DAT[9]	DISP_D9	ALT0
			EPDC_SDCLKN	ALT3
	lcdif_rxdata[10]	Module: LCDIF, Protocol: 16B, Port: DAT[10]	DISP_D10	ALT0
			EPDC_SDSHR	ALT3
	lcdif_rxdata[11]	Module: LCDIF, Protocol: 16B, Port: DAT[11]	DISP_D11	ALT0
			EPDC_PWRCOM	ALT3
	lcdif_rxdata[12]	Module: LCDIF, Protocol: 16B, Port: DAT[12]	DISP_D12	ALT0
			EPDC_PWRSTAT	ALT3

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
ELCDIF	lcdif_rxdata[13]	Module: LCDIF, Protocol: 16B, Port: DAT[13]	DISP_D13	ALT0
			EPDC_PWRCTRL0	ALT3
	lcdif_rxdata[14]	Module: LCDIF, Protocol: 16B, Port: DAT[14]	DISP_D14	ALT0
			EPDC_PWRCTRL1	ALT3
	lcdif_rxdata[15]	Module: LCDIF, Protocol: 16B, Port: DAT[15]	DISP_D15	ALT0
			EPDC_PWRCTRL2	ALT3
	vsync_i	Module: LCDIF, Protocol: DOTCLK, Port: VSYNC	ECSPI2_MISO	ALT5
			DISP_RS	ALT2
ELCDIF	vsync_i	Module: LCDIF, Protocol: DOTCLK, Port: VSYNC	EPDC_D2	ALT4
ESDHC2	ipp_card_det	Module: ESDHC, Protocol: CARD_DET, Port: CD	UART3_RXD	ALT5
			SD2_CD	ALT0
	ipp_wp_on	Module: ESDHC, Protocol: WRITE_PROT, Port: WP	UART3_TXD	ALT5
			SD2_WP	ALT0
ESDHC4	ipp_card_clk_in	Module: ESDHC, Protocol: CE_ATA, Port: CLK	UART1_RTS	ALT5
			UART2_RTS	ALT4
			DISP_D9	ALT4
ESDHC4	ipp_cmd_in	Module: ESDHC, Protocol: CE_ATA, Port: CMD	UART1_CTS	ALT5
			UART2_CTS	ALT4
			DISP_D8	ALT4
	ipp_dat0_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT0	UART3_TXD	ALT4
			DISP_D10	ALT4
	ipp_dat1_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT1	UART3_RXD	ALT4
			DISP_D11	ALT4
	ipp_dat2_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT2	UART4_TXD	ALT4

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
ESDHC4	ipp_dat2_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT2	DISP_D12	ALT4
	ipp_dat3_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT3	UART4_RXD	ALT4
			DISP_D13	ALT4
	ipp_dat4_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT4	UART1_CTS	ALT4
			UART2_TXD	ALT5
	ipp_dat5_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT5	UART1_RTS	ALT4
			UART2_RXD	ALT5
	ipp_dat6_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT6	UART2_TXD	ALT4
ESDHC4	ipp_dat6_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT6	UART2_CTS	ALT5
	ipp_dat7_in	Module: ESDHC, Protocol: CE_ATA, Port: DAT7	UART2_RXD	ALT4
			UART2_RTS	ALT5
FEC	fec_col	Module: FEC, Protocol: MII, Port: COL	SSI_RXFS	ALT5
			DISP_D3	ALT4
	fec_mdi	Module: FEC, Protocol: MII, Port: MDIO	I2C3_SDA	ALT2
			SSI_RXC	ALT6
	fec_rdata[0]	Module: FEC, Protocol: MII, Port: RDATA[0]	DISP_D4	ALT2

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
FEC	fec_rdata[0]	Module: FEC, Protocol: MII, Port: RDATA[0]	DISP_D12	ALT6
	fec_rdata[1]	Module: FEC, Protocol: MII, Port: RDATA[1]	DISP_D3	ALT2
			DISP_D11	ALT6
	fec_rx_clk	Module: FEC, Protocol: MII, Port: RX_CLK	SSI_RXC	ALT5
			DISP_D6	ALT4
	fec_rx_dv	Module: FEC, Protocol: MII, Port: RX_DV	DISP_D2	ALT2
			DISP_D10	ALT6
	fec_rx_er	Module: FEC, Protocol: MII, Port: RX_ER	DISP_D1	ALT2
FEC	fec_rx_er	Module: FEC, Protocol: MII, Port: RX_ER	DISP_D9	ALT6
	fec_tx_clk	Module: FEC, Protocol: MII, Port: TX_CLK	DISP_D0	ALT2
			DISP_D8	ALT6
kpp	ipp_ind_col[4]	Module: KPP, Protocol: KPP, Port: COL[4]	SD2_D0	ALT3
			DISP_D8	ALT5
			EIM_DA0	ALT3
	ipp_ind_col[5]	Module: KPP, Protocol: KPP, Port: COL[5]	SD2_D2	ALT3
			DISP_D10	ALT5

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
kpp	ipp_ind_col[5]	Module: KPP, Protocol: KPP, Port: COL[5]	EIM_DA2	ALT3
	ipp_ind_col[6]	Module: KPP, Protocol: KPP, Port: COL[6]	SD2_D4	ALT3
			DISP_D12	ALT5
			EIM_DA4	ALT3
	ipp_ind_col[7]	Module: KPP, Protocol: KPP, Port: COL[7]	SD2_D6	ALT3
			DISP_D14	ALT5
			EIM_DA6	ALT3
	ipp_ind_row[4]	Module: KPP, Protocol: KPP, Port: ROW[4]	SD2_D1	ALT3
kpp	ipp_ind_row[4]	Module: KPP, Protocol: KPP, Port: ROW[4]	DISP_D9	ALT5
			EIM_DA1	ALT3
	ipp_ind_row[5]	Module: KPP, Protocol: KPP, Port: ROW[5]	SD2_D3	ALT3
			DISP_D11	ALT5
			EIM_DA3	ALT3
	ipp_ind_row[6]	Module: KPP, Protocol: KPP, Port: ROW[6]	SD2_D5	ALT3
			DISP_D13	ALT5
			EIM_DA5	ALT3
kpp	ipp_ind_row[7]	Module: KPP, Protocol: KPP, Port: ROW[7]	SD2_D7	ALT3
			DISP_D15	ALT5
			EIM_DA7	ALT3
rawnand	u_gpmi_input_gpmi_dqs_in	Module: GPMI, Protocol: GPMI, Port: DQS	KEY_ROW3	ALT2
			DISP_D15	ALT2
			EIM_DA15	ALT2
	u_gpmi_input_gpmi_rdy0	Module: GPMI, Protocol: 8RDY, Port: READY0	KEY_COL3	ALT2
			DISP_D14	ALT2
rawnand	u_gpmi_input_gpmi_rdy0	Module: GPMI, Protocol: 8RDY, Port: READY0	EIM_DA14	ALT2

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
SDMA	events[14]	Module: SDMA, Protocol: SDMA_EXT_EVENT, Port: SDMA_EXT_EVENT[0]	KEY_COL3	ALT6
			EPDC_PWRCTRL2	ALT6
	events[15]	Module: SDMA, Protocol: SDMA_EXT_EVENT, Port: SDMA_EXT_EVENT[1]	KEY_ROW3	ALT6
			EPDC_PWRCTRL3	ALT6
UART1	ipp_uart_rts_b	Module: UARTV2, Protocol: 4WIRE, Port: CTS	UART1_CTS	ALT0
		Module: UARTV2, Protocol: 4WIRE, Port: RTS	UART1_RTS	ALT0
	ipp_uart_rxd_mux	Module: UARTV2, Protocol: 2WIRE, Port: TXD_MUX	UART1_TXD	ALT0
UART1	ipp_uart_rxd_mux	Module: UARTV2, Protocol: 2WIRE, Port: RXD_MUX	UART1_RXD	ALT0

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
UART2	ipp_uart_rts_b	Module: UARTV2, Protocol: 4WIRE, Port: CTS	I2C2_SCL	ALT2
		Module: UARTV2, Protocol: 4WIRE, Port: RTS	I2C2_SDA	ALT2
		Module: UARTV2, Protocol: 4WIRE, Port: CTS	UART2_CTS	ALT0
		Module: UARTV2, Protocol: 4WIRE, Port: RTS	UART2_RTS	ALT0
	ipp_uart_rxd_mux	Module: UARTV2, Protocol: 2WIRE, Port: TXD_MUX	I2C1_SCL	ALT2
		Module: UARTV2, Protocol: 2WIRE, Port: RXD_MUX	I2C1_SDA	ALT2
		Module: UARTV2, Protocol: 2WIRE, Port: TXD_MUX	UART2_TXD	ALT0
UART2	ipp_uart_rxd_mux	Module: UARTV2, Protocol: 2WIRE, Port: RXD_MUX	UART2_RXD	ALT0

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
UART3	ipp_uart_rts_b	Module: UARTV2, Protocol: 4WIRE, Port: CTS	UART4_TXD	ALT2
		Module: UARTV2, Protocol: 4WIRE, Port: RTS	UART4_RXD	ALT2
			ECSPI1_SCLK	ALT4
		Module: UARTV2, Protocol: 4WIRE, Port: CTS	ECSPI1_MOSI	ALT4
	ipp_uart_rxd_mux	Module: UARTV2, Protocol: 2WIRE, Port: TXD_MUX	UART3_TXD	ALT0
		Module: UARTV2, Protocol: 2WIRE, Port: RXD_MUX	UART3_RXD	ALT0
UART4	ipp_uart_rts_b	Module: UARTV2, Protocol: 4WIRE, Port: RTS	ECSPI1_MISO	ALT4
UART4	ipp_uart_rts_b	Module: UARTV2, Protocol: 4WIRE, Port: CTS	ECSPI1_SS0	ALT4
	ipp_uart_rxd_mux	Module: UARTV2, Protocol: 2WIRE, Port: TXD_MUX	UART4_TXD	ALT0
		Module: UARTV2, Protocol: 2WIRE, Port: RXD_MUX	UART4_RXD	ALT0

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
UART5	ipp_uart_rts_b	Module: UARTV2, Protocol: 4WIRE, Port: RTS	ECSPI2_SCLK	ALT4
		Module: UARTV2, Protocol: 4WIRE, Port: CTS	ECSPI2_MOSI	ALT4
	ipp_uart_rxd_mux	Module: UARTV2, Protocol: 2WIRE, Port: TXD_MUX	SSI_RXFS	ALT2
		Module: UARTV2, Protocol: 2WIRE, Port: RXD_MUX	SSI_RXC	ALT2
		Module: UARTV2, Protocol: 2WIRE, Port: TXD_MUX	UART1_CTS	ALT2
UART5	ipp_uart_rxd_mux	Module: UARTV2, Protocol: 2WIRE, Port: RXD_MUX	UART1_RTS	ALT2
		Module: UARTV2, Protocol: 2WIRE, Port: TXD_MUX	ECSPI2_MISO	ALT4
		Module: UARTV2, Protocol: 2WIRE, Port: RXD_MUX	ECSPI2_SS0	ALT4
USB1	ipp_ind_otg_oc	Module: USBOH1, Protocol: OTGUSB_OC, Port: USBOTG_OC	I2C3_SCL	ALT7
			PWM1	ALT2
EIM	ipp_ind_read_data[0]	Module: EIM, Protocol: 16B, Port: D[0]	SD2_D4	ALT4
			EPDC_D0	ALT2
	ipp_ind_read_data[1]	Module: EIM, Protocol: 16B, Port: D[1]	SD2_D5	ALT4

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
EIM	ipp_ind_read_data[1]	Module: EIM, Protocol: 16B, Port: D[1]	EPDC_D1	ALT2
	ipp_ind_read_data[2]	Module: EIM, Protocol: 16B, Port: D[2]	SD2_D6	ALT4
			EPDC_D2	ALT2
	ipp_ind_read_data[3]	Module: EIM, Protocol: 16B, Port: D[3]	SD2_D7	ALT4
			EPDC_D3	ALT2
	ipp_ind_read_data[4]	Module: EIM, Protocol: 16B, Port: D[4]	SD2_WP	ALT4
			EPDC_D4	ALT2
	ipp_ind_read_data[5]	Module: EIM, Protocol: 16B, Port: D[5]	SD2_CD	ALT4
EIM	ipp_ind_read_data[5]	Module: EIM, Protocol: 16B, Port: D[5]	EPDC_D5	ALT2
	ipp_ind_read_data[6]	Module: EIM, Protocol: 16B, Port: D[6]	SSI_RXFS	ALT3
			EPDC_D6	ALT2
	ipp_ind_read_data[7]	Module: EIM, Protocol: 16B, Port: D[7]	SSI_RXC	ALT3
			EPDC_D7	ALT2
	ipp_ind_read_data[8]	Module: EIM, Protocol: 16B, Port: D[8]	ECSPI1_SCLK	ALT7
			ECSPI2_SCLK	ALT7
			EPDC_D8	ALT2
EIM	ipp_ind_read_data[9]	Module: EIM, Protocol: 16B, Port: D[9]	ECSPI1_MOSI	ALT7
			ECSPI2_MOSI	ALT7
			EPDC_D9	ALT2
	ipp_ind_read_data[10]	Module: EIM, Protocol: 16B, Port: D[10]	ECSPI1_MISO	ALT7
			ECSPI2_MISO	ALT7
			EPDC_D10	ALT2
	ipp_ind_read_data[11]	Module: EIM, Protocol: 16B, Port: D[11]	ECSPI1_SS0	ALT7
			ECSPI2_SS0	ALT7

Table continues on the next page...

Table 4-3. Daisy Chain Report (continued)

Instance	IN Pin	Module, Protocol, Port	Pad	Mode
EIM	ipp_ind_read_data[11]	Module: EIM, Protocol: 16B, Port: D[11]	EPDC_D11	ALT2
	ipp_ind_read_data[12]	Module: EIM, Protocol: 16B, Port: D[12]	UART3_TXD	ALT6
			EPDC_D12	ALT2
	ipp_ind_read_data[13]	Module: EIM, Protocol: 16B, Port: D[13]	UART3_RXD	ALT6
			EPDC_D13	ALT2
	ipp_ind_read_data[14]	Module: EIM, Protocol: 16B, Port: D[14]	UART4_TXD	ALT6
			EPDC_D14	ALT2
	ipp_ind_read_data[15]	Module: EIM, Protocol: 16B, Port: D[15]	UART4_RXD	ALT6
EIM	ipp_ind_read_data[15]	Module: EIM, Protocol: 16B, Port: D[15]	EPDC_D15	ALT2

Chapter 5
Clock Controller Module (CCM)

5.1 Introduction

The Clock Controller Module (CCM) controls the clocks for i.MX50 blocks. This block uses the available clock sources to generate the root clocks. Figure 5-1 shows a high level CCM clock tree diagram.

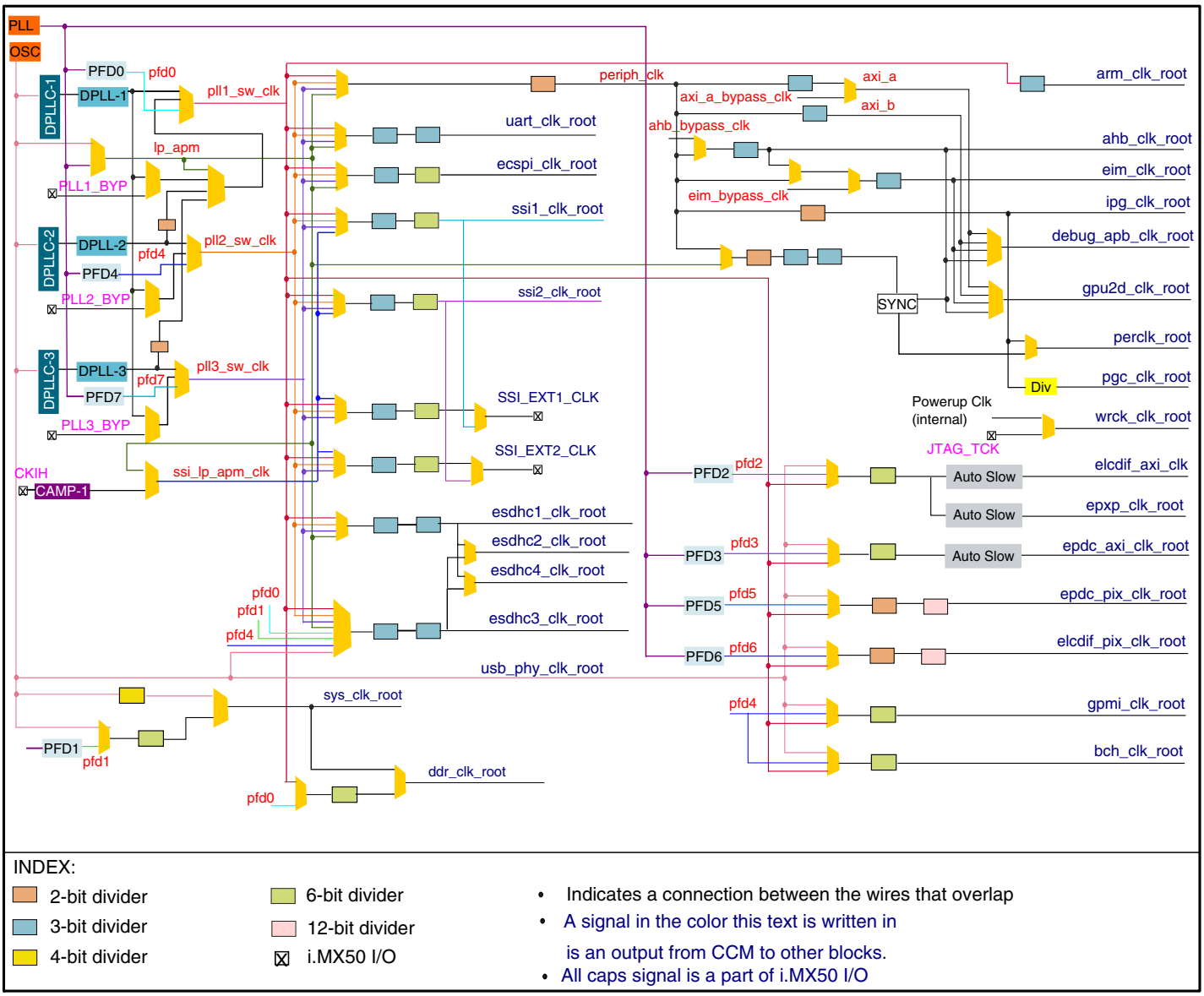


Figure 5-1. High Level Clock Tree Diagram

5.1.1 Overview

The Clock Controller Module controls the following functions in i.MX50:

- Uses the available clock sources to generate root clocks for various parts of the chip.
- Uses programmable bits to control frequencies of the clock roots.
- Controls the low power mechanism.
- Provides control signals to LPCG sub-block for gating clocks.
- Provides handshake with SRC for reset performance.
- Provides handshake with GPC for support of DVFS, DPTC and power gating operations.

5.1.2 Features

The CCM includes the following features:

- Clock switcher module to generate three switcher source clocks based on four PLLs.
- Root clocks generation—provides root clocks to i.MX50's modules based on three switcher source clocks.
- ARM core root clock generated from dedicated switcher source clock.
- Includes separate dividers to control generation of core and bus root clocks .
- Includes separate dividers and clock source selectors for each serial root clock.
- Option for external clocks to bypass PLL clocks.
- Selects clock signals to output on CLKO and CKOH on to the pads for observability.
- Controllable registers are accessible through IP bus.
- Manages the Low Power Modes, namely RUN, WAIT, and STOP. The peripheral clock gating is programmable in RUN and WAIT modes.
- Manages frequency scaling procedure for ARM core clock by shifting between PLL sources, without loss of clocks.
- Manages frequency scaling procedure for peripheral clock roots by programable divider. The dividing is done on the fly without loss of clocks.
- DFT feature support.
- Interface for the following IPs
    - PLLC—PLL Controller interfaces for each PLL on the IC.
    - LPCG—Low Power Clock Gating unit
    - SRC —System reset Controller
    - GPC—General Power Controller

5.1.3 CCM Block Diagram

CCM includes the following submodules:

CCM_CLK_IGNITION

Manages the ignition process. This sub-block starts its functionality once CCM comes out of reset. It manages the process that begins with starting the CAMPs, PLLs and finishes with creation of stable output root clocks after reset.

CCM_CLK_SWITCHER

This sub-block receives the clock outputs of the three PLLs, together with the bypass clocks for three PLLs , and generates three switcher clock outputs (pll1_sw_clk, pll2_sw_clk, pll3_sw_clk) for the ccm_clk_root_gen sub-block.

CCM_CLK_ROOT_GEN

This sub-block receives the three switcher clocks and generates the output root clocks.

CCM_CLK_LOGIC

This sub-block generates the clock enables based on information from CCM_LPM and CCM_IP. The clock enables are used by LPCG to turn off and on the split-ed clocks.

CCM_LPM

This sub-block manages the low power modes of the IC.

CCM_REG

This sub-block manages the memory map of CCM. It holds the programmable registers and the connectivity to the IP bus. This module is connected to all the sub-blocks that need definition of programmable bits. Note that in the diagram below it appears to be connected only to the ccm_clk_logic sub-block.

CCM_HND_SK

This sub-block manages the handshake when changing root clock dividers that require handshake, and manages the frequency change in case of DVFS scenario.

CCM_CLK_PUSM

This sub-block manages the powerup clock and asynchronous chip_reset_n for the CCM and OCOTPC block

Figure 5-2 describes the CCM boundary.

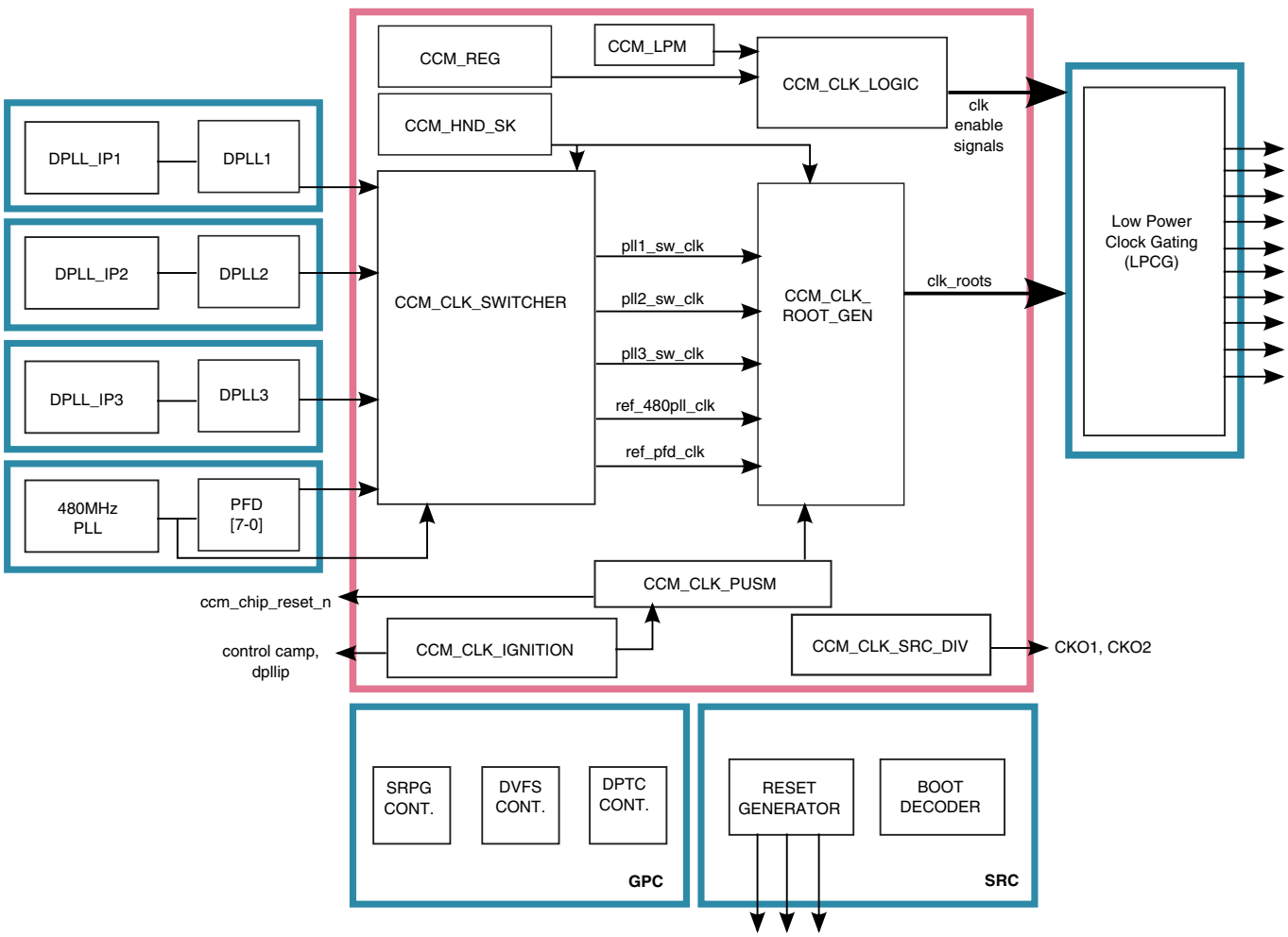


Figure 5-2. Block Diagram

5.2 CCM External Signals Description

Table 5-1 describes the input / output signals of CCM module:

Table 5-1. Signal Descriptions

Signal	I/O	Description
init_ahb_podf	I	SJC PODF values for AHB
init_ipg_podf	I	SJC PODF values for IPG
init_axi_a_podf	I	SJC PODF values for AXI_A

Table continues on the next page...

Table 5-1. Signal Descriptions (continued)

Signal	I/O	Description
init_axi_b_podf	I	SJC PODF values for AXI_B
init_weim_podf	I	SJC PODF values for EIM
init_perclk_pred2	I	SJC PODF values for PERCLK
init_perclk_podf	I	SJC PODF values for PERCLK
init_pll1_sw_sel	I	SJC values for PLL1 mux select
init_periph_clk_sel	I	SJC values for PERIPH mux select
init_ssi_apm_clk_sel	I	SJC values for SSI_APM mux select
osc_clk	I	24MHz clock input
ahb_bypass_clk	I	Bypass clock for ahb root clock
axi_a_bypass	I	Bypass select for AXI A root clock
axi_a_bypass_clk	I	Bypass clock for AXI A root clock
weim_bypass	I	bypass of input of eim clock divider
weim_bypass_clk	I	clock for bypass of input of eim clock divider
shd_current_state [1:0]	O	clock shutdown state machine current state
arm_debug_clk_enable	O	lpcg clkgate enable for debug_apb_clk_root
debug_apb_clk_root	O	DEBUG APB root clock
ecspi_clk_root	O	ECSPI root clock
gpu2d_clk_root	O	GPU2D root clock
display_axi_clk_root	O	root clock
sys_clk_root	O	root clock
gpmi_clk_root	O	root clock
bch_clk_root	O	root clock

5.3 Functional Description

5.3.1 Clock Generation

Clock frequency selection for some domains can be a function of multiple reference clock sources and divide parameters that are set in the CCM control registers. The most extreme case is using a PLL or PFD source, a multiplexer that selects either the osc_clk (24 MHz xtal), PLL or PFD clock as a source to drive the clock divider, and the divide value for the clock divider itself. When programming a selected frequency, the sequence

of events to achieve a given frequency must maintain the integrity of the system as a whole. During a clock system context switch, intermediate clock frequencies for selected domains cannot be faster than the sub system or I/O interface it is designed to support.

5.3.1.1 External Low Frequency Clock—CKIL

The i.MX50 can use either a 32 kHz, 32.768 kHz or a 38.4 kHz crystal as the external low frequency source. Throughout this chapter, the low frequency crystal is referred to as the 32 kHz crystal. CMOS input buffer with Schmitt trigger feature is used as receiver of 32 kHz clock. This clock source should be active all the time the i.MX50 is powered on. The 32 kHz entering the CCM is referred as CKIL. It is synchronized to ipg_clk and supplied to blocks that require it.

5.3.1.2 External High Frequency Clock—CKIH and Internal Oscillator

The i.MX50 uses an internal oscillator (OSC) to generate the reference clock. The internal oscillator will be connected to an external crystal. The oscillator generates frequencies in the range of 24 MHz.

Moreover, i.MX50 uses a CKIH input pin to generate special serial clock demands (like eLCDIF, ePDC, SSI1 and so on) as a backup to the PLL clocks.

5.3.1.3 DPLL Reference Clock

There are three DPLLs in i.MX50 project named:

- PLL1 (typical functional frequency 800 MHz)
- PLL2 (typical functional frequency 400 MHz)
- PLL3 (typical functional frequency 216 MHz)

Each DPLL is controlled by a DPLLC interface block. Each DPLLC uses the 24 MHz osc_clk as a reference.

5.3.1.4 Analog PLL Reference Clock

There is a fixed 480 MHz PLL that feeds eight independent Phase Fractional Dividers (PFDs). The PFDs allow multiple fractional clocks to be generated from one PLL without the trouble of relocking the PLL each time. There is a specific startup sequence that must be performed when powering up the PLL.

1. The TZIC_DSMINT[DSM] bit must be set before powering up the PLL.

2. After the 50us lock counter indicates lock, the DSM bit is then cleared and at least 10us extra time must be waited before using the output of the PLL.

After the fixed 480 MHz PLL has been enabled and locked after 50us counter indicates pll lock, each PFD can be programmed to a target frequency of f (MHz) = (480 × 18) / N; where N = PFD*X*_FRAC with a range from 18 to 35.

The control registers for the fixed 480 MHz PLL and 8 PFDs are located in the analog digital control module. For programming convenience to setup the clocks, the register descriptions for the 480 MHz PLL and 8 PFDs are shown in CCM_ANALOG.

5.3.1.5 CCM Internal Clock Generation

The clock generation is comprised of three submodules:

- CCM_CLK_SWITCHER
- CCM_CLK_IGNITION
- CCM_CLK_ROOT_GEN

5.3.1.5.1 CCM_CLK_SWITCHER

CCM_CLK_SWITCHER sub module receives the pll output clocks and the pll bypass clocks. It generates 3 main switcher clocks to be delivered for CCM_CLK_ROOT_GEN:

- pll1_sw_clk (typical functional frequency 800 MHz—used to supply ARM platform)
- pll2_sw_clk (typical functional frequency 400 MHz—used to supply AXI/AHB/IP buse clocks)
- pll3_sw_clk (typical functional frequency 216 MHz—used to supply serial clocks like USB, SSI, and so on)

Figure 5-3 describes the generation of the three switcher clocks. The figure also includes Frequency Switch Control sub-block that is responsible for frequency change during DVFS scenario.

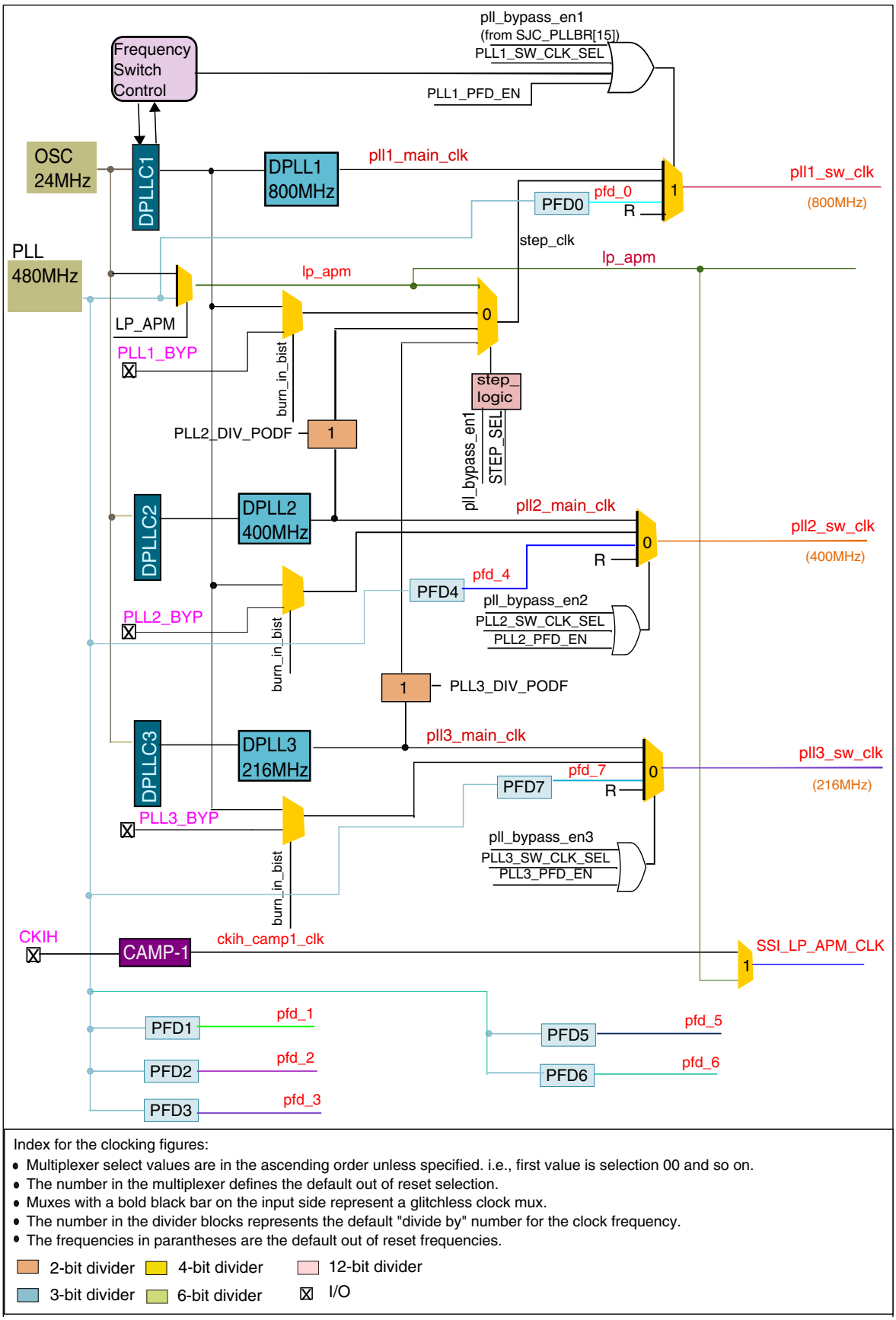


Figure 5-3. Switcher Clock Generation

5.3.1.5.2 PLL Bypass Procedure

CCM_CLK_SWITCHER sub-block includes the capability for each of the pll*x*_main_clk's to be bypassed with an external bypass clock to supply the pll*x*_sw_clk outputs an external bypass clock. In burn_in_bist mode the PLL bypass clocks for all PLLs will be supplied from pll1_ref_clk (the output of DPLLC1).

The decision to shift from pll_main_clk to pll_bypass_clk is done either by programable bits in CCSR or by pll_bypass_en signals which are driven from the jtag module. Since the switch between the clocks are done by a glitch less multiplexer, then both pll_main_clk and pll_bypass_clk should be running to complete the shift without a glitch. In case the selection is programmed in reset by jtag, then both clocks do not need to be active.

There are 4 bypass clock options for pll1_sw_clk. Since the pll1_sw_clk bypass clock is selected from a standard cell mux, changing the CCSR[STEP_SEL] selection should only be done when the glitch less clock multiplexer is selecting the PLL1 reference clock via CCSR[PLL1_SE_CLK_SEL].

5.3.1.5.3 DPLLC Reference Clock Connectivity

For all four DPLLCs:

- The clock2 input port is connected to the 24 MHz osc_clk.
- The input init_dp_ctl_dpllip [8] is connected to gnd.
- The input init_dp_ctl_dpllip [9] is connected to vcc.

The above connections allows DPLLC to choose the 24 MHz osc_clk as the default reference clock to the DPLL. Out of reset, software may select the clock0 input, CKIH, as an alternate reference clock source.

NOTE

clock1 and clock3 are unconnected for the i.MX50.

5.3.1.5.4 PLL Clock Change

The following sequence should be followed when changing the PLL source, frequency or DPLL reference clock for downstream dividers:

1. All active downstream dividers should either be gated off or moved to an alternate PLL source via glitchless clock multiplexers.
2. Reprogram the respective DPLLC:
   a. Configure DPLLC-n_CONFIG[1] to auto restart the PLL on the next update of the reference clock.

b. Configure DPLLC-n_CTL[8:9] to change the reference clock multiplexer.
c. Configure the DPLL divide values.

3. After the DPLLC is reprogrammed, the DPLLC will generate a sticky signal DPLLCx_CTL[LRF] (pll lock) to the CCM. Based on the CCM_CIMR[2:0] settings, the PLL lock may generate a software interrupt.
4. If needed, reprogram downstream dividers to handle the new PLL frequency.
5. Switch downstream dividers back to the updated PLL and if needed, enable the downstream divider clocks.

5.3.1.5.5 CCM_CLK_IGNITION

The responsibility of the ccm_clk_ignition sub-block is to manage the wake up after reset of the on chip oscillator, CAMP1, DPLLC and DPLL blocks. Its task starts with de-assertion of early_reset from the Reset controller (SRC). Its task finishes with assertion of signal that notifies the reset controller that the root clocks are ready.

Figure 5-4 describes the connectivity of ccm_clk_ignition sub module.

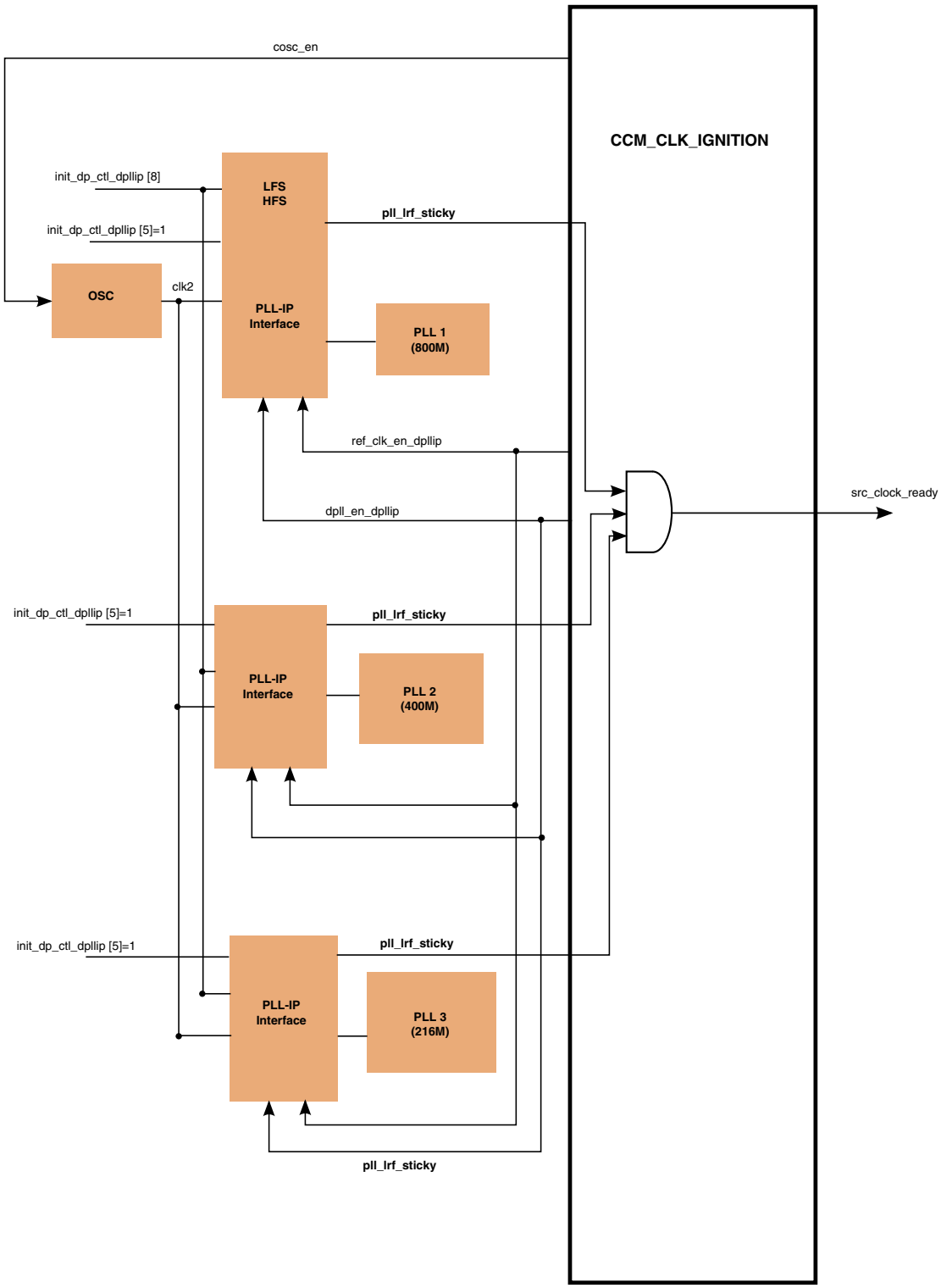


Figure 5-4. Connectivity of ccm_clk_ignition Sub Module

The 24 MHz oscillator will be used as the default reference for all of the system PLLs in the design.

Out of reset, the CCM_CCR[COSC_EN] signal will be asserted and CCM_CLPCR[COSC_POWERDOWN] will remain deasserted to start the on board oscillator. The external oscillator and camps will also be enabled during the ignition process: CCM_CCR[CAMP1_EN] = 1 and CCM_CSR[REF_EN_B] = 0 so that the ckih_camp1 clock will be started. Next ccm_clk_ignition will count the number of 32 kHz CKIL cycles defined by CCM_CCR[OSCNT] bit field. Once the counter has elapsed, it is assumed that the 24 MHz osc_clk and the ckih_camp1 clock are both ready. In this stage CCM will assert the CCM_CSR[COSC_READY] and CCM_CSR[CAMP1_READY] bits. At this time, the ccm_clk_ignition will also assert the start_pusm bit to the powerup state machine in the CCM.

With the PLL's disabled out of reset, the ccm_clk_ignition will generate the src_clock_ready signal after the assertion of COSC_READY to indicate to the SRC module that the oscillator clocks are ready.

Upon arrival of src_clock_ready signal to reset controller, the reset sequence will continue. In parallel with the memory repair mode of the SRC, the CCM's clk_pusm sub-block will execute it's startup sequence, see CCM_CLK_PUSM Reset Sequence. Following the completion of the clk_pusm sequence the SRC reset sequence will run to completion. Review SRC spec for details on this sequence.

5.3.1.5.6 CCM_CLK_PUSM Reset Sequence

A unique aspect is that the system reset clock is shared with the system scan clock. All registers throughout the design see a clock at 6 MHz during power up and system reset. A state sequence begins when the start_pusm bit goes active. The state sequence is as follows:

1. chip_reset_n deassert, functional clock select disabled to select system wide synchronous 6 MHz clock. Functional clock sources are gated off.
2. Propagate 32 power-up clock cycles to propagate reset through the system.
3. Disable the 6 MHz system reset clock generator.
4. Wait # number of xtal clock cycles.
5. De-assert the asynchronous reset, chip_reset_n
6. Wait # number of xtal clock cycles
7. With the reset clock low, reset inactive and functional clock sources gated off (logic low0, switch the multiplexor to select functional clock sources (which are still gated low also).
8. Wait # number of xtal clock cycles.

9. De-assert the clock gate which allows the functional clocks to start at the selected frequency.
10. Continue SRC reset sequence.

5.3.1.5.7 Reset Values for DPLLC

Reset values which are hard coded into the dpll-ip are:

- PLL1—initial value = 208 MHz
- PLL2—initial value = 208 MHz
- PLL3—initial value = 182 Mhz

These frequencies correspond to reference clock source of the 24 MHz on chip oscillator as the source of the DPLL.

5.3.1.5.8 Analog 480 PLL and PFD

The CCM has the ability to use the analog fixed 480 MHz PLL and eight independently programmable PFDs as reference clock sources. The frequency of the PFD can be calculated as follows:

- Freq = (480 × 18) / N; where N = 18 – 35.

5.3.1.5.9 CCM_CLK_ROOT_GEN

CCM_CLK_ROOT_GEN sub-block generates the root clocks to be delivered to LPCG.

The following is a list of the root clocks generated by this module. The clocks are supposed to be asynchronous unless mentioned differently:

- ARM_CLK_ROOT — Generated from pll1_sw_clk.
- WEIM_CLK_ROOT
- AHB_CLK_ROOT
- IPG_CLK_ROOT — Divide by 2 of the AHB_CLK_ROOT. This clock is edge-aligned and balanced with the AHB_CLK_ROOT.
- PERCLK_ROOT — This clock is synchronized and balanced to AHB_CLK_ROOT. For the synchronization process , perclk predivider and podf should generate a clock frequency 2.5 times slower than the AHB clock. In peripheral DVFS scenario, when AHB's frequency is reduced, perclk frequency should be configured to be remain 2.5 times slower than the minimum value of ahb_clk (the DVFS value of ahb_clk).
- PGC_CLK_ROOT—Generated as a division of ipg_clk and balanced to it.
- CKIL_SYNC_CLK_ROOT—When not in stop mode, the ckil clock is synchronized to the IPG_CLK_ROOT and balanced with it. In stop mode, the synchronizer is bypassed and ckil_sync_clk_root is driven by the raw ckil clock.

- DDR_CLK
- ESDHC1_CLK_ROOT
- ESDHC2_CLK_ROOT
- ESDHC3_CLK_ROOT
- UART_CLK_ROOT
- SSI1_CLK_ROOT
- SSI2_CLK_ROOT
- SSI_EXT1_CLK—Connected to external pad
- SSI_EXT2_CLK—Connected to external pad
- USB_PHY_CLK_ROOT
- ECSPI_CLK_ROOT
- WRCK_CLK_ROOT
- EPDC_PIX_CLK_ROOT
- ELCDIF_PIX_CLK_ROOT
- EPDC_AXI_CLK_ROOT
- DISPLAY_AXI_CLK_ROOT
- SYS_CLK
- GPMI_CLK_ROOT

Figure 5-5 and Figure 5-6 describe the above clocks generation. The frequencies in parentheses are the default typical frequencies.

NOTE

Muxes with a bold black bar on the input side represent a glitchless clock multiplexer.

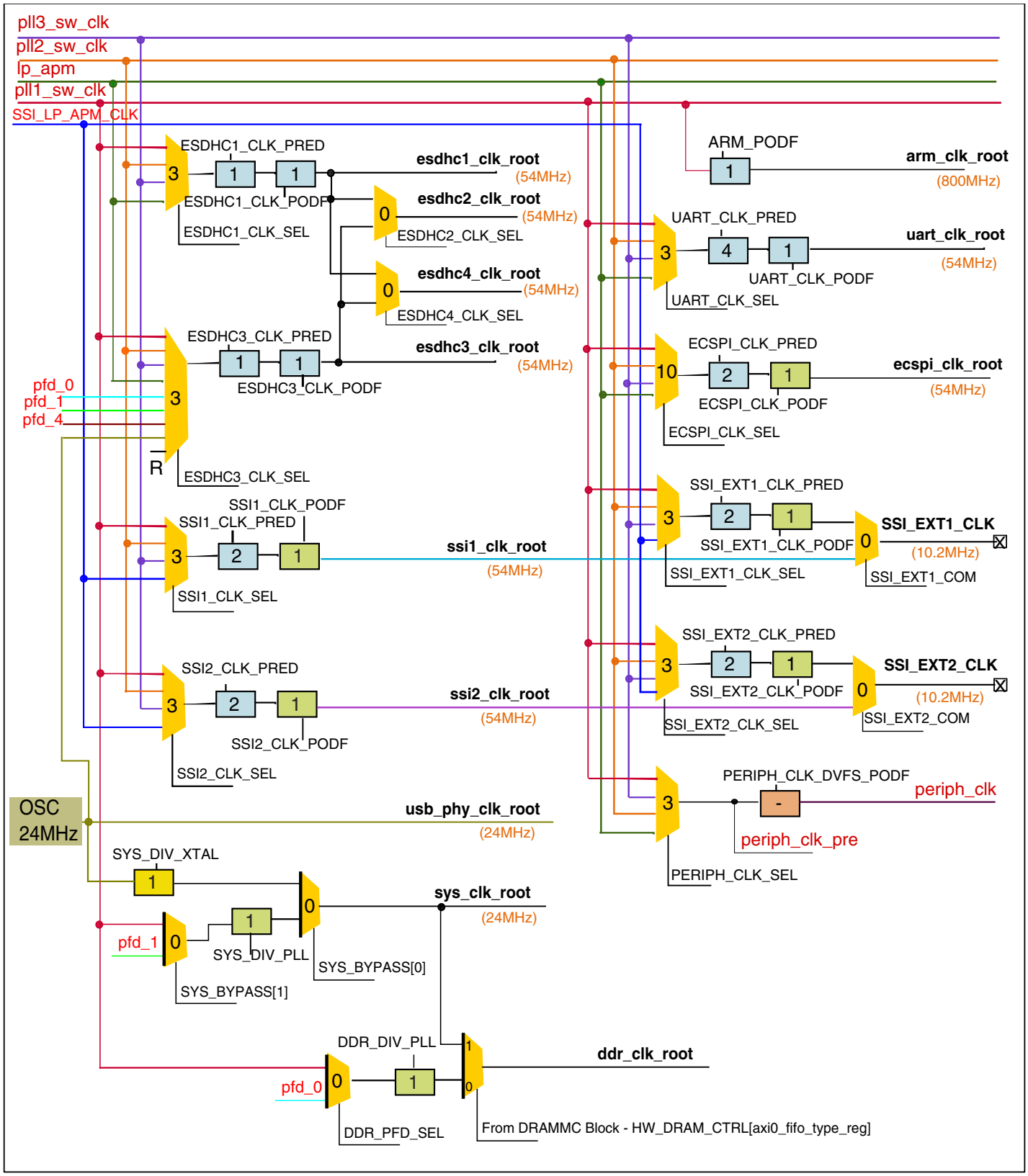


Figure 5-5. Root Clock Generation

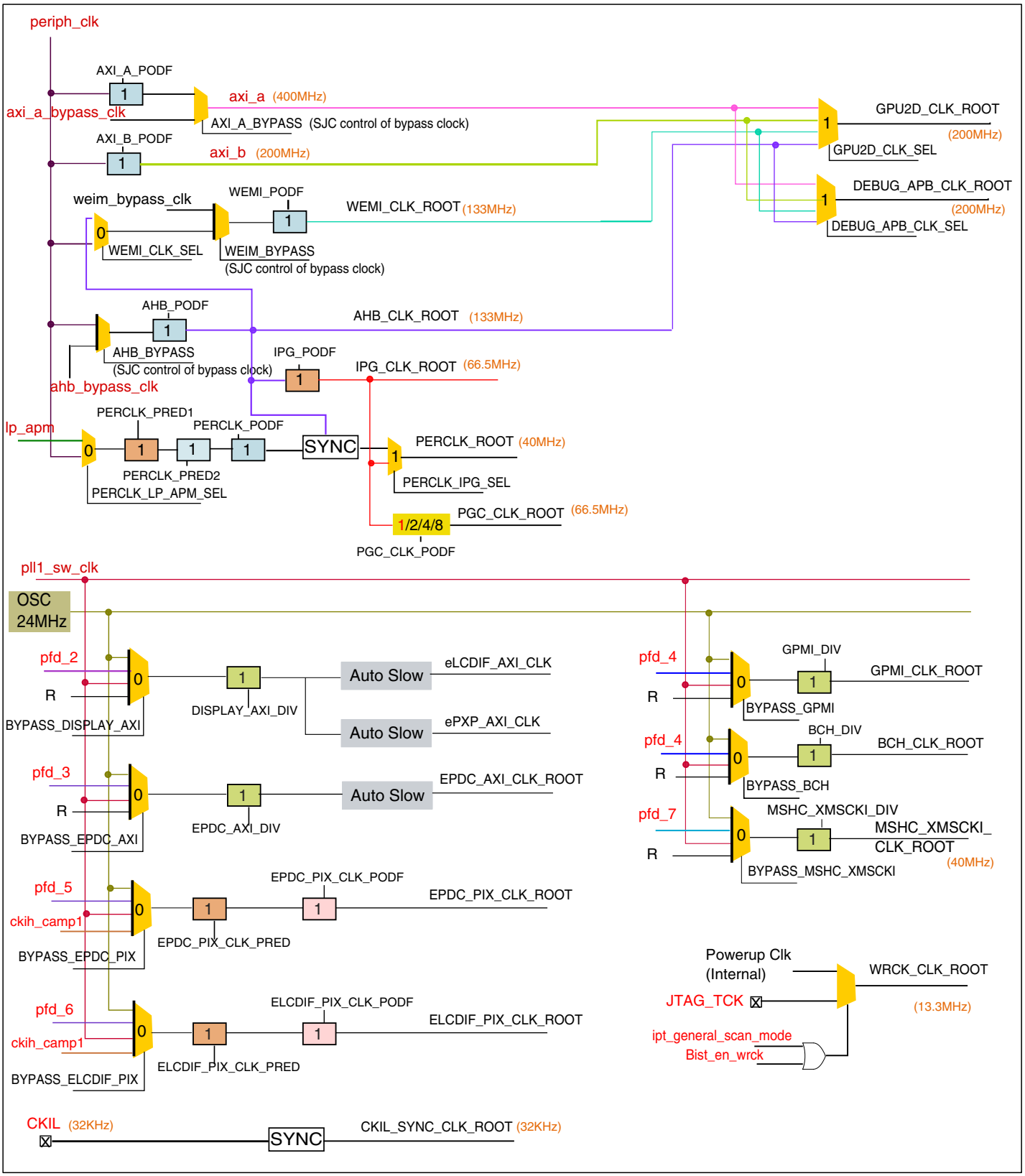


Figure 5-6. Root Clock Generation (cont)

NOTE

The following pre dividers should use divider values larger than '1'. The option '000' is not allowed for them: SSI1_CLK_PRED, SSI2_CLK_PRED, SSI_EXT1_CLK_PRED, SSI_EXT2_CLK_PRED, ECSPI_CLK_PRED.

5.3.1.5.10 Initial Reset Values Controlled by SJC

The control register reset values of the following dividers and muxes are controlled by the jtag module (SJC). These reset values are programmable. In normal operation, the SJC will drive the reset values stated in the CCM register memory map. If the SJC is programmed to change those values, then the "programmed" values will be applied to the control registers during system reset. Table 5-2 shows the control signals and the dividers/muxes.

Table 5-2. SJC Init Control

init Signal	Control Reset Value for Divider or mux
init_perclk_pred[1:0]	perclk_pred1 divider
init_perclk_pred2[2:0]	perclk_pred2 divider
init_perclk_podf[2:0]	perclk_podf divider
init_ipg_podf[1:0]	ipg_podf divider
init_ahb_podf[2:0]	ahb_podf divider
init_axi_a_podf[2:0]	axi_a_podf divider
init_axi_b_podf[2:0]	axi_b_podf divider
init_weim_podf[2:0]	weim_podf divider
init_pll1_sw_sel	pll1_sw_sel mux
init_periph_clk_sel[1:0]	periph_clk_sel mux
init_ssi_apm_clk_sel[1:0]	ssi_apm_clk_sel mux

5.3.1.5.11 Divider Change Handshake

For the i.MX50, the hardware modules the CCM handshaked with on the previous i.MX designs were removed. To maintain compatability with the previous designs, the state machine has been retained, but the acknowledge from the modules have been replaced with a single cycle wait state.

Divider	Handshake with Module	Comment
weim_podf	—	NA

Table continues on the next page...

Divider	Handshake with Module	Comment
axi_a_podf	—	NA
axi_b_podf	—	NA
ahb_podf	—	NA

Once the CCM_CBCDR register is updated, CCM will check which of the above dividers was updated. For the updated dividers, CCM will perform the frequency change. The signal, periph_dvfs_sw_ack, is held for two clock cycles of ipg_clk on each divider change.

Since there are no blocks to handshake with, a write to the respective divider will commence immediately after the CCM_CBCDR register is updated.

Interrupts can be generated for each of the above divider changes. Refer to CCM_CISR register for details on those interrupts.

In addition, the CCM has divider handshake in process register (CCM_CDHIPR). This register has status bits for each of the above dividers that are involved in the handshake process. When the respective bit is asserted, it means that the divider is being updated. During the update, software should not attempt to write to this divider until the respective CCM_CDHIPR bit is de-asserted. Any reads of the divide value during the update will reflect the new value being applied to the divider. To guarantee that software reads the actual dividers value, reads should be performed after the respective CCM_CDHIPR bit is de-sasserted.

NOTE

When DVFS is enabled (through GPC), software should not "manually" change the control register divide value because this might corrupt the handshake process. The frequency of the system should be setup prior to enabling the DVFS.

Dividers without hardware handshake:

For dividers with non-glitchless clock muxes, software needs to ensure that and downstream clocks are gated off before changing multiplexer settings because there is no hardware handshake for changing multiplexer settings to prevent glitching the clock.

Likewise, the multiplexers of the AXI buses that are controlled by CCM_CBCDR register are not glitchless clock muxes, the reference clock should not be switched on the fly. Software should make sure that any downstream clock branches are gated off before changing the multiplexer configuration.

5.3.1.5.12 CKIL Synchronizing to ipg_clk

CKIL is synchronized to ipg_clk when system is in functional mode. When system is in stop mode, that is, when there is no ipg_clk, the CKIL synchronizer will be bypassed, and raw ckil will be supplied to the system.

5.3.1.6 DPLL's Disabling/Enabling

PLL disabling and enabling is done via DPLLC block. Software should first move all the clocks generated from a specific PLL to another PLL, before disabling the PLL through DPLLC. This move of clocks can be done via glitch less multiplexer for clocks which are critical to the system, that is, bus clocks. For serial clocks, software should first disable the downstream modules and gate off the respective clock branch. Next, switch the mux controlling the source of the clocks to another PLL, and finally start back the module and its clocks. Only when the PLL is not being used as a reference clock source is it safe to disable the PLL.

5.3.1.7 Observability Output Signals

CCM has three mux's to generate critical signals to the IO PADS for observability. The three MUX outputs are connected to the three CCM outputs named obs_output_0, obs_output_1, obs_output_2. The three MUX's are controlled via CCM_CTOR register.

It is possible to generate also the obs_input_0, obs_input_1, obs_input_2, obs_input_3, obs_input_4 and obs_input_5 to the external pads through the MUX's controlled by CCM_CTOR. This allows to observe the signals connected to those 6 CCM input pins. Below list describes the signals connected for observability to those input pins:

- obs_input_0 connect to DPLLC1 : dpllip_cpen
- obs_input_1 connect to DPLLC1 : dpllip_cpres
- obs_input_2 connect to DPLLC1 : dpllip_crstrt
- obs_input_3 connect to DPLLC1 : dpllip_load_req
- obs_input_4 connect to DPLLC2 : dpllip_cpen
- obs_input_5 connect to DPLLC3 : dpllip_cpen

5.3.1.8 Low Power Clock Gating Module (LPCG)

The LPCG sub-block receives the root clocks and splits them to clock branches for each block. The clock branches are gated clocks. The enables for those gates can come from six sources:

1. Clock enable signal from CCM—this signal is generated by configuration of the CCGR bits in CCM control register map.
2. Clock enable signal from the module—this signal is generated by the module based on internal logic of the module. For clock enable signals from the module that are used, the CCM will generate an override signal based on a CCM CMEOR control register bit.
3. Clock enable signal from Reset controller (SRC)—this signal will enable the clock during the reset procedure. Please refer to SRC spec for details on the clock enable signal during reset procedure.
4. Hard coded enable from fuse box.
5. Enable or disable clock on bist mode, based on bist_en signal. This will be applicable for clocks that are not functional and needed only on memory repair or bist sequence (like sms_clk's and wrck).
6. Auto-slow mode enable from CCM.

The enable sources are AND'ed to generate the enable signal for the gating cell.

In order to prevent glitches on the gated clock, the enable signal for the gating cell is synchronized with the clock it needs to gate.

Feedback signals are generated for the CCM, to indicate when to gate on/off the clock roots. All feedback signals that correspond to the same clock root will be OR'ed to generate one notification signal to CCM for clock root gating.

Figure 5-7 describes the implementation for each gating cell.

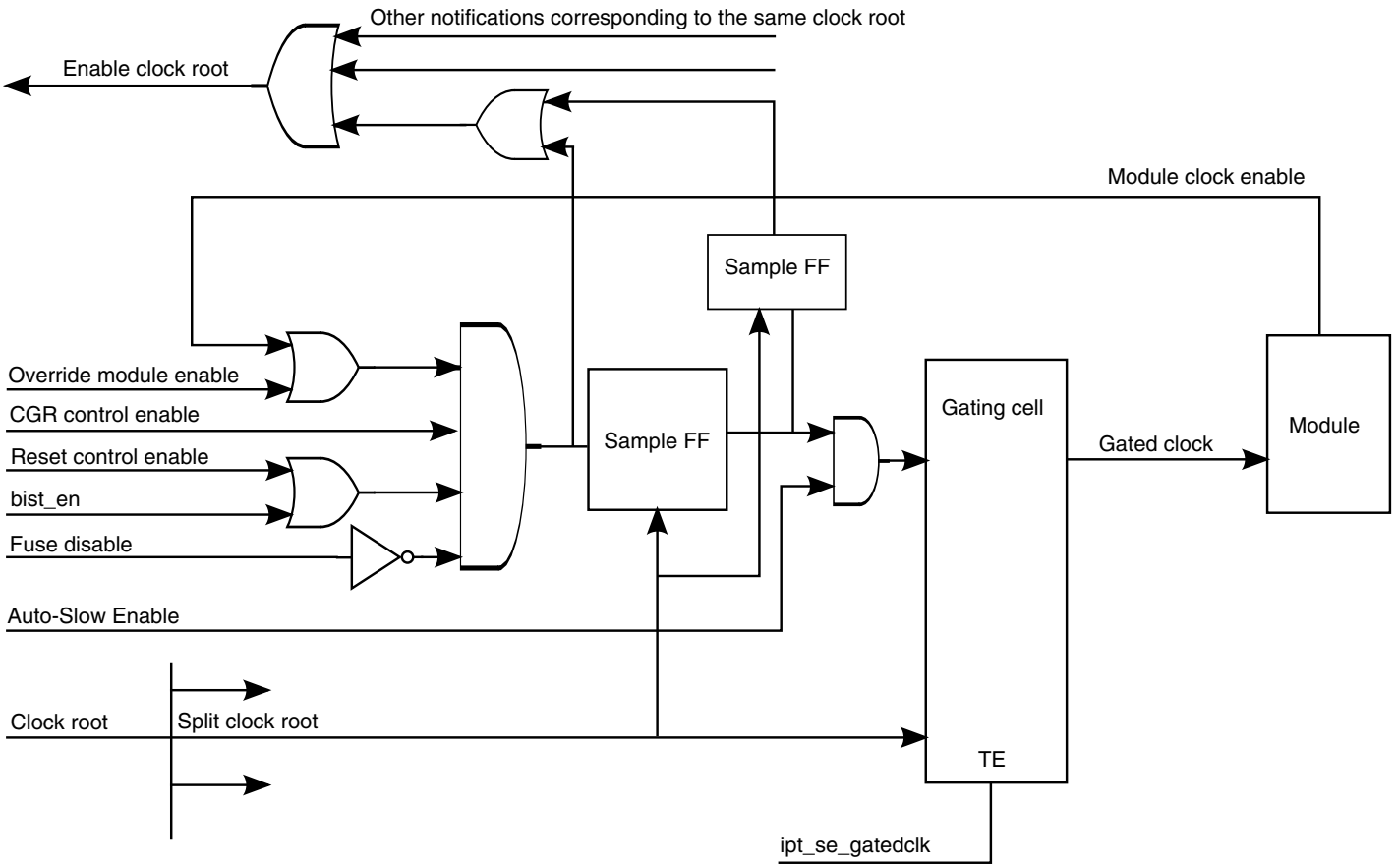


Figure 5-7. Implementaion of Gating Cell

Figure 5-8 describes the clock split inside LPCG sub-block. It describes the case of 2 blocks, one block without an enable signal and one with an enable signal. (SRC enable signals and sync FF's are omitted from this figure).

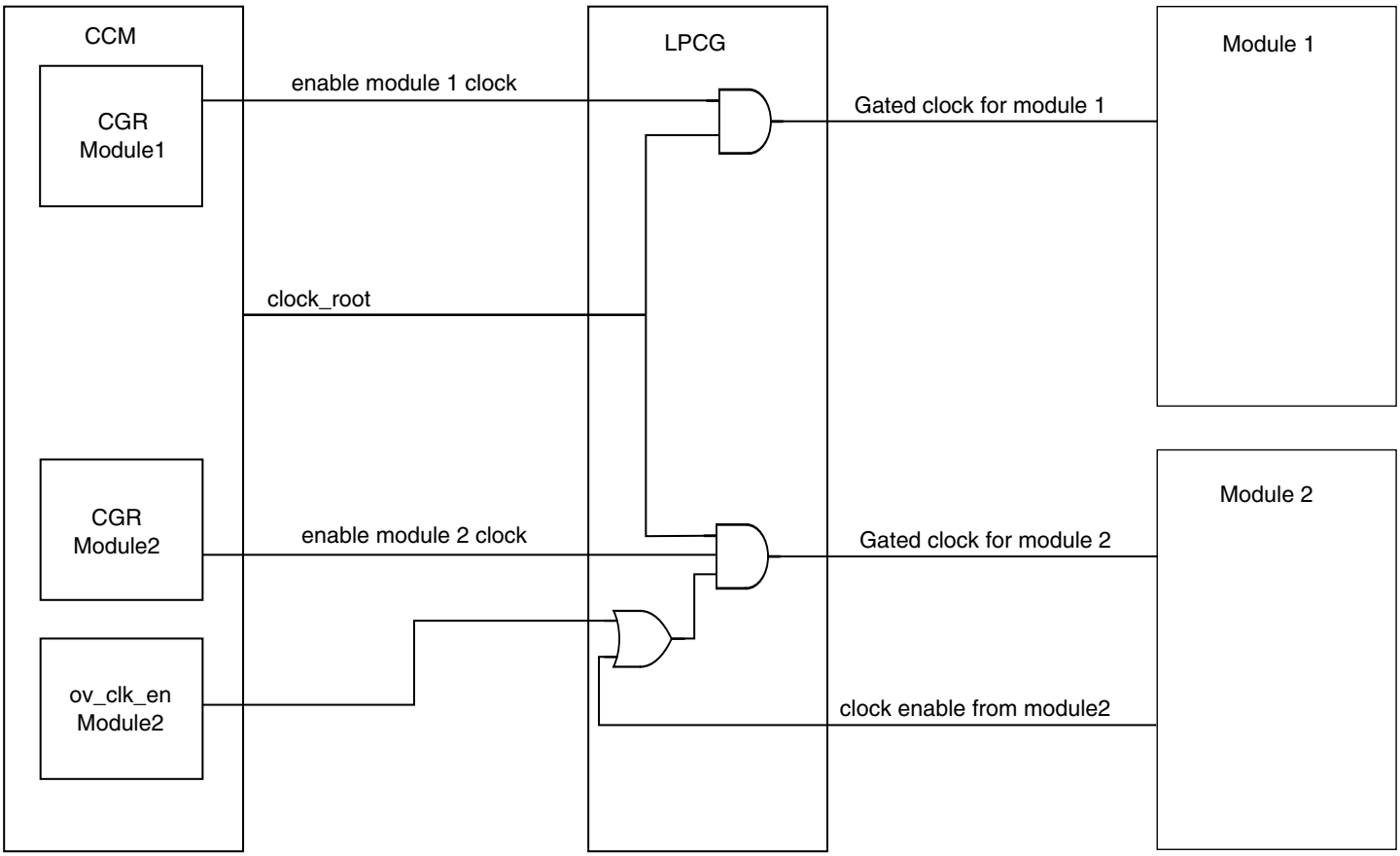


Figure 5-8. Clock Split in LPCG

5.3.2 DVFS Support

Dynamic frequency changing in the i.MX50 can be done on two domains:

1. ARM clock domain frequency change
2. Peripherals clock domain frequency change (including buses)

The General Power Controller (GPC) will initiate the DVFS procedure. It will initiate only one frequency change at a time. There is not an option to commence both of the DVFS domains simultaneously. The GPC will be aided by SDMA and CCM to execute the frequency change.

5.3.2.1 ARM Clock Domain Frequency Change

The SDMA can request an ARM frequency change by dividers, or by PLL1 re-lock.

The following steps will be configured by SDMA:

1. SDMA can request ARM frequency change by dividers, or by PLL1 relock. This request will be written to CCM_CDCR[ARM_FREQ_SHIFT_DIVIDER] bit:
   a. ARM_FREQ_SHIFT_DIVIDER = 0: PLL1 relock shift is requested.
   b. ARM_FREQ_SHIFT_DIVIDER = 1: ARM_PODF divider shift is requested. In this case any new writes to ARM_PODF will be held until arm_clk_switch_req signal is asserted. The CCM holds a status bit to notify that the arm_podf divider being updated. Software should wait until the write finishes before writing any new value to arm_podf. Pease refer to CCM_CDHIPR register for details on the status bit. The CCM can also generate an interrupt once the actual arm_podf has been written in the DVFS scenario. Please refer to CCM_CISR register for details.

      SDMA will continue with next steps only if (ARM_FREQ_SHIFT_DIVIDER = 0: PLL1 relock shift is requested).
2. Check which frequency mode is active in DPLLIP1, HFS or LFS: DPLLIP bit HFSM of DP_CTL register.
3. Configure the non active frequency mode (HFS or LFS) of DPLLIP1, with the new frequency needed for PLL1 to relock on: DPLLIP1 registers DP_OP, DP_MFD, DP_MFN, DP_HFS_OP, DP_HFS_MFD, DP_HFS_MFN.
4. Configure the step frequency divider: CCM CCSR register, bit pll2_div_podf or pll3_div_podf (depending on the decision in the step mux in step 5).
5. Configure the step frequency mux: CCM bit step_sel of CCM_CNT register.

Once the configuration is completed, the SDMA will notify the GPC to continue with the frequency change procedure.

The actual frequency change will start by the assertion of the arm_clk_switch_req signal from the GPC to the CCM.

The CCM will start the following steps when the arm_clk_switch_req signal is asserted:

If ARM_FREQ_SHIFT_DIVIDER = 1 dividers shift is requested):

1. Load the ARM domain new divider values.
2. After loading the divider (all counters in 0), assert clk_switch_ack for 2 ipg cycles as an acknowledge to the GPC that the frequency has been updated.

If ARM_FREQ_SHIFT_DIVIDER = 0 (PLL1 relock shift is requested):

1. Switch to the step frequency by a glitchless clock mux.
2. After the glitchless clock mux has changed to the step frequency, invert pll_lvs signal to DPLLC-1 so that PLL1 will lock on the new frequency.
3. When the pll_lrf_sticky1 is asserted switch back to DPLL1 with the glitchless mux.

4. After the glitchless clock mux has switched, assert clk_switch_ack for 2 ipg cycle as an acknowledge to the GPC that frequency has been updated.

When the clk_switch_ack is asserted, the GPC will de-assert the arm_clk_switch_req signals.

At this point, the ARM has changed frequency and system is ready for a new frequency switch procedure.

Figure 5-9 shows the ARM DVFS Connectivity.

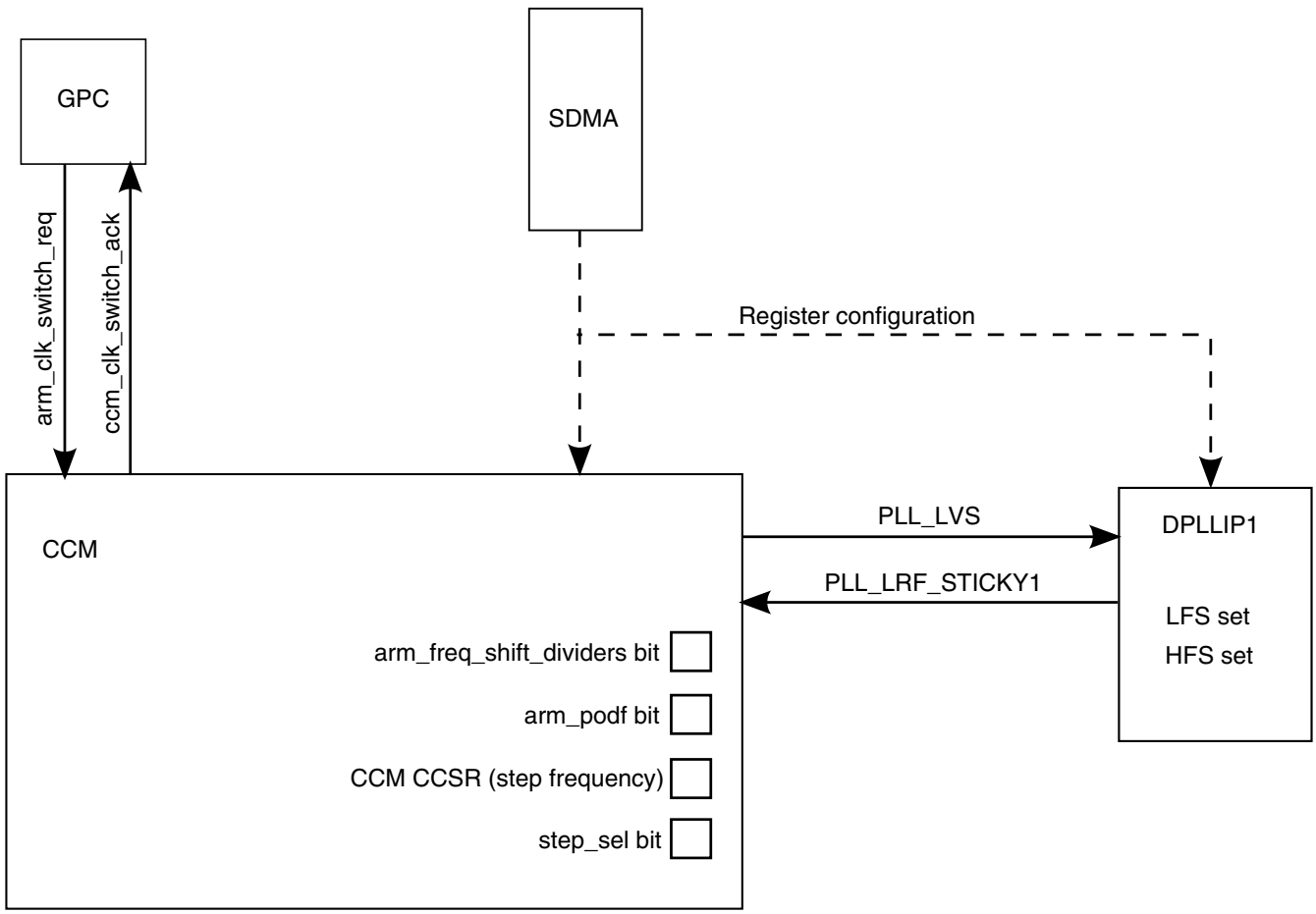


Figure 5-9. ARM DVFS Connectivity

5.3.2.2 Peripheral Clock Domain Frequency Change

The CCM includes the capability to slow the peripheral bus clock by divide down by 2/3/4. This will affect the root of bus clocks: ahb, axi, ipg, weim_clk. The actual division factor is defined by bits in CCM_CDCR[1:0]. The software is expected to set those bits

prior to enabling DVFS operation. The value written to the DVFS register divider (CCM_CDCR[1:0]) will not be loaded immediately to the divider, but will wait until the peripheral DVFS procedure commences.

While in the DVFS scenario, the root clocks which are not affected by DVFS divider should be generated from a low frequency pll. This frequency should not be higher than 240 MHz. Software can use pll3 with a frequency of 216 MHz for this operation. Note: Prior to the DVFS clock change, some modules with a root clock that will be slowed down, may need to be re-configured with new control register values for proper operation.

Divide by 2/3/4 procedure (lowering frequency):

1. In the CCM control register, write the new peripheral divide value.
2. Enable peripheral DVFS.
3. The GPC will assert the periph_clk_div_req signal to the CCM.
4. The CCM will switch to the new peripheral divide value as long as the GPC asserts the periph_clk_div_req.
5. When the GPC de-asserts the periph_clk_div_req, the periph divider will switch back to divide by

NOTE

The DVFS operation can also start by software controlling the sw_periph_clk_div_req. In this case, the GPC control will be ignored and the ccm_clk_switch_ack will not be asserted.

There is no need to synchronize the 4 dividers for the DVFS change since they work on asynchronous clock domains.

5.3.2.3 Peripherals Restrictions in DVFS Scenario

In the DVFS scenario for peripherals, ipg and ahb clocks are divided from their nominal frequency.

The serial dividers in the low voltage case, should use a low frequency PLL input (below 300 MHz) to work correctly.

Table 5-4 describes restrictions that need to be followed to support the DVFS frequency shift:

Table 5-4. DVFS Restrictions

Module	Support	Restriction
R—Modules that will work with restriction		

Table continues on the next page...

Table 5-4. DVFS Restrictions (continued)

Module	Support	Restriction
UART	R	Need to configure uart rate to be low_frequency_ipg_clk/16. Hence rates of 1.875 M and 4 M cannot be reached in DVFS scenario.
ESDHC	R	Need to configure the AHB clock to be faster than the SD Card clock.
SSI	R	Master clock should be generated from ssi_ext1 and ssi_ext2 pads.
CSPI	R	Need to configure spi rate to be low_ipg_clk/4.
N—Modules that will not work		
USB_PHY	N	Needs ipg_clk>60 MHz. Hence will not work under DVFS scenario
FEC	N	Needs ipg_clk>50 MHz. Hence will not work under DVFS scenario

5.3.3 Auto-Slow Mode

The output of the Auto-Slow Mode logic will be referred to as the clk_asm clock domain. The reference clock will be referred to as the int_ref_clk. The clk_asm branches of the int_ref_clk domain is gated by an instantiated ICG that uses a counter in the int_ref_clk domian to enable/disable the clk_asm branch based on a divide value programmed in the clk_asm fast divide register. This is the mechanism that provides the desired frequency on the clk_asm domain with respect to the int_ref_clk domain.

A dynamic power management mechanism that further divides the clk_asm branch is also provided and throttled based on system activity. An asm_enable value of 0 disables this feature and the clk_asm frequency will be as programmed in the clk_asm divide register. When the asm_enable register is not 0, then this feature is enabled.

When a system component is busy and needs full throughput on the clk_asm branch, its busy signal is asserted. The clk_asm branch will then assume the clock rate indicated by the clk_asm divide register. When the system component is not busy, it will deactivate its busy signal.

5.3.4 Power Modes

The i.MX50 supports 3 low power modes: RUN mode, WAIT mode and STOP mode.

5.3.4.1 Run Mode

This is the normal/functional operating mode. In this mode the ARM runs in its normal operational mode. The frequency and voltage can be changed upon dvfs scenario as described in DVFS Support. Clocks to the modules can be gated by configuring the corresponding CCM_CCGR bits.

5.3.4.2 Wait Mode

In this mode the ARM clock is gated. All other clocks are functional and can be gated by programing their CCGR bits. Power gating can be done on the ARM platform.

5.3.4.2.1 Wait Mode Procedure

The wait mode procedure is as follows:

1. The ARM will write to CCM_CLPCR[LPM] bit to set it for WAIT mode.
2. The ARM will write to DSM_INT_HOLDOFF bit in TZIC.
3. The TZIC_DSMINT[DSM] (DSM Interrupt Hold off) assertion forces the interrupt controller to stop the synchronizer clock for incoming interrupts. Any interrupts that had previously been synchronized by the interrupt controller will remain asserted.
4. If a wakeup event occurred or an interrupt was serviced or is pending, the ARM platform may abort the DSM sequence by exiting the main shutdown code sequence and returning to the desired security mode.
5. If not the software should execute the Wait For Interrupt (WFI) instruction.
6. Upon execution, the WFI instruction will cause the ARM to drain its write buffers and enter a quiescent state. At this point the ARM will assert the STANDBYWFI signal.
7. The ARM platform will then wait until the L2 cache controller has become inactive. Once the L2 is inactive and the ARM platform's STANDBYWFI output is asserted, the ARM platform will assert the ARM_DSM_REQUEST signal to the CCM.
8. The CCM will continuously synchronize both the ARM_DSM_REQUEST and the TZIC_DSM_WAKEUP inputs into the clock domain used by its internal state machine.
9. If the TZIC_DSM_WAKEUP is negated and the ARM_DSM_REQUEST is asserted, the CCM will begin the WAIT shutdown sequence:
10. The CCM will stop the ARM clock. (soc_mxclk input to the ARM will be gated only if the respective CCM_CCGR defines to gate it in WAIT mode).
11. Stop of the ARM clock will take place only if CCM_CLPCR[5]=1 and debug_arm_clk_off_on_lpm=0 (connected to sjc register SJC_GPUCR3[15]). If either CLPCR[5]=0 or debug_arm_clk_off_on_lpm=1 then the ARM clocks will not be gated off and the CCM will continue to next step.

12. The CCM will generate the ccm_ipg_wait to indicate that it has started the wait procedure.
13. The CCM will request an acknowledge to close clocks of SDMA, RNGP and AHBMAX if their CCM_CCGR bits indicate to close their clocks in wait mode and if their clocks were not already closed in run mode. The request will be issued if the handshake is not bypassed by programing the CCM_CLPCR[23:16]. If the corresponding bits are set, then the request signal will not be issued to the corresponding module and the CCM will not wait for its acknowledge in the process of entering low power mode. The requests will commence via the following signals:
    - SDMA,RNGB: sdma_ipg_stop_req and rngb_ipg_stop_req
    - EIM: weim_lpmd - this request will be asserted based on the definition of CCM_CCGRs.

NOTE

Since EIM needs the aclk_fast (eim clock) to be running for it to answer with weim_lpmd, then software should make sure to turn on the aclk_fast (weim_clk) before entering any low power mode that requests the weim_lpmd.

    - AHBMAX: ahbmax_halt_req
    - Once the above modules finish their operation and are ready to enter low power mode, they acknowledge the CCM that its safe to turn of their clocks. The acknowledge signals are as follows:
    - SDMA,RNGB: sdma_ipg_stop_ack and rngb_ipg_stop_ack
    - EIM: weim_lpack
    - AHBMAX: ahbmax_halt_ack
14. The requests will be generated by the CCM in stages. CCM will continue from stage to stage once all the acknowledges of a present stages were received. The stages are the following:
    - Stage 1: SDMA
    - Stage 2: AHBMAX
15. Once the CCM receives all the acknowledge signals needed, and if at least 8 ipg_clk cycles have elapsed since the assertion of ccm_ipg_wait signal, then it will enter wait mode and do the following:
    - Close the clocks to the blocks which were defined to be shut at WAIT mode in the CCM_CCGR bits.
    - Assert system_in_wait_mode signal. This signal is connected to the IO pads for observability, to indicate wait mode is active.

Once the CCM is in WAIT mode, it will check if the TZIC_DSM_WAKEUP signal has been asserted or if the ARM_DSM_REQUEST has been negated during the process of entering wait mode. If this has occurred, then the CCM will exit wait mode.

If the TZIC_DSM_WAKEUP is not asserted and ARM_DSM_REQUEST is still asserted, then the CCM will move to the state SRPG_ARM and generate a request to the GPC to power down the ARM platform and GPU2D by asserting signals ccm_pdn_4arm_req and ccm_pdn_4all_req. If those modules are defined to be powered down on wait mode in the GPC, then the GPC will commence powering them down. Please refer to the Figure 5-10 diagram. The GPC will send a pdn_ack at the end of the power down sequence.

The CCM's low power state machine will remain in the state "SRPG_ARM" until the wait mode is exited.

5.3.4.2.2 Exiting Wait Mode

As soon as the synchronized TZIC_DSM_WAKEUP signal is asserted or the ARM_DSM_REQUEST is negated, the CCM will begin the process of exiting WAIT mode.

1. Enable GPU2D clocks only.
2. CCM will request the GPC to restore power. The CCM will assert the ccm_pup_req to request to the GPC to power up the ARM platform if it was powered down on the entrance to wait mode.
3. The GPC will notify the CCM by asserting the signal PUP_ACK to indicate power to the ARM is restored, and its safe to exit from WAIT mode. Only then will the CCM do the following:
4. Once the SRC has asserted the src_power_gating_reset_done to indicate completion of reset to the power gated modules, negate the low power request signals to all modules and enable all modules clocks including the ARM clocks and return to run mode based on their CCGR values.
5. Once the system is in run mode, the CCM will negate the ccm_ipg_wait and system_in_wait_mode.
6. Once the interrupt propagates through all the synchronization logic, the ARM platform will recognize and service it. This will force the negation of the ARM_DSM_REQUEST output. Prior to this point, both the ARM_DSM_REQUEST and TZIC_DSM_WAKEUP were asserted. Since the TZIC_DSM_WAKEUP has top priority, the system is able to wake up.

Before the ARM exits the ISR, it will clear the interrupt source. This will cause the TZIC_DSM_WAKEUP signal to be negated. At this point, the two low power control signals (TZIC_DSM_WAKEUP and ARM_DSM_REQUEST) are negated, and the WAIT sequence can be repeated at any time.

5.3.4.3 Stop Mode

In this mode all system clocks are stopped, the PLL's are stopped and power gating can be done on the ARM platform.

5.3.4.3.1 Entering Stop Mode

The Stop mode is entered by the following procedure:

1. The ARM will write to the CCM_CLPCR[LPM] bit to set it for stop mode.
2. The ARM will write to the TZIC_DSMINT[DSM] (DSM interrupt hold off) bit.
3. The DSM assertion forces the interrupt controller to stop the synchronizer clock for incoming interrupts. Any interrupts that had previously been synchronized by the interrupt controller will remain asserted.
4. If a wakeup event occurred or an interrupt was serviced or is pending, the ARM platform may abort the DSM sequence by exiting the main shutdown code sequence and returning to the desired security mode. If not the software should execute the Wait For Interrupt (WFI) instruction.
5. Upon execution, the WFI instruction will cause the ARM to drain its write buffers and enter a quiescent state. At this point the ARM will assert the STANDBYWFI signal.
6. The ARM platform will then wait until the L2 cache controller has become inactive. Once the L2 is inactive and the ARM platform's STANDBYWFI output is asserted, the ARM platform will assert the ARM_DSM_REQUEST signal to the CCM.
7. The CCM will continuously synchronize both the ARM_DSM_REQUEST and the TZIC_DSM_WAKEUP inputs into the clock domain used by its internal state machine.The only time those signals will not be synchronized is when the pll's will be disabled. In this case exiting from stop mode will be done asynchronously (please refer to the section below describing the exit from stop mode).

If the TZIC_DSM_WAKEUP is negated and the ARM_DSM_REQUEST is asserted, the CCM will begin the STOP shutdown sequence:

1. The CCM will stop the ARM clock.
2. The CCM will generate the ccm_ipg_stop to indicate that it has started the stop procedure.
3. The CCM will request an acknowledge to gate off the SDMA, RNGB and AHBMAX clocks if their clocks were not already gated off in run mode. The stop request to these modules will be issued if the handshake is not bypassed by programing the CCM_CLPCR[23:16]. If the corresponding bits are set, then the request signal will not be issued to the corresponding module and CCM will not wait for its acknowledge in the process of entering low power mode. The requests will commence via the following signals:

- SDMA,RNGB: sdma_ipg_stop_req and rngb_ipg_stop_req
- EIM: weim_lpmd

NOTE

Since EIM needs the aclk_fast (weim clock) to be running for it to answer with weim_lpmd, then software should make sure to turn on the aclk_fast (weim_clk) before entering any low power mode that requests the weim_lpmd.

- AHBMAX: ahbmax_halt_req
- Once the above modules finished their operation and are ready to enter low power mode, they acknowledge CCM that its safe to turn of their clocks. The acknowledge signals are as follows:
- SDMA,RNGB : sdma_ipg_stop_ack and rngb_ipg_stop_ack
- EIM: weim_lpack
- AHBMAX: ahbmax_halt_ack

The requests will be generated by CCM in stages. CCM will continue from stage to stage once all the acknowledges of a present stages were received. The stages are the following:

- Stage 1: SDMA, RNGB
- Stage 2: AHBMAX

Once CCM receives all the acknowledge signals needed, and if at least 8 ipg_clk cycles have elapsed since the assertion of the ccm_ipg_stop signal, then it will enter stop mode and do the following tasks in the order they are written below:

- Gate off the system block clocks, not including pgc_clk. The clock gating will include the ckil_sync clocks of Megamix (all blocks except ARM, DPLL and GPU) only if the indication from GPC is that Megamix is about to be powered down (by megamix_power_down indication). The gating signal output for the CKIL sync gating cell is gate_ckil_sync.
- Assert the system_in_stop_mode signal. This signal is connected to the IO pads for observability, to indicate stop mode is active.

Once the CCM is in STOP mode, it will check if the TZIC_DSM_WAKEUP signal has asserted or if the ARM_DSM_REQUEST has negated during the process of entering stop mode. If this has occurred, then the CCM will exit stop mode.

If the TZIC_DSM_WAKEUP is not asserted and the ARM_DSM_REQUEST is still asserted, then the CCM will move to state STOP_GPC and generate a request to the GPC to power down the ARM platform and the GPU2D by asserting the signals ccm_pdn_4arm_req and ccm_pdn_4all_req. If those modules are defined to be powered down on stop mode in the GPC, then the GPC will commence powering them down.

After powering down the modules, the GPC will send the pdn_ack signal and the following will take place:

- Shut down the PLLs and DPLLCs by de-asserting dpll_en_dpllip and ref_clk_en_dpllip.
- After de-assertion of the lock ready flag from the PLL's, close the CAMP by de-asserting ccm_camp1_en.
- If the CCM_CLPCR[SBYOS] bit was set, then de-assert the CCM_CSR[REF_EN_B] signal to close the external oscillator and assert the CCM_CLPCR[COSC_POWERDOWN] to put on the board oscillator in power down mode. (if they were not already closed by software. If one of them was already closed by software then discard the SBYOS operation for it and perform the SBYOS operation only on the one that is working. If both of them were closed in run mode , then discard SBYOS operation for both of them).
- If the CCM_CLPCR[VSTBY] bit was set, then assert the pmic_vstby_req signal, to indicate to the power management IC to shift the voltage to standby voltage.

The CCM's low power state machine will remain in the state "STOP_GPC" until the stop mode is exited.

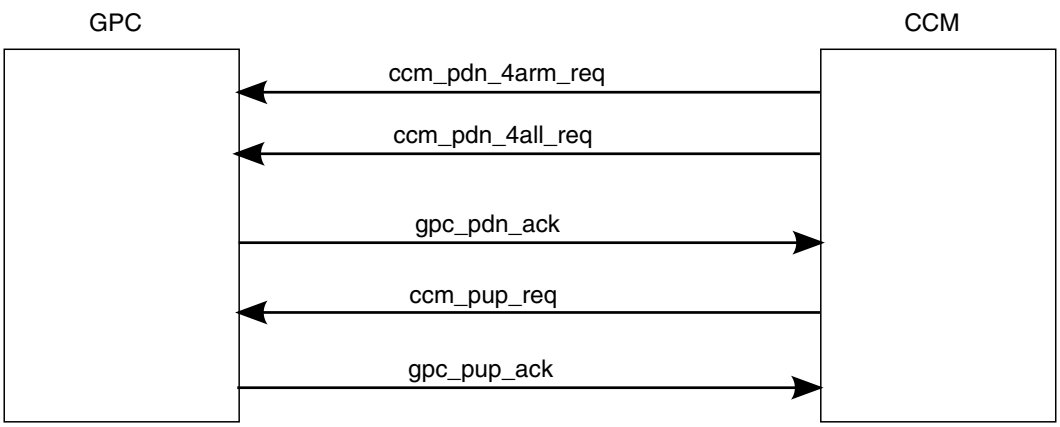


Figure 5-10. CCM-GPC Connectivity

5.3.4.3.2 Exiting Stop Mode

As soon as the unsynchronized TZIC_DSM_WAKEUP signal is asserted or the ARM_DSM_REQUEST is de-asserted, the CCM will begin the process of exiting the STOP mode.

The following steps will take place:

- If the CCM_CLPCR[VSTBY] bit was set, de-assert the pmic_vstby_req to notify the power management IC to change the voltage from standby voltage to functional voltage.
- If the CCM_CLPCR[SBYOS] was set, and the CCM closed either the external oscillator or the on board oscillator, then the CCM will start the external oscillator and the on board oscillator by asserting the CCM_CSR[REF_EN_B] signal and de-asserting the CCM_CLPCR[COSC_PWRDOWN] sigal respectively.
- After the amount of CKIL cycles defined in the CCM_CLPCR[STBY_COUNT] bits, wait until the pmic_vfunctional_ready signal is asserted. This is the notification from the power management IC that the functional voltage is ready. (Figure 5-13 describes the pmic signals connectivity.) Only after the functional voltage is ready will the CCM continues with the next steps:
  - Start the CAMP (based on definition of CCM_CCR[CAMP1_EN] bits)
  - If either the CAMP or on board oscillator were started, wait until the oscnt has finished counting to make sure that the external oscillator, camp1 and the on board oscillator are ready.
  - Start the PLLs. Only the PLLs that were configured to be on prior to the enterance to stop mode will be started.
  - Enable GPU2D clocks only.

The CCM will request the GPC to restore power by the GPC_PUP_REQ. If power was removed from the ARM platform or the GPU2D, the GPC will notify the CCM by asserting signal the GPC_PUP_ACK that power to the ARM and the GPU2D is back on, and its safe to exit from STOP mode. Only then will the CCM do the following:

- Once the SRC has asserted the src_power_gating_reset_done to indicate the reset to the power gated modules have completed, negate the low power request signals to all modules and enable all modules clocks based on their CCGR value including the ARM clocks and the ckil sync of the Megamix (if it was turned off), and return to run mode.
- Once the system is in run mode, negate the signals ccm_ipg_stop and system_in_stop_mode.

Once the interrupt propagates through all the synchronization logic, the ARM platform will recognize and service it. This will force the negation of the ARM_DSM_REQUEST output. Prior to this point, the ARM_DSM_REQUEST and TZIC_DSM_WAKEUP were both asserted. Since the TZIC_DSM_WAKEUP has top priority, the system is able to wake up.

Before the ARM exits the ISR, it will clear the interrupt source. This will cause the TZIC_DSM_WAKEUP signal to be negated. At this point, the two low power control signals are negated, and the STOP sequence can be repeated at any time.

Figure 5-11 describes the connectivity of the ARM, CCM and TZIC.

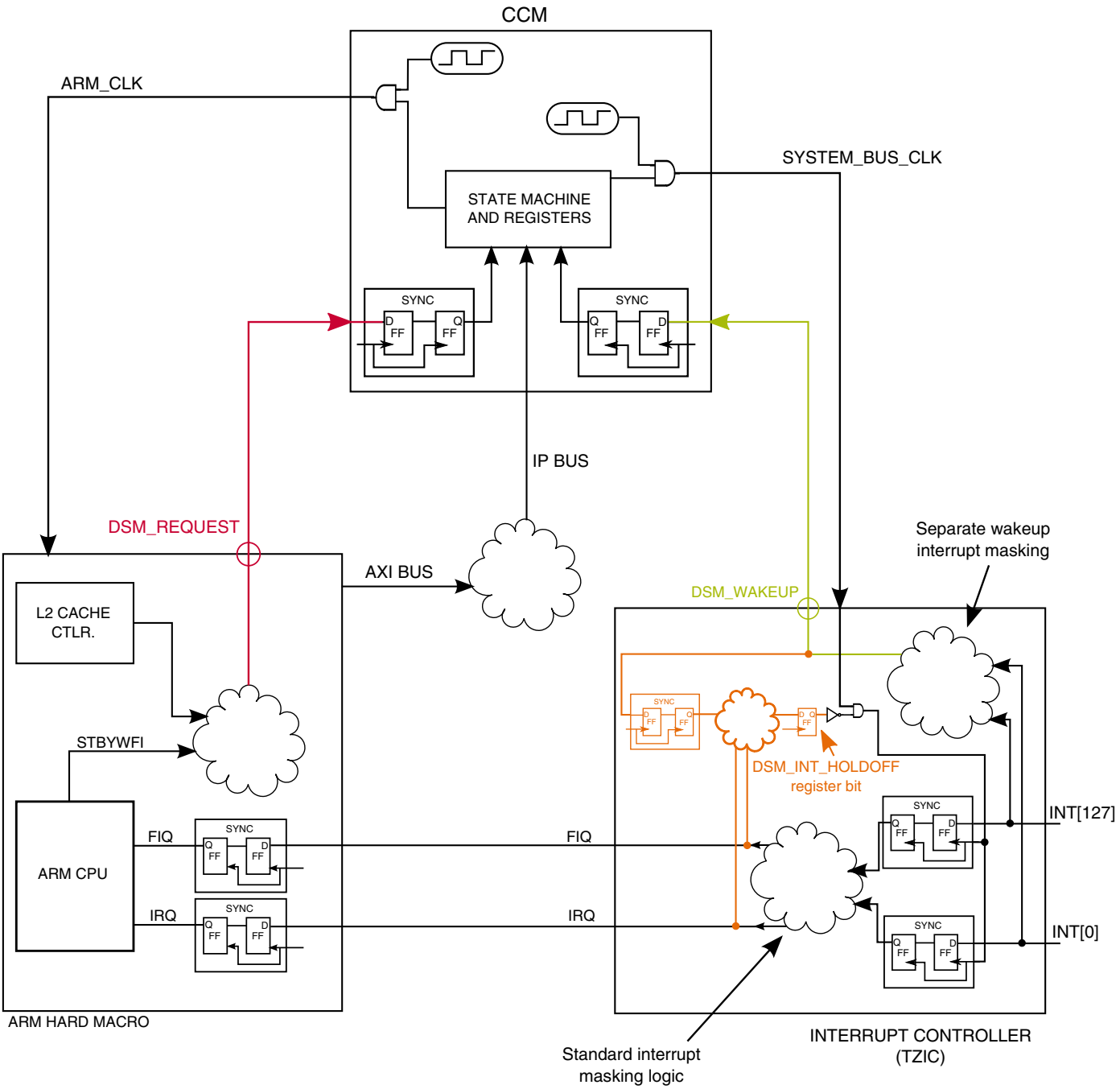


Figure 5-11. ARM Low Power Request

Figure 5-12 describes the CCM's Low Power State Machine.

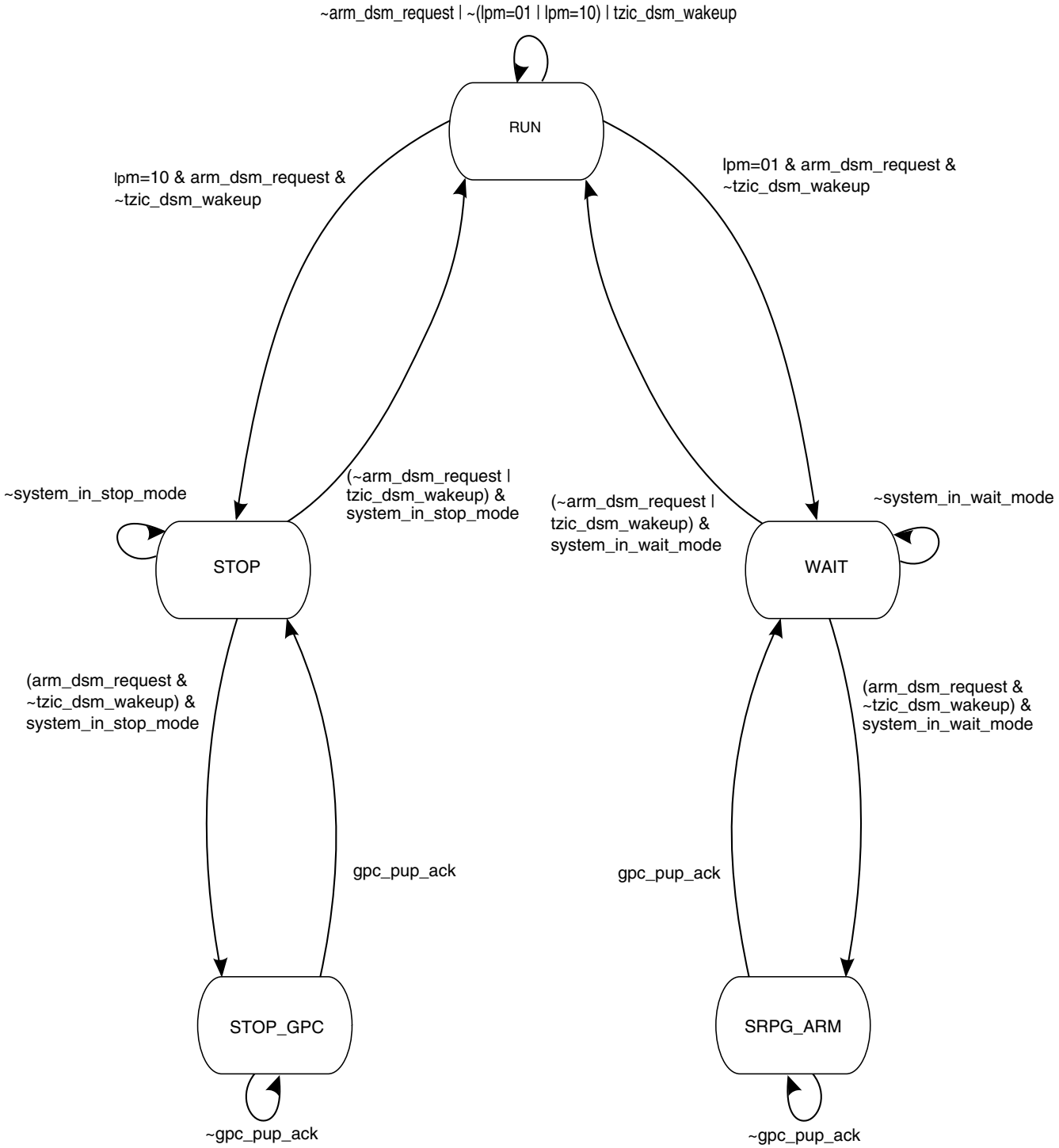


Figure 5-12. CCM Low Power State Machine

5.3.4.3.3 PMIC Signal Description

The CCM_CLPCR[STBY_COUNT] value should be larger than t1, which is the amount of time between the CCM's negation of pmic_vstby_req and the negation of pmic_vfuncional_ready (the signal coming back from PMIC to indicate that the voltage started to change).

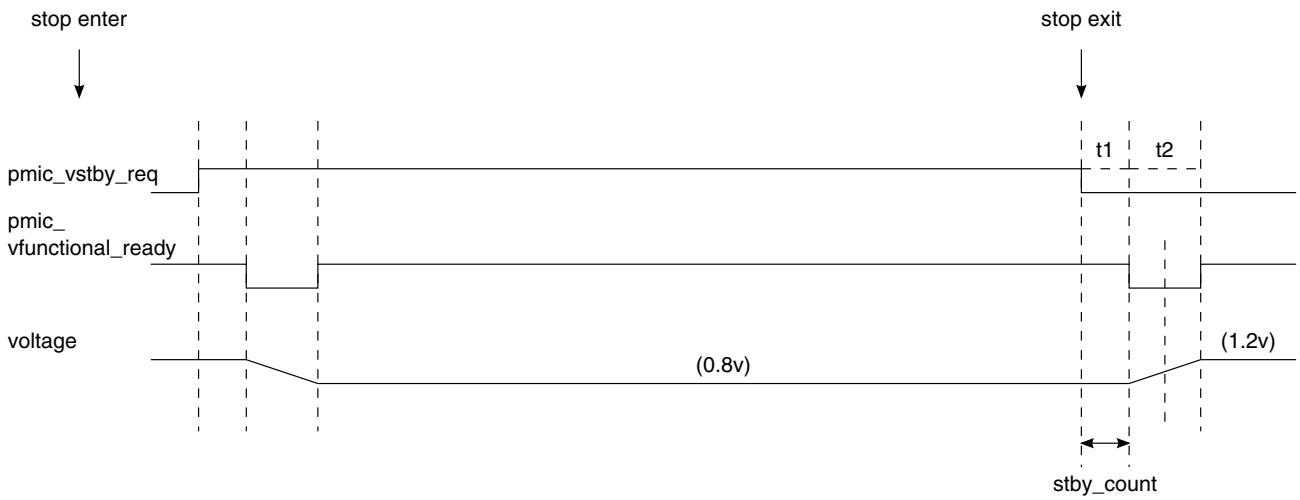


Figure 5-13. PMIC Signal Description

NOTE

Software should configure the IOMUXC so that pmic_vstby_req and pmic_vfuncional_ready will be operating in the correct ALT mode, so that they will be connected between CCM and the pads.

5.3.4.4 Low Power Audio Playback Mode (LP-APM)

Low Power Audio Playback mode (referred to as APM across this document) defines a special low power mode dedicated to the audio only playback mode. It involves PLL on/off and frequency settings, as well as voltage and clock gating aspects, to enable lowest possible power consumption.

This section covers APM, as well as assumptions, and Enter/Exit flows from the APM.

5.3.4.4.1 Low Power APM Mode Definition

Only one PLL of i.MX50 is running—ARM PLL. This PLL is set to the maximal frequency that the ARM can run at that lowest point while satisfying the ARM and bus system processing needs.

The peripherals will also work on low voltage (1.0 v). The peripherals will work at a frequency suitable for this low voltage (about a third of the nominal frequency).

For i.MX50 used as master on SSI clocks, the MCU PLL frequency must be set to an integer multiplication of SSI clock.

5.3.4.4.2 LP-APM Mode Restrictions

Entering and exiting LP-APM mode has an impact on the SSI clocks. If the chip is a master of the SSI clocks, audio playback will be interrupted during the switch, and therefore it is not recommended to switch modes during audio playback.

5.3.4.4.3 LP-APM Entry and Exit

Table 5-5 describes an example sequence for switching from a high power setting to LP-APM. Table 5-6 describes the corresponding sequence to exit from LP-APM. These entry and exit flows are valid for DDR types that support 24 MHz frequency.

Table 5-5. Stages of Entering LP-APM Mode

Stage	ARM Frequency (MHz)	ARM Voltage (V)	AHB Frequency (MHz)	DDR Frequency (MHz)	Vcc Voltage (V)	Comments
Initial State	Initial freq	Initial Voltage	Initial freq, sourced from PLL2	Initial freq, sourced from PLL1	Initial Voltage	ARM and DDR are sourced from PLL1 at initial frequency. AHB, AXI, etc. sourced from PLL2. ARM and Vcc at initial voltage.
DVFS on ARM	**Low freq**	**Low Voltage**	Initial freq, sourced from PLL2	Initial freq, sourced from PLL1	Initial Voltage	Use ARM_PODF to reduce ARM to low frequency.
Change DDR Frequency	**Low freq**	**Low Voltage**	Initial freq, sourced from PLL2	**Low freq, sourced from 24 MHz OSC**	Initial Voltage	Set DDR to be in Sync mode and source from 24 MHz OSC.
Change bus dividers	**Low freq**	**Low Voltage**	**Low freq, sourced from 24 MHz OSC**	**Low freq, sourced from 24 MHz OSC**	Initial Voltage	Change AHB, AXI dividers and source periph_clk from 24 MHz OXC. Relock PLL1 to low freq.
Reduce Peripheral Voltage (Vcc)	**Low freq**	**Low Voltage**	**Low freq, sourced from 24 MHz OSC**	**Low freq, sourced from 24 MHz OSC**	**Low Voltage**	Change Vcc.

NOTE

It is possible to source ARM from 24MHz OSC when the system enters WAIT mode in LPAPM state, so that all PLLs are OFF in LPAPM mode.

Table 5-6. Stages of Exiting LP-APM Mode

Stage	ARM Frequency (MHz)	ARM Voltage (V)	AHB Frequency (MHz)	DDR Frequency (MHz)	Vcc Voltage (V)	Comments
Initial State	**Low freq**	**Low Voltage**	**Low frequency, sourced from 24 MHz OSC**	**Low frequency, sourced from 24 MHz OSC**	**Low Voltage**	PLL1 locked at low frequency.
Relock PLL1	**Low freq**	**Low Voltage**	**Low frequency, sourced from 24 MHz OSC**	**Low frequency, sourced from 24 MHz OSC**	**Low Voltage**	PLL1 relocked at higher frequency.
Increase Peripheral Voltage (Vcc)	**Low freq**	**Low Voltage**	**Low frequency, sourced from 24 MHz OSC**	**Low frequency, sourced from 24 MHz OSC**	Normal Voltage	Change Vcc.
Change bus dividers	**Low freq**	**Low Voltage**	Normal frequency, sourced from PLL2	**Low frequency, sourced from 24 MHz OSC**	Normal Voltage	Change AHB, AXI dividers and source periph_clk from PLL2.
Change DDR Frequency	**Low freq**	**Low Voltage**	Normal frequency, sourced from PLL2	Normal frequency, sourced from PLL1	Normal Voltage	Set DDR to be in async mode and source from PLL1.
Raise ARM Frequency	Normal freq	Normal Voltage	Normal frequency, sourced from PLL2	Normal frequency, sourced from PLL1	Normal Voltage	Change ARM_PODF to increase ARM frequency (if needed).

Figure 5-14 and Figure 5-15 provide examples of the LP-APM entry flow using specific PLL frequencies and supply voltages:

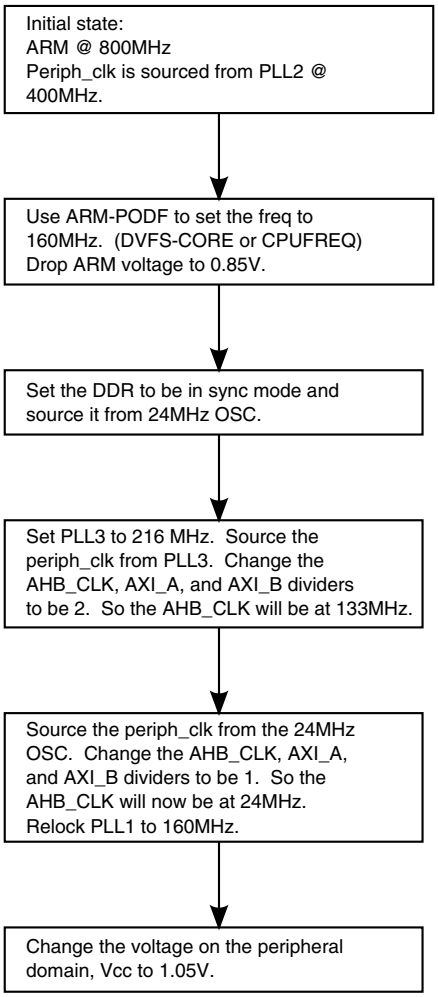


Figure 5-14. Flow of Entering Low Power APM on i.MX50

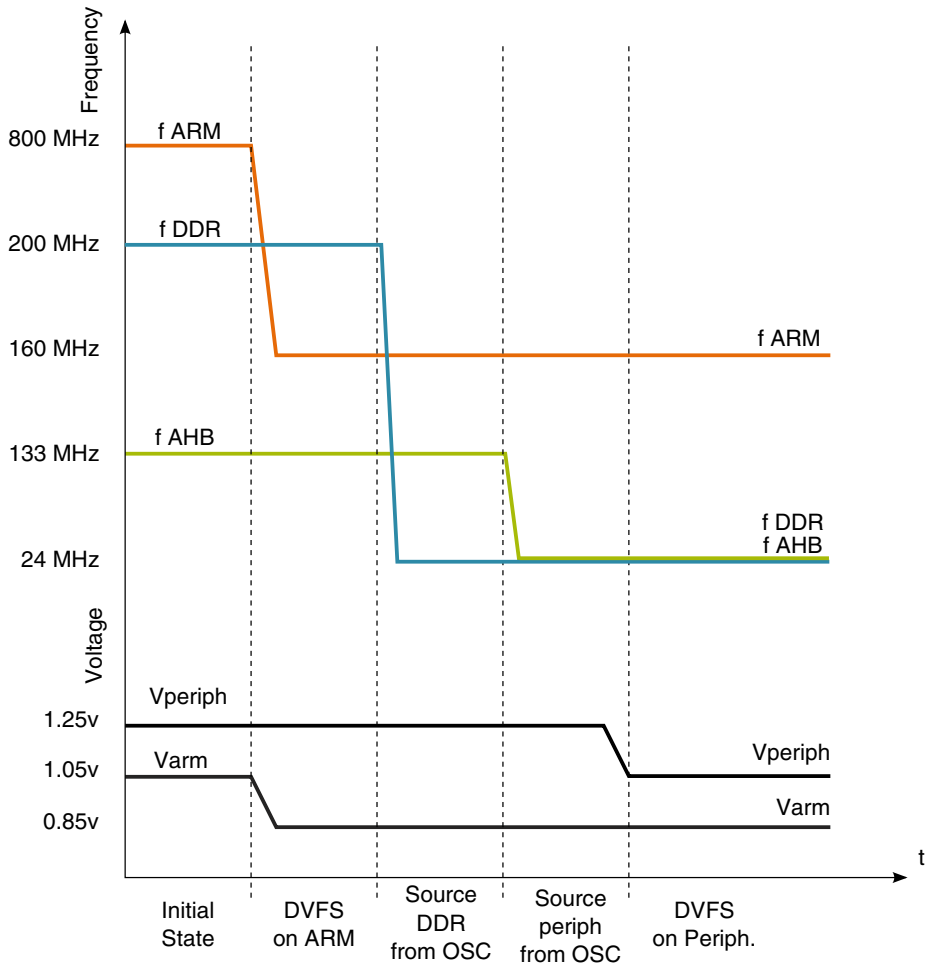


Figure 5-15. Frequency and Voltage Flow

5.3.4.5 Recomendations for Using Low Power Consumption from CCM

In addition to using APM mode for low power, here are recomendations of configuring CCM so that it will consume less power. Those recomendations should be followed when applicable:

- Using AHB clock source as the source of all AXI buses—in this case the power of the AXI dividers will be preserved.
- Working from one PLL or PFD and disabling the other PLLs.
- Generate SSI external clocks from SSI clock roots—in this case the SSI external clocks dividers' power will be preserved.

5.4 CCM Memory Map/Register Definition

CCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FD_4000	CCM Control Register (CCM_CCR)	32	R/W	0000_02FFh	5.4.1/323
53FD_4004	CCM Control Divider Register (CCM_CCDR)	32	R	0000_0000h	5.4.2/324
53FD_4008	CCM Status Register (CCM_CSR)	32	R	0000_0010h	5.4.3/325
53FD_400C	CCM Clock Switcher Register (CCM_CCSR)	32	R/W	0000_0004h	5.4.4/326
53FD_4010	CCM ARM Clock Root Register (CCM_CACRR)	32	R/W	0000_0000h	5.4.5/328
53FD_4014	CCM Bus Clock Divider Register (CCM_CBCDR)	32	R/W	0600_0000h	5.4.6/329
53FD_4018	CCM Bus Clock Multiplexer Register (CCM_CBCMR)	32	R/W	0001_0005h	5.4.7/332
53FD_401C	CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1)	32	R/W	F363_F130h	5.4.8/333
53FD_4020	CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)	32	R	0000_0000h	5.4.9/336
53FD_4024	CCM Serial Clock Divider Register 1 (CCM_CSCDR1)	32	R/W	0000_0000h	5.4.10/336
53FD_4028	CCM SSI1 Clock Divider Register (CCM_CS1CDR)	32	R/W	0040_0040h	5.4.11/339
53FD_402C	CCM SSI2 Clock Divider Register (CCM_CS2CDR)	32	R/W	0040_0040h	5.4.12/340
53FD_4030	CCM DI Clock Divider Register (CCM_CDCDR)	32	R	0000_0000h	5.4.13/342
53FD_4034	CCM HSC Clock Divider Register (CCM_CHSCCDR)	32	R	0000_0000h	5.4.14/342
53FD_4038	CCM Serial Clock Divider Register 2 (CCM_CSCDR2)	32	R/W	0200_0000h	5.4.15/343
53FD_403C	CCM Serial Clock Divider Register 3 (CCM_CSCDR3)	32	R	0000_0000h	5.4.16/344
53FD_4040	CCM Serial Clock Divider Register 4 (CCM_CSCDR4)	32	R	0000_0000h	5.4.17/344
53FD_4044	CCM Wakeup Detector Register (CCM_CWDR)	32	R	0000_0000h	5.4.18/345
53FD_4048	CCM Divider Handshake In-Process Register (CCM_CDHIPR)	32	R	0000_0000h	5.4.19/346
53FD_404C	CCM DVFS Control Register (CCM_CDCR)	32	R/W	0000_0000h	5.4.20/348
53FD_4050	CCM Testing Observability Register (CCM_CTOR)	32	R/W	0000_0000h	5.4.21/349
53FD_4054	CCM Low Power Control Register (CCM_CLPCR)	32	R/W	0000_0061h	5.4.22/351
53FD_4058	CCM Interrupt Status Register (CCM_CISR)	32	w1c	0000_0000h	5.4.23/354
53FD_405C	CCM Interrupt Mask Register (CCM_CIMR)	32	R/W	FFFF_ FFFFh	5.4.24/356
53FD_4060	CCM Clock Output Source Register (CCM_CCOSR)	32	R/W	001D_0001h	5.4.25/358
53FD_4064	CCM General Purpose Register (CCM_CGPR)	32	R	0000_0000h	5.4.26/361
53FD_4068	CCM Clock Gating Register 0 (CCM_CCGR0)	32	R/W	3FC0_03F3h	5.4.27/361

Table continues on the next page...

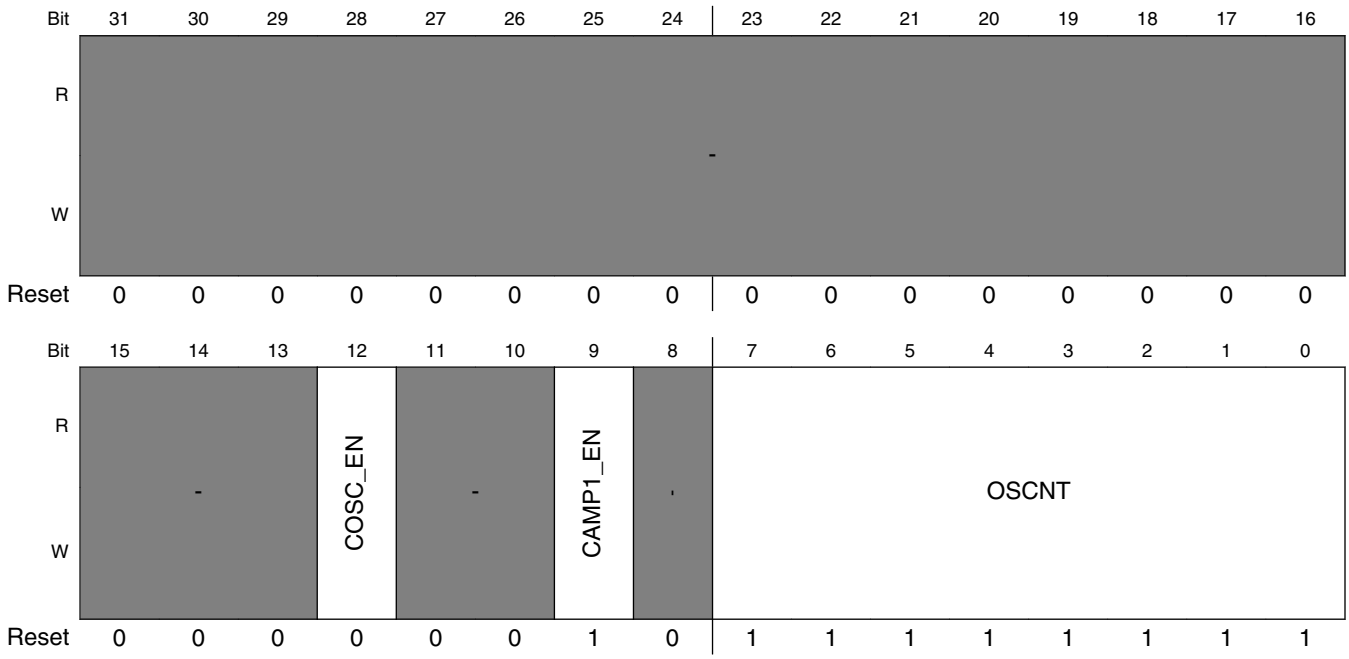
CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FD_406C	CCM Clock Gating Register 1 (CCM_CCGR1)	32	R/W	3FC0_03F3h	5.4.28/364
53FD_4070	CCM Clock Gating Register 2 (CCM_CCGR2)	32	R/W	0000_0000h	5.4.29/366
53FD_4074	CCM Clock Gating Register 3 (CCM_CCGR3)	32	R/W	0000_003Fh	5.4.30/369
53FD_4078	CCM Clock Gating Register 4 (CCM_CCGR4)	32	R/W	C000_0000h	5.4.31/371
53FD_407C	CCM Clock Gating Register 5 (CCM_CCGR5)	32	R/W	0003_3000h	5.4.32/374
53FD_4080	CCM Clock Gating Register 6 (CCM_CCGR6)	32	R/W	0000_000Fh	5.4.33/376
53FD_4084	CCM Clock Gating Register 7 (CCM_CCGR7)	32	R/W	0C00_0000h	5.4.34/379
53FD_4088	CCM Module Enable Override Register (CCM_CMEOR)	32	R/W	FFFF_ FFFFh	5.4.35/382
53FD_408C	CCM Control Status Register 2 (CCM_CSR2)	32	R/W	0000_0400h	5.4.36/383
53FD_4090	CCM Clock Sequence Bypass (CCM_CLKSEQ_BYPASS)	32	R/W	0000_0000h	5.4.37/385
53FD_4094	CCM System Clock Register (CCM_CLK_SYS)	32	R/W	8000_0041h	5.4.38/387
53FD_4098	CCM DDR Clock Register (CCM_CLK_DDR)	32	R/W	0000_0001h	5.4.39/388
53FD_409C	CCM ELCDIF PIX Clock Serial Divide Register (CCM_ELCDIFPIX)	32	R/W	0000_1001h	5.4.40/389
53FD_40A0	CCM EPDC PIX Clock Serial Divide Register (CCM_EPDCPIX)	32	R/W	0000_1001h	5.4.41/390
53FD_40A4	CCM DISPLAY_AXI Clock Divide Register (CCM_DISPLAY_AXI)	32	R/W	0000_0001h	5.4.42/392
53FD_40A8	CCM EPDC_AXI Clock Divide Register (CCM_EPDC_AXI)	32	R/W	0000_0001h	5.4.43/394
53FD_40AC	CCM GPMI Clock Divide Register (CCM_GPMI)	32	R/W	0000_0001h	5.4.44/395
53FD_40B0	CCM BCH Clock Divide Register (CCM_BCH)	32	R/W	0000_0001h	5.4.45/396
53FD_40B4	CCM MSHC_XMSCKI Clock Divide Register (CCM_MSHC_XMSCKI)	32	R/W	0000_0001h	5.4.46/397

5.4.1 CCM Control Register (CCM_CCR)

The following figure represents the CCM Control Register (CCR), which contains bits to control general operation of CCM.

Address: CCM_CCR is 53FD_4000h base + 0h offset = 53FD_4000h



CCM_CCR field descriptions

Field	Description
31–13 -	Reserved
12 COSC_EN	On chip oscillator enable bit - this bit value is reflected on the output cosc_en. The system will start with on chip oscillator enabled to supply source for the PLL's. Software can change this bit if a transition to the bypass PLL clocks was performed for all the PLLs. In cases that this bit is changed from '0' to '1' then CCM will enable the on chip oscillator and after counting oscnt ckil clock cycles it will notify that on chip oscillator is ready by a interrupt cosc_ready and by status bit cosc_ready. The cosc_en bit should be changed only when on chip oscillator is not chosen as the clock source.0 disable on chip oscillator 0 disable on chip oscillator 1 enable on chip oscillator
11–10 -	Reserved
9 CAMP1_EN	CAMP1 Enable bit - the ignition process will always set this bit. In run mode, software can control camp1 enable/disable through this bit. If this bit is changed from '0' to '1' then CCM will enable the camp1 and after counting oscnt ckil's it will notify that camp1 is ready by a interrupt camp1_ready and by status bit camp1_ready. The camp1_en bit should be changed only when camp1 is not chosen as a clock source. **NOTE:** CCM has an output ccm_camp1_dis. this signal will be inverted from the camp1_en bit, that is, when camp1_en bit is '1' the ccm_camp1_dis signal is '0'.

Table continues on the next page...

CCM_CCR field descriptions (continued)

Field	Description
	0 disable camp1 1 enable camp1
8 -	Reserved
7–0 OSCNT	Oscillator ready counter value. These bits define value of 32KHz counter, that serve as counter for oscillator lock time. This is used for both the on chip oscillator lock time and the time that CAMP1 will be ready since the CAMP receives the external oscillator clock. Current estimation is ~5ms. This counter will be used in ignition sequence and in wake from stop sequence if sbyos bit (CCM Low Power Control Register) was defined, to notify that on chip oscillator output is ready for the DPLLC to use and only then the gate in DPLLC can be opened. 000000 count 1 ckil 111111 count 256 ckil's

5.4.2 CCM Control Divider Register (CCM_CCDR)

The following figure represents the CCM Control Divider Register (CCDR), which contains bits that control the loading of the dividers that need handshake with the modules they affect.

Address: CCM_CCDR is 53FD_4000h base + 4h offset = 53FD_4004h

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

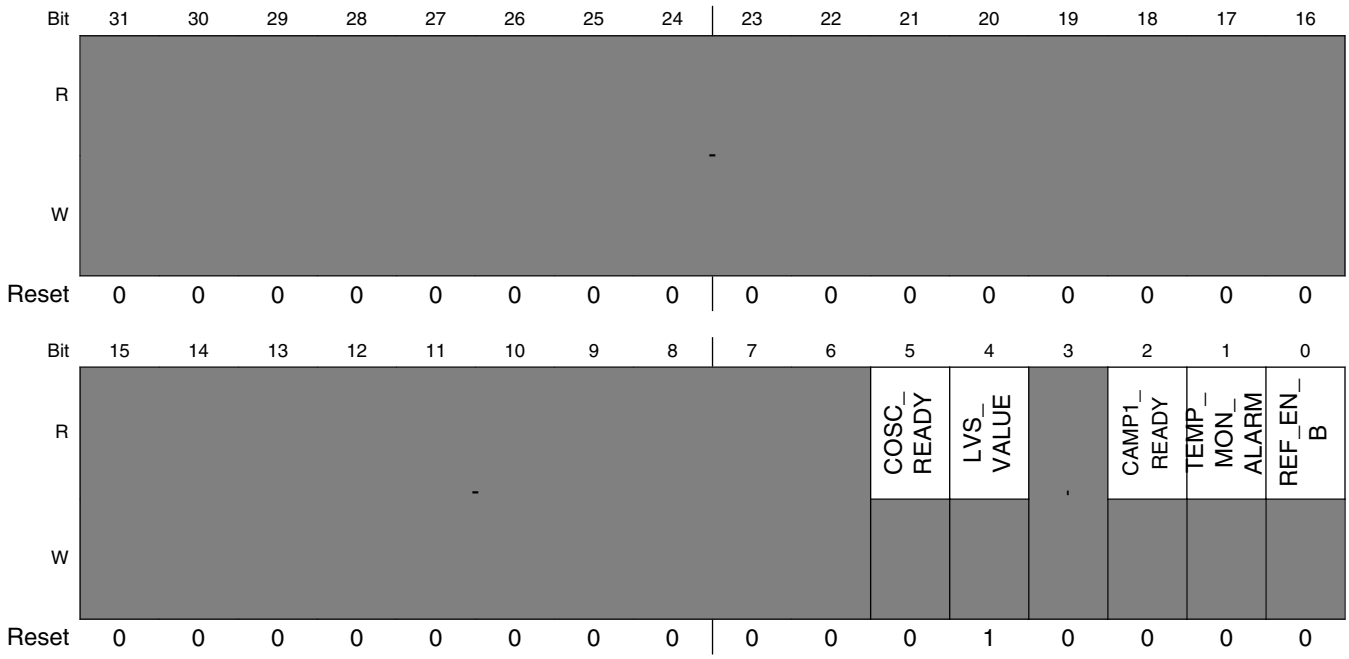
CCM_CCDR field descriptions

Field	Description
31–0 -	Reserved

5.4.3 CCM Status Register (CCM_CSR)

The following figure represents the CCM status Register (CSR). The status bits are read only bits.

Address: CCM_CSR is 53FD_4000h base + 8h offset = 53FD_4008h



CCM_CSR field descriptions

Field	Description
31–6 -	Reserved
5 COSC_READY	Status indication of on board oscillator. This bit will be asserted if on chip oscillator is enabled and on chip oscillator is not powered down, and if oscnt counter has finished counting. 0 on board oscillator is not ready. 1 on board oscillator is ready.
4 LVS_VALUE	Status of the value of pll_lvs control from the CCM to DPLLC 0 value of pll_lvs output is '0' 1 value of pll_lvs output is '1'
3 -	Reserved
2 CAMP1_READY	Status indication of camp1. This will be asserted if camp1 was enabled, and if oscnt counter has finished counting. 0 camp1 is not ready. 1 camp1 is ready.

Table continues on the next page...

CCM_CSR field descriptions (continued)

Field	Description
1 TEMP_MON_ALARM	Indicates the status of input "temp_mon_alarm" of the Temperature Monitor logic. 0 value of "temp_mon_alarm" is '0' 1 value of "temp_mon_alarm" is '1'
0 REF_EN_B	Status of the value of ref_en_b output of ccm 0 value of ref_en_b is '0' 1 value of ref_en_b is '1'

5.4.4 CCM Clock Switcher Register (CCM_CCSR)

The following figure represents the CCM Clock Switcher register (CCSR). The CCSR register contains bits to control the switcher sub module dividers and multiplexers.

Address: CCM_CCSR is 53FD_4000h base + Ch offset = 53FD_400Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-		PLL3_PFD_EN	PLL2_PFD_EN	PLL1_PFD_EN	LP_APM	-	STEP_SEL [1:0]		PLL2_DIV_PODF [1:0]		PLL3_DIV_PODF [1:0]		PLL1_SW_CLK_SEL	PLL2_SW_CLK_SEL	PLL3_SW_CLK_SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

CCM_CCSR field descriptions

Field	Description
31–14 -	Reserved
13 PLL3_PFD_EN	PFD enable for pll3_sw_clk 0 select pll3_main_clk or pll3 bypass clock 1 select PFD source if PLL3 not in bypass mode. Refer to CCSR bit 0.

Table continues on the next page...

CCM_CCSR field descriptions (continued)

Field	Description
12 PLL2_PFD_EN	PFD enable for pll2_sw_clk 0 select pll2_main_clk or pll2 bypass clock 1 select PFD source if PLL2 not in bypass mode. Refer to CCSR bit 1.
11 PLL1_PFD_EN	PFD enable for pll1_sw_clk 0 select pll1_main_clk or pll1 step clock 1 select PFD source if PLL1 not in step clock mode. Refer to CCSR bit 2.
10 LP_APM	Selects the option to be chosen for the Low Power Audio Playback source clock. 0 Oscillator clock output 1 Fixed 480MHz PLL clock output
9 -	Reserved
8–7 STEP_SEL [1:0]	Selects the option to be chosen for the step frequency when shifting ARM frequency. this will control the step_clk. **NOTE:** This mux is allowed to be changed only if its output is not used, i.e. ARM uses the output of pll1. **NOTE:** To consume power, it is preferred to not set this mux to the pll2 and pll3 options when they are not needed - if they are not chosen then the path from pll2 and pll3 to this mux will be shut down and its power will be consumed. 00 clock source 4 - source for lp_apm. (Default) 01 pll1 bypass clock 10 divided pll2 clock 11 divided pll3 clock
6–5 PLL2_DIV_ PODF [1:0]	Divider for pll2 clock. **NOTE:** This podf is allowed to be changed only during period that its output is not used. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
4–3 PLL3_DIV_ PODF [1:0]	Divider for pll3 clock. **NOTE:** This podf is allowed to be changed only during period that its output is not used. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
2 PLL1_SW_CLK_ SEL	Selects source to generate pll1_sw_clk. **NOTE:** This bit will be ored with pll_bypass_en1 signal and dvfs_control signal. If one of the sources requests to move to step_clk (pll1_sw_clk=1 or pll_bypass_en1=1 or dvfs_control=1) then the pll1_sw_clk will be step_clk. Only if both sources request pll1_main_clk (pll1_sw_clk=0 and pll_bypass_en1=0 and dvfs_control=0) then the pll1_sw_clk will be pll1_main_clk.

Table continues on the next page...

CCM_CCSR field descriptions (continued)

Field	Description
	0 pll1_main_clk 1 step_clk (Default)
1 PLL2_SW_CLK_SEL	Selects source to generate pll2_sw_clk. This bit should only be used for testing purposes. **NOTE:** This bit will be ored with pll_bypass_en2 signal. If one of the sources requests to move to pll2 bypass clk (pll2_sw_clk=1 or pll_bypass_en2=1) then the pll2_sw_clk will be pll2 bypass clk. Only if both sources request pll2_main_clk (pll2_sw_clk=0 and pll_bypass_en2=0) then the pll2_sw_clk will be pll2_main_clk. 0 pll2_main_clk(Default) 1 pll2 bypass clock
0 PLL3_SW_CLK_SEL	Selects source to generate pll3_sw_clk. This bit should only be used for testing purposes. **NOTE:** This bit will be ored with pll_bypass_en3 signal. If one of the sources requests to move to pll3 bypass clk (pll3_sw_clk=1 or pll_bypass_en3=1) then the pll3_sw_clk will be pll3 bypass clk. Only if both sources request pll3_main_clk (pll3_sw_clk=0 and pll_bypass_en3=0) then the pll3_sw_clk will be pll3_main_clk. 0 pll3_main_clk(Default) 1 pll3 bypass clock

5.4.5 CCM ARM Clock Root Register (CCM_CACRR)

The following figure represents the CCM ARM Clock Root register (CACRR). The CACRR register contains bits to control the ARM clock root generation.

Address: CCM_CACRR is 53FD_4000h base + 10h offset = 53FD_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-													ARM_PODF [2:0]		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CACRR field descriptions

Field	Description
31–3 -	Reserved
2–0 ARM_PODF [2:0]	Divider for ARM clock root. **NOTE:** If arm_freq_shift_divider is set to '1' then any new write to arm_podf will be held until arm_clk_switch_req signal is asserted.

Table continues on the next page...

CCM_CACRR field descriptions (continued)

Field	Description
	000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8

5.4.6 CCM Bus Clock Divider Register (CCM_CBCDR)

The following figure represents the CCM Bus Clock Divider Register (CBCDR). The CBCDR register contains bits to control the clock generation sub module dividers.

Address: CCM_CBCDR is 53FD_4000h base + 14h offset = 53FD_4014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-				WEIM_CLK_SEL	PERIPH_CLK_SEL		WEIM_PODF [2:0]			AXI_B_PODF [2:0]			AXI_A_PODF [2:0]		
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-			AHB_PODF [2:0]			IPG_PODF [1:0]		PERCLK_PRED1 [1:0]		PERCLK_PRED2 [2:0]			PERCLK_PODF [2:0]		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CBCDR field descriptions

Field	Description
31–28 -	Reserved
27 WEIM_CLK_SEL	Selector for weim clock group **NOTE:** Any change of this multiplexer might involve handshake with WEIM. See CDHIPR register for the handshake busy bits. 0 derive clock from ahb clock root 1 derive clock from dvfs divider
26–25 PERIPH_CLK_SEL	Selector for peripheral main clock. **NOTE:** Any change of this multiplexer will involve handshake with WEIM- a similar handshake to the one that is performed on peripheral dvfs.

Table continues on the next page...

CCM_CBCDR field descriptions (continued)

Field	Description
	00 derive clock from pll1_sw_clk clock source. 01 derive clock from pll2_sw_clk clock source. 10 derive clock from pll3_sw_clk clock source. 11 derive clock from lp_apm clock source
24–22 WEIM_PODF [2:0]	Divider for weim podf. **NOTE:** Any change of this divider might involve handshake with WEIM. See CDHIPR register for the handshake busy bits. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
21–19 AXI_B_PODF [2:0]	Divider for axi b podf. **NOTE:** Any change of this divider might involve handshake with WEIM See CDHIPR register for the handshake busy bits. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
18–16 AXI_A_PODF [2:0]	Divider for axi a podf. **NOTE:** Any change of this divider might involve handshake with WEIM. See CDHIPR register for the handshake busy bits. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
15–13 -	Reserved
12–10 AHB_PODF [2:0]	Divider for ahb podf. **NOTE:** Any change of this divider might involve handshake with WEIM. See CDHIPR register for the handshake busy bits.

Table continues on the next page...

CCM_CBCDR field descriptions (continued)

Field	Description
	000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
9–8 IPG_PODF [1:0]	Divider for ipg podf. **NOTE:** Any change of this divider might involve handshake with WEIM. See CDHIPR register for the handshake busy bits. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
7–6 PERCLK_ PRED1 [1:0]	Divider for perclk pred1 **NOTE:** Divider should be updated when output clock is gated. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
5–3 PERCLK_ PRED2 [2:0]	Divider for perclk pred2. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
2–0 PERCLK_PODF [2:0]	Divider for perclk podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8

5.4.7 CCM Bus Clock Multiplexer Register (CCM_CBCMR)

The following figure represents the CCM Bus Clock Multiplexer Register (CBCMR). The CBCMR register contains bits to control the multiplexers that generate the bus clocks.

NOTE

Any change to the above multiplexers must be done while the block whose clock is affected is not functional and the respective clock is gated in LPCG. If the change is made while the block is in the middle of an operation, it is not guaranteed that the module's operation will not be harmed.

When switching the divider from a slow clock source to a fast clock source, it is recommended to follow the steps below. The divider is not guaranteed to stop at the beginning of the divide sequence:

1. Change the divide value.
2. Enable the divider to allow the new divide to take effect.
3. Gate off divider, and switch to fast clock source.
4. Enable divider.

Address: CCM_CBCMR is 53FD_4000h base + 18h offset = 53FD_4018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-														GPU2D_CLK_SEL [1:0]	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-												DEBUG_APB_CLK_SEL [1:0]		PERCLK_LP_APM_SEL	PERCLK_IPG_SEL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

CCM_CBCMR field descriptions

Field	Description
31–18 -	Reserved
17–16 GPU2D_CLK_SEL [1:0]	Selector for open vg clock multiplexer 00 derive clock from axi a

Table continues on the next page...

CCM_CBCMR field descriptions (continued)

Field	Description
	01 derive clock from axi b 10 derive clock from weim_clk_root 11 derive clock from ahb clock root
15–4 -	Reserved
3–2 DEBUG_APB_CLK_SEL [1:0]	Selector for debug apb clock multiplexer 00 derive clock from axi a 01 derive clock from axi b 10 derive clock from weim_clk_root 11 derive clock from ahb clock root
1 PERCLK_LP_APM_SEL	Controls if the root clock generation of perclk will be from peripherals main clock or lp_apm source. 0 generate perclk_root from peripherals main clock source. 1 generate perclk_root from lp_apm source.
0 PERCLK_IPG_SEL	Selector for perclk multiplexer - allows to select between ipg_clk or the division of chosen PLL. 0 select perclk generation from division of pll frequency 1 select perclk generation from ipg_clk

5.4.8 CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1)

The following figure represents the CCM Serial Clock Multiplexer Register 1 (CSCMR1). The CSCMR1 register contains bits to control the multiplexers that generate the serial clocks.

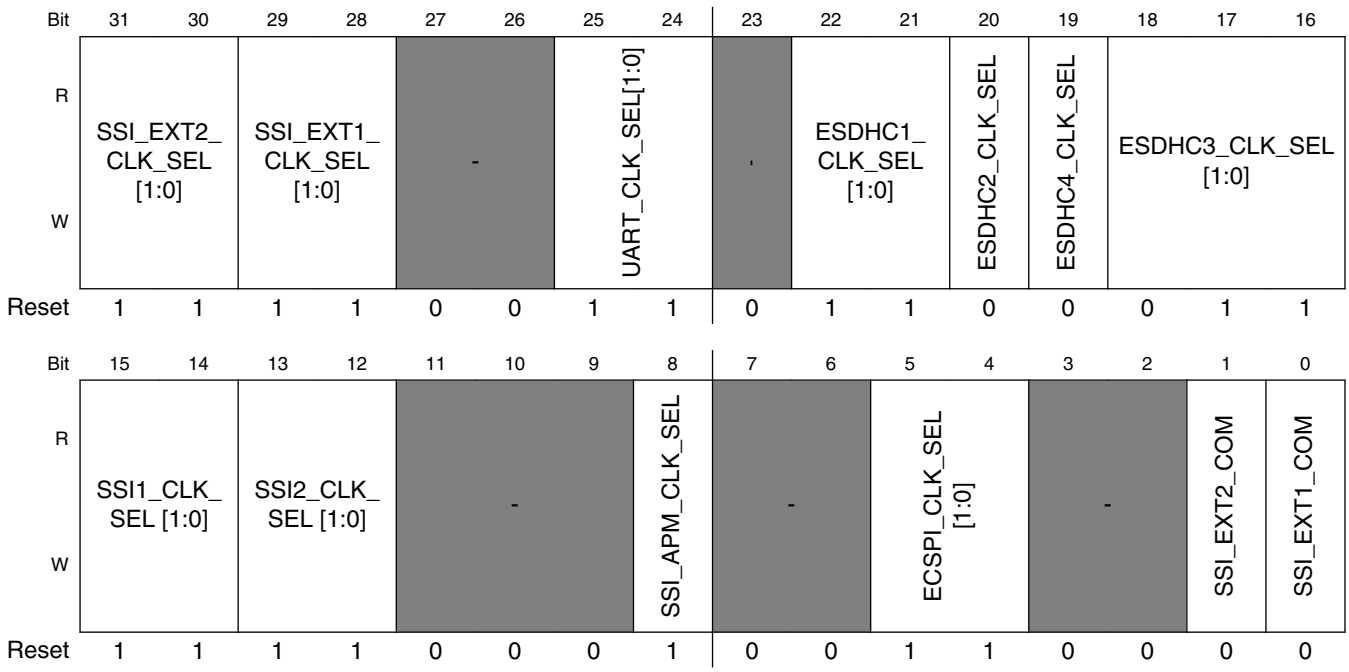
NOTE

Any change to the above multiplexers will have to be done while the block whose clock is affected is not functional and the respective clock is gated in LPCG. If the change is made while the block is in the middle of an operation, it is not guaranteed that the module's operation will not be harmed.

When switching the divider from a slow clock source to a fast clock source, it is recommended to follow the steps below. The divider is not guaranteed to stop at the beginning of the divide sequence:

1. Change the divide value.
2. Enable the divider to allow the new divide to take effect.
3. Gate off divider, and switch to fast clock source.
4. Enable divider.

Address: CCM_CSCMR1 is 53FD_4000h base + 1Ch offset = 53FD_401Ch



CCM_CSCMR1 field descriptions

Field	Description
31–30 SSI_EXT2_CLK_SEL [1:0]	Selector for ssi_ext2 clock multiplexer 00 derive clock from pll1_sw_clk 01 derive clock from pll2_sw_clk 10 derive clock from pll3_sw_clk 11 derive clock from ssi_lp_apm_clk
29–28 SSI_EXT1_CLK_SEL [1:0]	Selector for ssi_ext1 clock multiplexer 00 derive clock from pll1_sw_clk 01 derive clock from pll2_sw_clk 10 derive clock from pll3_sw_clk 11 derive clock from ssi_lp_apm_clk
27–26 -	Reserved
25–24 UART_CLK_SEL[1:0]	Selector for uart clock multiplexer 00 derive clock from pll1_sw_clk 01 derive clock from pll2_sw_clk 10 derive clock from pll3_sw_clk 11 lp_apm clock
23 -	Reserved
22–21 ESDHC1_CLK_SEL [1:0]	Selector for esdhc1 clock multiplexer 00 derive clock from pll1_sw_clk

Table continues on the next page...

CCM_CSCMR1 field descriptions (continued)

Field	Description
	01 derive clock from pll2_sw_clk 10 derive clock from pll3_sw_clk 11 lp_apm clock
20 ESDHC2_CLK_SEL	Selector for esdhc2 clock multiplexer 0 derive clock from esdhc1_clk_sel 1 derive clock from esdhc3_clk_sel
19 ESDHC4_CLK_SEL	Selector for esdhc4 clock multiplexer 0 derive clock from esdhc1_clk_sel 1 derive clock from esdhc3_clk_sel
18–16 ESDHC3_CLK_SEL [1:0]	Selector for esdhc3 clock multiplexer 000 derive clock from pll1_sw_clk 001 derive clock from pll2_sw_clk 010 derive clock from pll3_sw_clk 011 lp_apm clock 100 derive clock from PFD0 101 derive clock from PFD1 110 derive clock from PFD4 111 derive clock from osc_clk
15–14 SSI1_CLK_SEL [1:0]	Selector for ssi1 clock multiplexer 00 derive clock from pll1_sw_clk 01 derive clock from pll2_sw_clk 10 derive clock from pll3_sw_clk 11 derive clock from ssi_lp_apm_clk
13–12 SSI2_CLK_SEL [1:0]	Selector for ssi2 clock multiplexer 00 derive clock from pll1_sw_clk 01 derive clock from pll2_sw_clk 10 derive clock from pll3_sw_clk 11 derive clock from ssi_lp_apm_clk
11–9 -	Reserved
8 SSI_APM_CLK_SEL	Controlls the multiplexer for the ssi_lp_apm_clk clock generation. 0 CAMP1 of CKIH 1 LP-APM clock selector output
7–6 -	Reserved
5–4 ECSPI_CLK_SEL [1:0]	Selector for ecspi clock multiplexer 00 derive clock from pll1_sw_clk 01 derive clock from pll2_sw_clk 10 derive clock from pll3_sw_clk 11 lp_apm clock

Table continues on the next page...

CCM_CSCMR1 field descriptions (continued)

Field	Description
3–2 -	Reserved
1 SSI_EXT2_COM	Controlls the multiplexer to generate the ssi_ext2 clock from the ssi2 clock root. 0 generate ssi_ext2_clk_root from dividers 1 generate ssi_ext2_clk_root from ssi2_clk_root.
0 SSI_EXT1_COM	Controlls the multiplexer to generate the ssi_ext1 clock from the ssi1 clock root. 0 generate ssi_ext1_clk_root from dividers 1 generate ssi_ext1_clk_root from ssi1_clk_root.

5.4.9 CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)

The following figure represents the CCM Serial Clock Multiplexer Register 2 (CSCMR2). The CSCMR2 register contains bits to control the multiplexers that generate the serial clocks.

NOTE

Any change to the above multiplexers must be done while the block whose clock is affected is not functional and the respective clock is gated in LPCG. If the change is made while the block is in the middle of an operation, it is not guaranteed that the module's operation will not be harmed.

Address: CCM_CSCMR2 is 53FD_4000h base + 20h offset = 53FD_4020h

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

CCM_CSCMR2 field descriptions

Field	Description
31–0 -	Reserved

5.4.10 CCM Serial Clock Divider Register 1 (CCM_CSCDR1)

The following figure represents the CCM Serial Clock Divider Register 1 (CSCDR1). The CSCDR1 register contains bits to control the clock generation submodule dividers.

NOTE

Any change to the above multiplexers will have to be done while the block whose clock is affected is not functional and the respective clock is gated in LPCG. If the change is made while the block is in the middle of an operation, it is not guaranteed that the module's operation will not be harmed.

When switching the divider from a slow clock source to a fast clock source, it is recommended to follow the steps below. The divider is not guaranteed to stop at the beginning of the divide sequence:

1. Change the divide value.
2. Enable the divider to allow the new divide to take effect.
3. Gate off divider, and switch to fast clock source.
4. Enable divider.

Address: CCM_CSCDR1 is 53FD_4000h base + 24h offset = 53FD_4024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-								ESDHC3_CLK_PRED [2:0]			ESDHC3_CLK_PODF[2:0]			ESDHC1_CLK_PRED [2:0]	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PGC_CLK_PODF[1:0]		ESDHC1_CLK_PODF[2:0]			-					UART_CLK_PRED[2:0]			UART_CLK_PODF[2:0]		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CSCDR1 field descriptions

Field	Description
31–25 -	Reserved
24–22 ESDHC3_CLK_PRED [2:0]	Divider for esdhc2 clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
21–19 ESDHC3_CLK_PODF[2:0]	Divider for esdhc3 clock podf. **NOTE:** Divider should be updated when output clock is gated.

Table continues on the next page...

CCM_CSCDR1 field descriptions (continued)

Field	Description
	000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
18–16 ESDHC1_CLK_ PRED [2:0]	Divider for esdhc1 clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
15–14 PGC_CLK_ PODF[1:0]	Divider for pgc (power gating controller) clock podf. **NOTE:** Divider should be updated when output clock is gated. 00 divide by 1 01 divide by 2 10 divide by 4 11 divide by 8
13–11 ESDHC1_CLK_ PODF[2:0]	Divider for esdhc1 clock podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
10–6 -	Reserved
5–3 UART_CLK_ PRED[2:0]	Divider for uart clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3

Table continues on the next page...

CCM_CSCDR1 field descriptions (continued)

Field	Description
	011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
2–0 UART_CLK_ PODF[2:0]	Divider for uart clock podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8

5.4.11 CCM SSI1 Clock Divider Register (CCM_CS1CDR)

The following figure represents the CCM SSI1 Clock Divider Register (CS1CDR). The CS1CDR register contains bits to control the ssi1 clock generation dividers.

Address: CCM_CS1CDR is 53FD_4000h base + 28h offset = 53FD_4028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-							SSI_EXT1_CLK_ PRED[2:0]			SSI_EXT1_CLK_PODF [5:0]					
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-							SSI1_CLK_ PRED[2:0]			SSI1_CLK_PODF [5:0]					
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

CCM_CS1CDR field descriptions

Field	Description
31–25 -	Reserved
24–22 SSI_EXT1_ CLK_PRED[2:0]	Divider for ssi_ext1 clock pred. 000 restricted 001 divide by 2 010 divide by 3

Table continues on the next page...

CCM_CS1CDR field descriptions (continued)

Field	Description
	011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
21–16 SSI_EXT1_ CLK_PODF [5:0]	Divider for ssi_ext1 clock podf. The input clock to this divider should be lower than 300 MHz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6
15–9 -	Reserved
8–6 SSI1_CLK_ PRED[2:0]	Divider for ssi1 clock pred. 000 restricted 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
5–0 SSI1_CLK_ PODF [5:0]	Divider for ssi1 clock podf. The input clock to this divider should be lower than 300 MHz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6

5.4.12 CCM SSI2 Clock Divider Register (CCM_CS2CDR)

The following figure represents the CCM SSI2 Clock Divider Register (CS2CDR). The CS2CDR register contains bits to control the ssi2 clock generation dividers.

Address: CCM_CS2CDR is 53FD_4000h base + 2Ch offset = 53FD_402Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-							SSI_EXT2_CLK_ PRED[2:0]			SSI_EXT2_CLK_PODF [5:0]					
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-							SSI2_CLK_ PRED[2:0]			SSI2_CLK_PODF [5:0]					
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

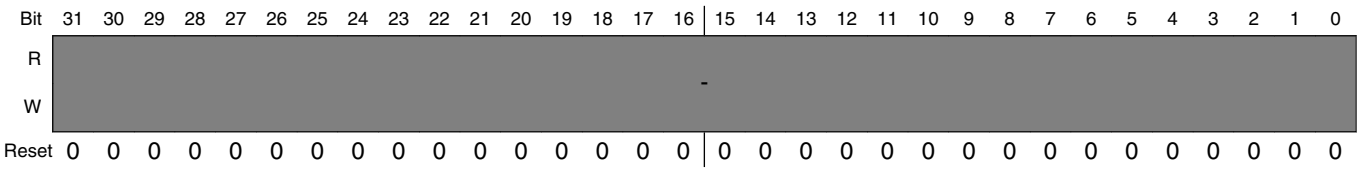
CCM_CS2CDR field descriptions

Field	Description
31–25 -	Reserved
24–22 SSI_EXT2_ CLK_PRED[2:0]	Divider for ssi_ext2 clock pred. 000 Restricted 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
21–16 SSI_EXT2_ CLK_PODF [5:0]	Divider for ssi_ext2 clock podf. The input clock to this divider should be lower than 300 MHz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6
15–9 -	Reserved
8–6 SSI2_CLK_ PRED[2:0]	Divider for ssi2 clock pred. 000 Restricted 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
5–0 SSI2_CLK_ PODF [5:0]	Divider for ssi2 clock podf. The input clock to this divider should be lower than 300 MHz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6

5.4.13 CCM DI Clock Divider Register (CCM_CDCDR)

The following figure represents the CCM DI Clock Divider Register (CDCDR). The CDCDR register contains bits to control the usb_phy generation dividers.

Address: CCM_CDCDR is 53FD_4000h base + 30h offset = 53FD_4030h



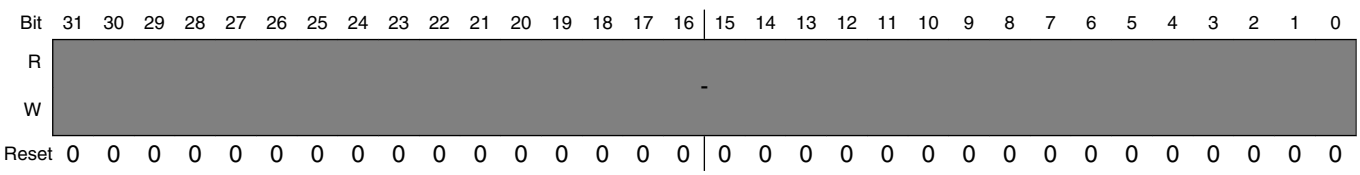
CCM_CDCDR field descriptions

Field	Description
31–0 -	Reserved

5.4.14 CCM HSC Clock Divider Register (CCM_CHSCCDR)

The following figure represents the CCM HSC Clock Divider Register (CHSCCDR). The CHSCCDR register contains bits to control the hsc clock generation dividers.

Address: CCM_CHSCCDR is 53FD_4000h base + 34h offset = 53FD_4034h



CCM_CHSCCDR field descriptions

Field	Description
31–0 -	Reserved

5.4.15 CCM Serial Clock Divider Register 2 (CCM_CSCDR2)

The following figure represents the CCM Serial Clock Divider Register 2 (CSCDR2). The CSCDR2 register contains bits to control the clock generation submodule dividers.

Address: CCM_CSCDR2 is 53FD_4000h base + 38h offset = 53FD_4038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-				ECSPI_CLK_ PRED[2:0]				ECSPI_CLK_PODF [5:0]					-		
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CSCDR2 field descriptions

Field	Description
31–28 -	Reserved
27–25 ECSPI_CLK_ PRED[2:0]	Divider for ecspi clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 Restricted 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
24–19 ECSPI_CLK_ PODF [5:0]	Divider for ecspi clock podf. **NOTE:** Divider should be updated when output clock is gated. The input clock to this divider should be lower than 300 MHz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6
18–0 -	Reserved

5.4.16 CCM Serial Clock Divider Register 3 (CCM_CSCDR3)

The following figure represents the CCM Serial Clock Divider Register 3 (CSCDR3). The CSCDR3 register contains bits to control the clock generation submodule dividers.

Address: CCM_CSCDR3 is 53FD_4000h base + 3Ch offset = 53FD_403Ch

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

CCM_CSCDR3 field descriptions

Field	Description
31–0 -	Reserved

5.4.17 CCM Serial Clock Divider Register 4 (CCM_CSCDR4)

The following figure represents the CCM Serial Clock Divider Register 4 (CSCDR4). The CSCDR4 register contains bits to control the clock generation submodule dividers.

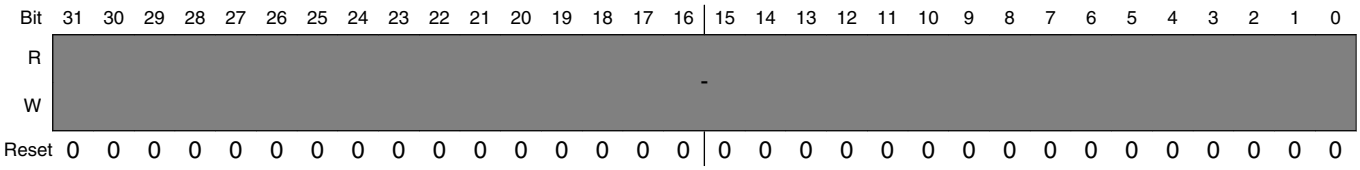
NOTE

Any change to the serial clock dividers (all dividers in the registers CSCDR1, CECDR, CSCDR2, CSCDR3 and CSCDR4) must be done while the block whose clock is affected is not functional. If the change is done while the block is in the middle of an operation, it is not guaranteed that the module's operation will not be harmed.

When switching the divider from a slow clock source to a fast clock source, these steps are recommended. The divider is not guaranteed to stop at the beginning of the divide sequence:

1. Change the divide value.
2. Enable the divider to allow the new divide to take effect.
3. Gate off divider, and switch to fast clock source.
4. Enable divider.

Address: CCM_CSCDR4 is 53FD_4000h base + 40h offset = 53FD_4040h



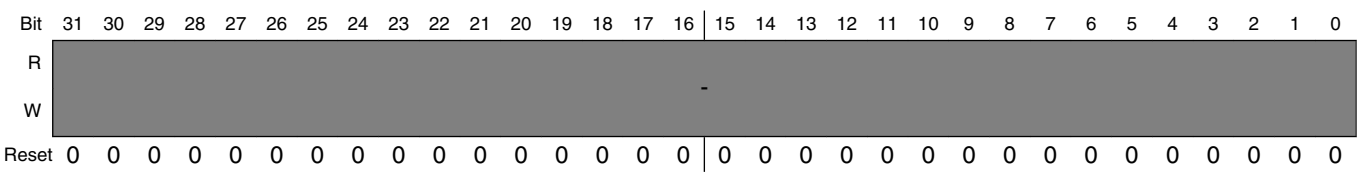
CCM_CSCDR4 field descriptions

Field	Description
31–0 -	Reserved

5.4.18 CCM Wakeup Detector Register (CCM_CWDR)

The following figure represents the CCM Wakeup Detector Register (CWDR). The CWDR register contains bits to control the functionality of the wakeup detector.

Address: CCM_CWDR is 53FD_4000h base + 44h offset = 53FD_4044h



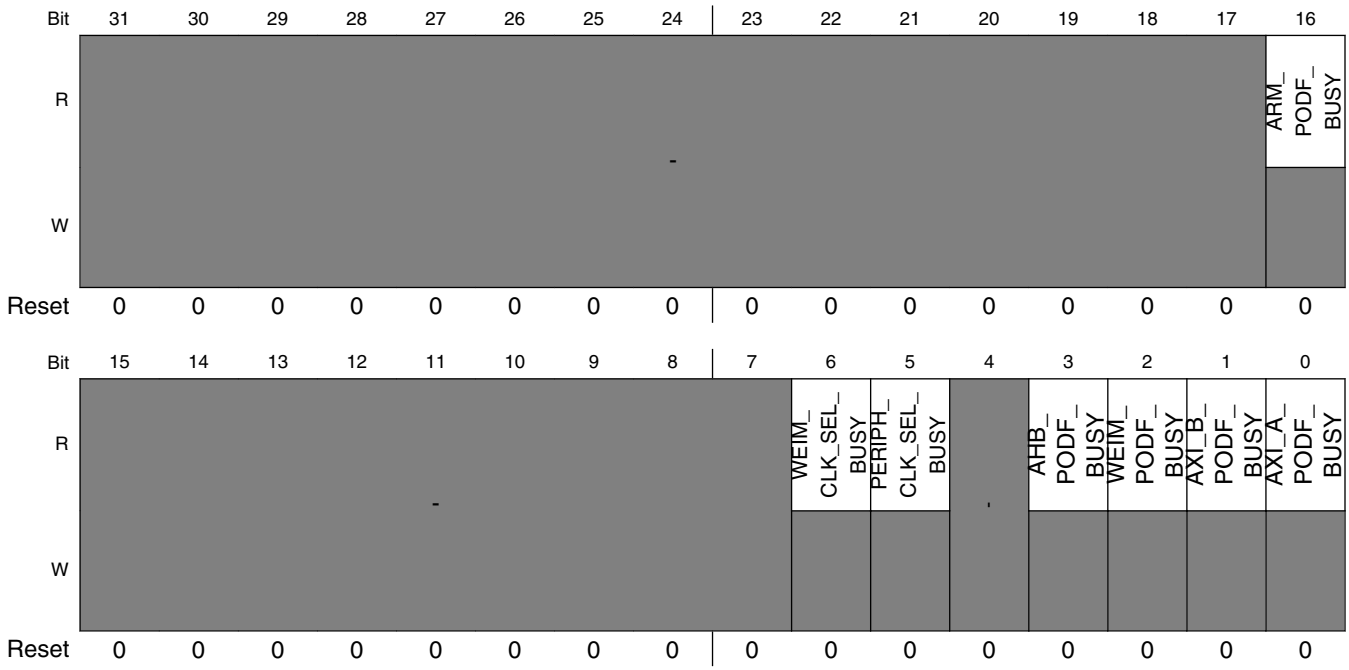
CCM_CWDR field descriptions

Field	Description
31–0 -	Reserved

5.4.19 CCM Divider Handshake In-Process Register (CCM_CDHIPR)

The following figure represents the CCM Divider Handshake In-Process Register (CDHIPR). The CDHIPR register contains read only bits that indicate that CCM is in process of updating dividers or muxes that might need handshake with modules.

Address: CCM_CDHIPR is 53FD_4000h base + 48h offset = 53FD_4048h



CCM_CDHIPR field descriptions

Field	Description
31–17 -	Reserved
16 ARM_PODF_BUSY	Busy indicator for arm_podf. This bit corresponds to the arm_podf divider only if ARM dvfs operation is done through arm_podf change (arm_freq_shift_divider = '1'). **NOTE:** Do not write arm_podf while arm_podf_busy is 1. This bit will not assert if arm_freq_shift_divider=0. **NOTE:** Transition from 1 to 0 can generate a ccm interrupt if not masked, refer to CIMR register description. 0 Divider is not busy and its value represents the actual division. 1 Divider is busy with handshake process. The value read in the divider represents the next value that the divider will hold after the handshake with gpc (after assertion of arm_clk_switch_req and the actual write to the divider).
15–7 -	Reserved

Table continues on the next page...

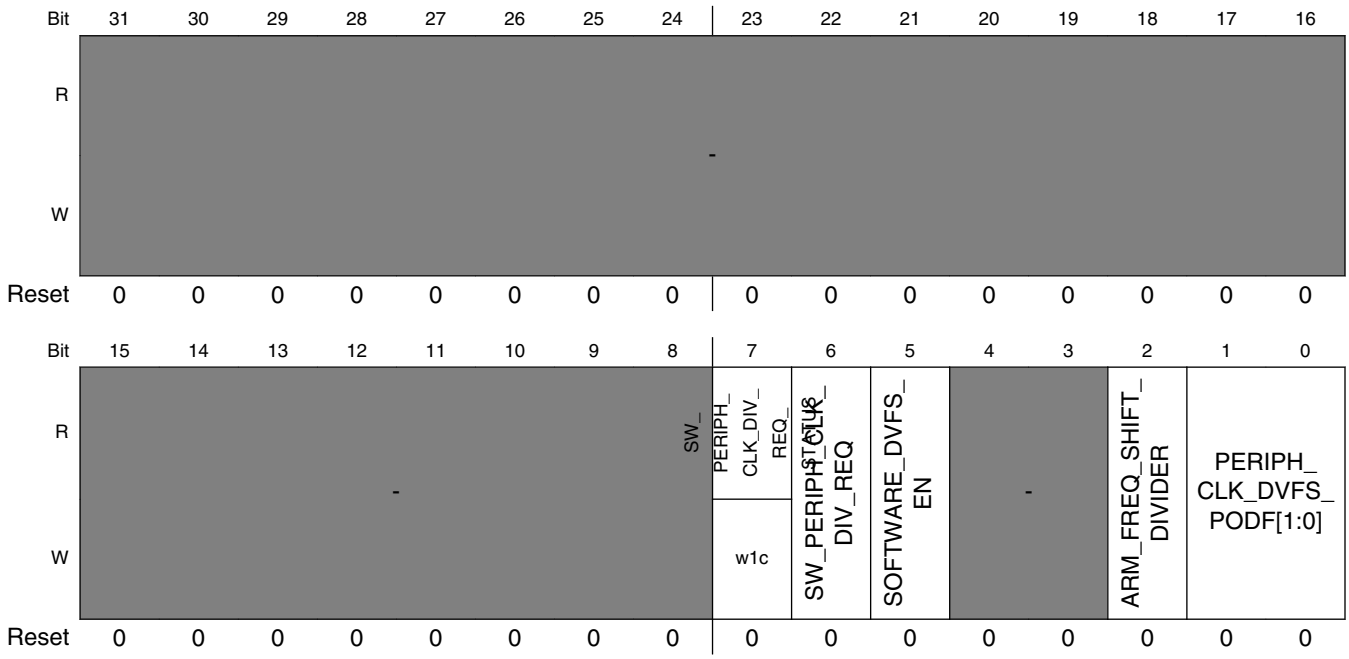
CCM_CDHIPR field descriptions (continued)

Field	Description
6 WEIM_CLK_ SEL_BUSY	Busy indicator for gcm weim_clk_sel. **NOTE:** Transition from 1 to 0 can generate a ccm interrupt if not masked, refer to CIMR register description. 0 mux is not busy. 1 mux is busy with handshake process.
5 PERIPH_CLK_ SEL_BUSY	Busy indicator for periph_clk_sel mux control. **NOTE:** Transition from 1 to 0 can generate a ccm interrupt if not masked, refer to CIMR register description. 0 mux is not busy . 1 mux is busy with handshake process.
4 -	Reserved
3 AHB_PODF_ BUSY	Busy indicator for ahb_podf. **NOTE:** Transition from 1 to 0 can generate a ccm interrupt if not masked, refer to CIMR register description. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the next value that the divider will hold after the handshake.
2 WEIM_PODF_ BUSY	Busy indicator for weim_podf. **NOTE:** Transition from 1 to 0 can generate a ccm interrupt if not masked, refer to CIMR register description. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the next value that the divider will hold after the handshake.
1 AXI_B_PODF_ BUSY	Busy indicator for axi_b_podf. **NOTE:** Transition from 1 to 0 can generate a ccm interrupt if not masked, refer to CIMR register description. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the next value that the divider will hold after the handshake.
0 AXI_A_PODF_ BUSY	Busy indicator for axi_a_podf. **NOTE:** Transition from 1 to 0 can generate a ccm interrupt if not masked, refer to CIMR register description. 0 Divider is not busy and its value represents the actual division. 1 Divider is busy with handshake process with module. The value read in the divider represents the next value that the divider will hold after the handshake.

5.4.20 CCM DVFS Control Register (CCM_CDCR)

The following figure represents the CCM DVFS Control Register (CDCR). The CDCR register contains bits to control the DVFS operation.

Address: CCM_CDCR is 53FD_4000h base + 4Ch offset = 53FD_404Ch



CCM_CDCR field descriptions

Field	Description
31–8 -	Reserved
7 SW_PERIPH_CLK_DIV_REQ_STATUS	Status bit to define the operation of DVFS driver in case of software control of DVFS divider. this bit is write '1' to clear - after a DVFS divider switch request it will be asserted and software should write '1' to clear it and prepare it for the next DVFS divider switch request. 1 DVFS frequency switch operation finished. 0 DVFS frequency switch is not finished, or there is no frequency switch request.
6 SW_PERIPH_CLK_DIV_REQ	Start DVFS frequency division operation. This bit will affect the DVFS divider only if software_dvfs_en bit was set to '1'. 0 Remove DVFS divider operation-DVFS divider will divide by '1'. 1 enable DVFS divider operation. DVFS divider will divide by the PERIPH_CLK_DVFS_PODF settings.
5 SOFTWARE_DVFS_EN	Defines if the DVFS operation will comence by software bit or by GPC signal "periph_clk_div_req". 0 DVFS operation will start by GPC signal "periph_clk_div_req" 1 DVFS operation will start by software bit sw_periphe_clk_div_req
4–3 -	reserved

Table continues on the next page...

CCM_CDCR field descriptions (continued)

Field	Description
2 ARM_FREQ_ SHIFT_DIVIDER	Define if next DVFS of ARM domain will be through podf change or pll relock. 0 Next ARM DVFS operation is done through pll relock. The pll relock process will start once arm_clk_switch_req is asserted. (in this case any new writes to arm_podf will be commences immediately, without waiting for arm_clk_switch_req). 1 Next ARM DVFS operation is done through arm_podf change. CCM will hold updates to arm_podf until signal arm_clk_switch_req is asserted.
1–0 PERIPH_CLK_ DVFS_ PODF[1:0]	Divider value for next operation of peripheral DVFS. This defines the value of the dividers affecting main_bus_clk. This divider will take place only during DVFS operation. Refer to Peripheral Clock Domain Frequency Change for details. 00 restricted 01 divide by 2 10 divide by 3 11 divide by 4

5.4.21 CCM Testing Observability Register (CCM_CTOR)

The following figure represents the CCM Testing Observability Register (CTOR). CCM includes three muxes to mux between different critical signals for testing observability. The output of the three muxes is generated on the three output signals obs_output_0, obs_output_1 and obs_output_2. Those three output signals can be generated on the IC pads by configuring the IOMUXC. The CTOR register contains bits to control the data generated for observability on the three output signals above.

Address: CCM_CTOR is 53FD_4000h base + 50h offset = 53FD_4050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																		OBS_EN	OBS_SPARE_OUTPUT_0_SEL					OBS_SPARE_OUTPUT_1_SEL				OBS_SPARE_OUTPUT_2_SEL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CTOR field descriptions

Field	Description
31–14 -	Reserved
13 OBS_EN	Observability enable bit. this bit enables the output of the three observability muxes. 0 Observability mux disabled. 1 Observability mux enabled.
12–8 OBS_SPARE_ OUTPUT_0_SEL	Selection of the signal to be generated on CCM_OUT_0 (output of CCM) for observability on the pads. 00000 ccm_system_in_stop_mode

Table continues on the next page...

CCM_CTOR field descriptions (continued)

Field	Description
	00001 lpm_current_state[0] 00010 hndsk_current_state[0] 00011 shd_current_state[0] 00100 ccm_ipg_stop 00101 ccm_pdn_4arm_req 00110 Reserved 00111 Reserved 01000 Reserved 01001 Reserved 01010 pll_lrf_sticky1 01011 Reserved 01100 clk_src_on 01101 Reserved 01110 src_warm_dvfs_req 01111 periph_clk_div_req 10000 arm_clk_switch_req 10001 ccm_clk_switch_ack 10010 Reserved 10011 weim_lpmd 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Reserved 11001 obs_input_6 11010 obs_input_0 11011 obs_input_1 11100 obs_input_2 11101 obs_input_3 11110 obs_input_4 11111 obs_input_5
7–4 OBS_SPARE_ OUTPUT_1_SEL	Selection of the signal to be generated on obs_output_1 (output of CCM) for observability on the pads. 0000 ccm_system_in_wait_mode 0001 lpm_current_state[1] 0010 hndsk_current_state[1] 0011 Reserved 0100 ccm_ipg_wait 0101 ccm_camp_dis 0110 dpll_en_dpllip 0111 ccm_pdn_4all_req 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 arm_dsm_request 1101 Reserved

Table continues on the next page...

CCM_CTOR field descriptions (continued)

Field	Description
	1110 gpc_pup_ack 1111 pll_lrf_sticky2
3–0 OBS_SPARE_ OUTPUT_2_SEL	Selection of the signal to be generated on obs_output_2 (output of CCM) for observability on the pads. 0000 Reserved 0001 lpm_current_state[2] 0010 hndsk_current_state[2] 0011 shd_current_state[1] 0100 pll_lvs 0101 src_clock_ready 0110 ref_clk_en_dpllip 0111 ccm_pup_req 1000 weim_lpack 1001 Reserved 1010 Reserved 1011 Reserved 1100 src_power_gating_reset_done 1101 tzic_dsm_wakeup 1110 gpc_pdn_ack 1111 pll_lrf_sticky3

5.4.22 CCM Low Power Control Register (CCM_CLPCR)

The following figure represents the CCM Low Power Control Register (CLPCR). The CLPCR register contains bits to control the low power modes operation.

Address: CCM_CLPCR is 53FD_4000h base + 54h offset = 53FD_4054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-						BYPASS_MAX_LPM_HS	BYPASS_SDMA_LPM_HS	BYPASS_RNGB_LPM_HS	-			BYPASS_WEIM_LPM_HS	-		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-				COSC_PWRDOWN	STBY_COUNT		VSTBY	DIS_REF_OSC	SBYOS	ARM_CLK_DIS_ON_LPM	-		BYPASS_PMIC_VFUNCTIONAL_READY	LPM[1:0]	
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1

CCM_CLPCR field descriptions

Field	Description
31–26 -	Reserved
25 BYPASS_MAX_LPM_HS	Bypass handshake with the ahbmax on next entrance to low power mode (wait or stop mode). CCM does not wait for the module's acknowledge. 0 handshake with max on next entrance to low power mode will be performed. 1 handshake with max on next entrance to low power mode will be bypassed.
24 BYPASS_SDMA_LPM_HS	Bypass handshake with sdma on next entrance to low power mode (wait or stop mode). CCM does not wait for the module's acknowledge. 0 handshake with sdma on next entrance to low power mode will be performed. 1 handshake with sdma on next entrance to low power mode will be bypassed.
23 BYPASS_RNGB_LPM_HS	Bypass handshake with rngb on next entrance to low power mode (wait or stop mode). CCM does not wait for the module's acknowledge. 0 handshake with rngb on next entrance to low power mode will be performed. 1 handshake with rngb on next entrance to low power mode will be bypassed.
22–20 -	Reserved
19 BYPASS_WEIM_LPM_HS	Bypass handshake with weimon next entrance to low power mode (wait or stop mode). CCM does not wait for the module's acknowledge. 0 handshake with sdma on next entrance to low power mode will be performed. 1 handshake with sdma on next entrance to low power mode will be bypassed.
18–12 -	Reserved
11 COSC_PWRDOWN	In run mode, software can manually control powering down of on chip oscillator, i.e. generating '1' on cosc_pwrdown signal. If software manually powered down the on chip oscillator, then sbyos functionality for on chip oscillator will be bypassed. The manual closing of onchip oscillator should be performed only in case the reference oscilator is not the source of all the clocks generation. 0 On chip oscillator will not be powered down, i.e. cosc_pwrdown = '0'. 1 On chip oscillator will be powered down, i.e. cosc_pwrdown = '1'.
10–9 STBY_COUNT	Standby counter definition. These two bits define, in the case of stop exit (if vstby bit was set), the amount of time CCM will wait between PMIC_VSTBY_REQ negation and the check of assertion of PMIC_VFUNCIONAL_READY. If bypass_pmic_vfunctional_ready is set, STBY_COUNT is the amount of time the CCM will wait between PMIC_VSTBY_REQ negation and exiting STOP mode. 00 CCM will wait 2 ckil clock cycles 01 CCM will wait 4 ckil clock cycles 10 CCM will wait 8 ckil clock cycles 11 CCM will wait 16 ckil clock cycles
8 VSTBY	Voltage standby request bit. This bit defines if PMIC_VSTBY_REQ pin, which notifies external power management IC to move from functional voltage to standby voltage, will be asserted in stop mode. **NOTE:** When returning from stop mode, the PMIC_VSTBY_REQ will be deasserted (if it was asserted when entering stop mode), and CCM will wait for indication that functional voltage is ready (by

Table continues on the next page...

CCM_CLPCR field descriptions (continued)

Field	Description
	sampling the assertion of pmic_vfuncional_ready) before continuing the process of exiting from stop mode. Please refer to stby_count bits. 0 voltage will not be changed to standby voltage after next entrance to stop mode. (PMIC_VSTBY_REQ will remain negated - '0') 1 voltage will be requested to change to standby voltage after next entrance to stop mode. (PMIC_VSTBY_REQ will be asserted - '1').
7 DIS_REF_OSC	dis_ref_osc - in run mode, software can manually control closing of external reference oscillator clock, i.e. generating '1' on ref_en_b signal. If software closed manually the external reference clock, then sbyos functionality will be bypassed. The manual closing of external reference oscilator should be performed only in case the reference oscilator is not the source of any clock generation. 0 external high frequency oscillator will be enabled, i.e. ref_en_b = '0'. 1 external high frequency oscillator will be disabled, i.e. ref_en_b = '1'
6 SBYOS	Standby clock oscillator bit. This bit defines if REF_EN_B pin, which disables external high frequency crystal, and cosc_pwrdown, which power down the on chip oscillator, will be asserted in stop mode. This bit is discarded if dis_ref_osc = '1' for external oscillator, and if cosc_pwrdown='1' for the on chip oscillator. 0 external high frequency oscillator will not be disabled and on chip oscillator will not be powered down, after next entrance to stop mode. (ref_en_b will remain asserted - '0' and cosc_pwrdown will remain de asserted - '0') 1 external high frequency oscillator will be disabled and on chip oscillator will be powered down, after next entrance to stop mode. (ref_en_b will be deasserted - '1' and cosc_pwrdown will be asserted - '1'). When returning from stop mode, external oscillator will be enabled again, on chip oscillator will return to oscillator mode , and after oscnt count ccm will continue with the exit from stop mode process.
5 ARM_CLK_DIS_ON_LPM	Define if ARM clocks (arm_clk, soc_mxclk, soc_pclk, soc_dbg_pclk, vl_wrck) will be disabled on wait mode. This is useful for debug mode, when the user still wants to simulate entering wait mode and still keep ARM clock functioning. **NOTE:** Software should not enable ARM power gating in wait mode if this bit is cleared. 0 ARM clock enabled on wait mode. 1 ARM clock disabled on wait mode.
4–3 -	Reserved
2 BYPASS_PMIC_VFUNCTIONAL_READY	By asserting this bit CCM will bypass waiting for pmic_vfunctional_ready signal when coming out of STOP mode. This should be used for PMIC's that don't support the pmic_vfunctional_ready signal. 0 Don't bypass the pmic_vfunctional_ready signal; CCM will wait for its assertion during exit of low power mode if standby voltage was enabled. 1 Bypass the pmic_vfunctional_ready signal; CCM will not wait for its assertion during exit of low power mode if standby voltage was enabled. Note that the CCM will wait for STBY_COUNT cycles after PMIC_VSTBY_REQ negation before exiting STOP mode.
1–0 LPM[1:0]	Setting the low power mode that system will enter on next assertion of dsm_request signal. 00 Remain in run mode 01 Transfer to wait mode

Table continues on the next page...

CCM_CLPCR field descriptions (continued)

Field	Description
	10 Transfer to stop mode 11 Reserved

5.4.23 CCM Interrupt Status Register (CCM_CISR)

The following figure represents the CCM Interrupt Status Register (CISR). This is a write one to clear register. Once a interrupt is generated, software should write one to clear it.

Address: CCM_CISR is 53FD_4000h base + 58h offset = 53FD_4058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-					ARM_PODF_LOADED	TEMP_MON_ALARM	-	WEIM_CLK_SEL_LOADED	PERIPH_CLK_SEL_LOADED	-	AHB_PODF_LOADED	WEIM_PODF_LOADED	AXI_B_PODF_LOADED	AXI_A_PODF_LOADED	DIVIDERS_LOADED
W						w1c	w1c		w1c	w1c		w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-									COSC_READY	-	CAMP1_READY	-	LRF_PLL3	LRF_PLL2	LRF_PLL1
W										w1c	w1c	w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CISR field descriptions

Field	Description
31–27 -	Reserved
26 ARM_PODF_LOADED	Interrupt ipi_int_1 generated due to frequency change of arm_podf. The interrupt will commence only if arm_podf is loaded during a arm dvfs operation. 0 interrupt is not generated due to frequency change of arm_podf 1 interrupt generated due to frequency change of arm_podf

Table continues on the next page...

CCM_CISR field descriptions (continued)

Field	Description
25 TEMP_MON_ ALARM	Interrupt ipi_int_1 generated due to "alarm" event by the temperature monitor (i.e. on exceeding temperature event): 0 no interrupt. 1 interrupt generated by rise edge of "temp_mon_alarm" signal. (i.e. temp_monitor "alarm" interrupt event).
24 -	Reserved
23 WEIM_CLK_ SEL_LOADED	Interrupt ipi_int_1 generated due to update of weim_clk_sel. 0 Interrupt is not generated due to update of weim_clk_sel. 1 Interrupt generated due to update of weim_clk_sel.
22 PERIPH_CLK_ SEL_LOADED	Interrupt ipi_int_1 generated due to update of periph_clk_sel. 0 interrupt is not generated due to update of periph_clk_sel. 1 interrupt generated due to update of periph_clk_sel.
21 -	Reserved
20 AHB_PODF_ LOADED	Interrupt ipi_int_1 generated due to frequency change of ahb_podf 0 interrupt is not generated due to frequency change of ahb_podf 1 interrupt generated due to frequency change of ahb_podf
19 WEIM_PODF_ LOADED	Interrupt ipi_int_1 generated due to frequency change of weim_podf 0 interrupt is not generated due to frequency change of weim_podf 1 interrupt generated due to frequency change of weim_podf
18 AXI_B_PODF_ LOADED	Interrupt ipi_int_1 generated due to frequency change of axi_b_podf 0 interrupt is not generated due to frequency change of axi_b_podf 1 interrupt generated due to frequency change of axi_b_podf
17 AXI_A_PODF_ LOADED	Interrupt ipi_int_1 generated due to frequency change of axi_a_podf 0 interrupt is not generated due to frequency change of axi_a_podf 1 interrupt generated due to frequency change of axi_a_podf
16 DIVIDERS_ LOADED	Interrupt ipi_int_1 generated due to updated of axi_a_podf, axi_b_podf, weim_podf, ahb_podf, periph_clk_sel, weim_clk_sel. 0 interrupt is not generated due to updated of axi_a_podf, axi_b_podf, weim_podf, ahb_podf, periph_clk_sel, weim_clk_sel. 1 interrupt generated due to updated of axi_a_podf, axi_b_podf, weim_podf, ahb_podf, periph_clk_sel, weim_clk_sel.
15–7 -	Reserved
6 COSC_READY	Interrupt ipi_int_2 generated due to on board oscillator ready, i.e. oscnt has finished counting. 0 interrupt is not generated due to on board oscillator ready 1 interrupt generated due to on board oscillator ready

Table continues on the next page...

CCM_CISR field descriptions (continued)

Field	Description
5 -	Reserved
4 CAMP1_READY	Interrupt ipi_int_2 generated due to camp1 ready, i.e. oscnt has finished counting. 0 interrupt is not generated due to camp1 ready 1 interrupt generated due to camp1 ready
3 -	Reserved
2 LRF_PLL3	Interrupt ipi_int_2 generated due to lock of pll3 0 interrupt is not generated due to lock ready of pll_3 1 interrupt generated due to lock ready of pll_3
1 LRF_PLL2	Interrupt ipi_int_2 generated due to lock of pll2 0 interrupt is not generated due to lock ready of pll_2 1 interrupt generated due to lock ready of pll_2
0 LRF_PLL1	Interrupt ipi_int_2 generated due to lock of pll1 0 interrupt is not generated due to lock ready of pll_1 1 interrupt generated due to lock ready of pll_1

5.4.24 CCM Interrupt Mask Register (CCM_CIMR)

The following figure represents the CCM Interrupt Mask Register (CIMR).

Address: CCM_CIMR is 53FD_4000h base + 5Ch offset = 53FD_405Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-					ARM_PODF_LOADED	TEMP_MON_ALARM	-	MASK_WEIM_CLK_SEL_LOADED	MASK_PERIPH_CLK_SEL_LOADED	-	MASK_AHB_PODF_LOADED	MASK_WEIM_PODF_LOADED	MASK_AXI_B_PODF_LOADED	MASK_AXI_A_PODF_LOADED	MASK_DIVIDERS_LOADED
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-									MASK_COSC_READY	-	MASK_CAMP1_READY	-	MASK_LRF_PLL3	MASK_LRF_PLL2	MASK_LRF_PLL1
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CIMR field descriptions

Field	Description
31–27 -	Reserved
26 ARM_PODF_ LOADED	mask interrupt generation due to frequency change of arm_podf 0 don't mask interrupt due to frequency change of arm_podf - interrupt will be created 1 mask interrupt due to frequency change of arm_podf
25 TEMP_MON_ ALARM	mask interrupt generation due to temperature monitor "alarm" event. (i.e. exceeding temperature) 0 Interrupt unmasked. 1 Temperature monitor event interrupt is masked.
24 -	Reserved
23 MASK_WEIM_ CLK_SEL_ LOADED	mask interrupt generation due to update of weim_clk_sel. 0 don't mask interrupt due to update of weim_clk_sel - interrupt will be created 1 mask interrupt due to update of weim_clk_sel
22 MASK_PERIPH_ CLK_SEL_ LOADED	mask interrupt generation due to update of periph_clk_sel. 0 don't mask interrupt due to update of periph_clk_sel - interrupt will be created 1 mask interrupt due to update of periph_clk_sel
21 -	Reserved
20 MASK_AHB_ PODF_LOADED	mask interrupt generation due to frequency change of ahb_podf 0 don't mask interrupt due to frequency change of ahb_podf - interrupt will be created 1 mask interrupt due to frequency change of ahb_podf
19 MASK_WEIM_ PODF_LOADED	mask interrupt generation due to frequency change of weim_podf 0 don't mask interrupt due to frequency change of weim_podf - interrupt will be created 1 mask interrupt due to frequency change of weim_podf
18 MASK_AXI_B_ PODF_LOADED	mask interrupt generation due to frequency change of axi_b_podf 0 don't mask interrupt due to frequency change of axi_b_podf - interrupt will be created 1 mask interrupt due to frequency change of axi_b_podf
17 MASK_AXI_A_ PODF_LOADED	mask interrupt generation due to frequency change of axi_a_podf 0 don't mask interrupt due to frequency change of axi_a_podf - interrupt will be created 1 mask interrupt due to frequency change of axi_a_podf
16 MASK_ DIVIDERS_ LOADED	mask interrupt generation due to updated of axi_a_podf, axi_b_podf, weim_podf, ahb_podf, periph_clk_sel, weim_clk_sel. 0 don't mask interrupt due to updated of axi_a_podf, axi_b_podf, weim_podf, ahb_podf, periph_clk_sel, weim_clk_sel. 1 mask interrupt due to updated of axi_a_podf, axi_b_podf, weim_podf, ahb_podf, periph_clk_sel, weim_clk_sel.
15–7 -	Reserved

Table continues on the next page...

CCM_CIMR field descriptions (continued)

Field	Description
6 MASK_COSC_READY	mask interrupt generation due to on board oscillator ready 0 don't mask interrupt due to on board oscillator ready - interrupt will be created 1 mask interrupt due to on board oscillator ready
5 -	Reserved
4 MASK_CAMP1_READY	mask interrupt generation due to camp1 ready 0 don't mask interrupt due to camp1 ready - interrupt will be created 1 mask interrupt due to camp1 ready
3 -	Reserved
2 MASK_LRF_PLL3	mask interrupt generation due to lrf of pll3 0 don't mask interrupt due to lrf of pll3 - interrupt will be created 1 mask interrupt due to lrf of pll3
1 MASK_LRF_PLL2	mask interrupt generation due to lrf of pll2 0 don't mask interrupt due to lrf of pll2 - interrupt will be created 1 mask interrupt due to lrf of pll2
0 MASK_LRF_PLL1	mask interrupt generation due to lrf of pll1 0 don't mask interrupt due to lrf of pll1 - interrupt will be created 1 mask interrupt due to lrf of pll1

5.4.25 CCM Clock Output Source Register (CCM_CCOSR)

The following figure represents the CCM Clock Output Source Register (CCOSR). The CCOSR register contains bits to control the clocks that will be generated on the output ipp_do_clko1 and ipp_do_clko2.

Address: CCM_CCOSR is 53FD_4000h base + 60h offset = 53FD_4060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-							CKO2_EN	CKO2_DIV[2:0]			CKO2_SEL[4:0]				
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-							CKO1_SLOW_SEL	CKO1_EN	CKO1_DIV[2:0]			CKO1_SEL[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

CCM_CCOSR field descriptions

Field	Description
31–25 -	Reserved
24 CKO2_EN	Enable of CKO2 clock 0 CKO2 disabled. 1 CKO2 enabled.
23–21 CKO2_DIV[2:0]	Setting the divider of CKO2 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
20–16 CKO2_SEL[4:0]	Selection of the clock to be generated on cko2 00000 Reserved 00001 Reserved 00010 display_axi_clk_root 00011 esdhc1_clk_root 00100 Reserved 00101 wrck_clk_root 00110 ecspi_clk_root 00111 pll1_ref_clk 01000 esdhc3_clk_root 01001 ddr_clk_root 01010 Reserved 01011 usbphy_pll_out_480 (only for USB_PHY1) 01100 gpu2d_clk_root 01101 debug_apb_clk_root 01110 osc_clk 01111 ckih_camp1_clk 10000 esdhc4_clk_root 10001 esdhc2_clk_root 10010 ssi1_clk_root 10011 ssi2_clk_root 10100 gpmi_clk_root 10101 epdc_axi_clk_root 10110 Reserved 10111 pgc_clk_root 11000 bch_clk_root 11001 usb_phy_clk_root 11010 Reserved 11011 lp_apm clock 11100 uart_clk_root

Table continues on the next page...

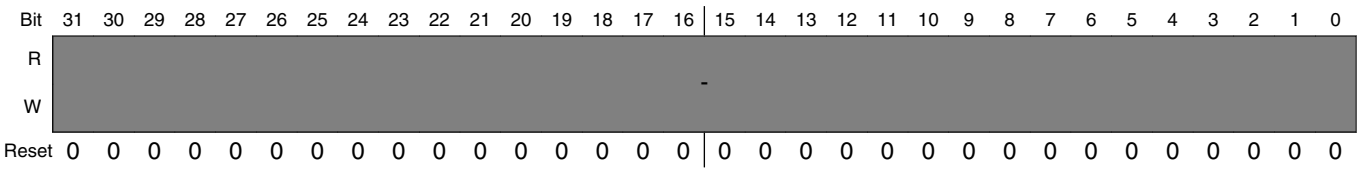
CCM_CCOSR field descriptions (continued)

Field	Description
	11101 sys_clk 11110 mshc_xmscki_clk_root 11111 async_ref_clock_obs div 8 (connected in SOC to async ref clock for reference clock generation of ARM) This clock source is always divided by 8 before supplying this clock to the cko2 mux.
15–9 -	Reserved
8 CKO1_SLOW_SEL	Select of CKO1 slow clocks 0 CKO1 fast clocks. 1 CKO1 slow clocks.
7 CKO1_EN	Enable of CKO1 clock 0 CKO1 disabled. 1 CKO1 enabled.
6–4 CKO1_DIV[2:0]	Setting the divider of CKO1 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
3–0 CKO1_SEL[3:0]	Selection of the clock to be generated on cko1 fast clocks xx00 arm_clk_root xx01 pll1_sw_clk xx10 pll2_sw_clk xx11 pll3_sw_clk 0000 ref_480pll_clk 0001 ref_pfd0_clk 0010 ref_pfd1_clk 0011 ref_pfd2_clk 0100 ref_pfd3_clk 0101 ref_pfd4_clk 0110 ref_pfd5_clk 0111 ref_pfd6_clk 1000 epdc_pix_clk_root 1001 elcdif_pix_clk_root 1010 weim_clk_root 1011 ahb_clk_root 1100 ipg_clk_root 1101 perclk_root 1110 ckil_sync_clk_root 1111 ref_pfd7_clk

5.4.26 CCM General Purpose Register (CCM_CGPR)

The following figure represents the CCM General Purpose Register (CGPR). This is a regular read/write implemented register, which can serve for future possible usage. The respective bits are connected to cgpr_dout[31:0] output of CCM. Several bits are already in use, while other bits are free for future usage.

Address: CCM_CGPR is 53FD_4000h base + 64h offset = 53FD_4064h



CCM_CGPR field descriptions

Field	Description
31–0 -	Reserved for future use. Those bits are connected to ccm output cgpr_dout[31-0]

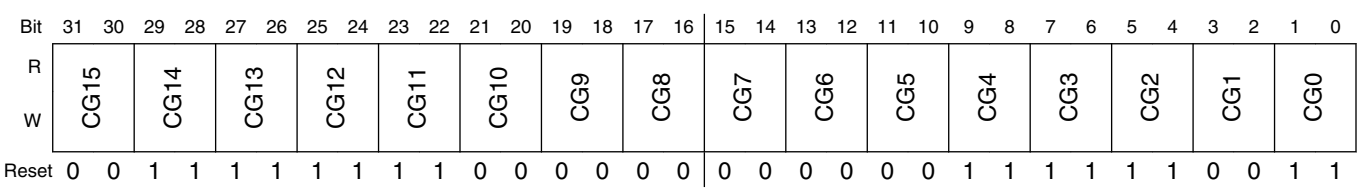
5.4.27 CCM Clock Gating Register 0 (CCM_CCGR0)

The following figure represents CCM Clock Gating Register 0 (CCGR0). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are eight CGR registers. The number of registers required corresponds to the number of peripherals in the system.

NOTE

Additional information about the bit fields of this register can be found in Table 5-55.

Address: CCM_CCGR0 is 53FD_4000h base + 68h offset = 53FD_4068h



CCM_CCGR0 field descriptions

Field	Description
31–30 CG15	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes

Table continues on the next page...

CCM_CCGR0 field descriptions (continued)

Field	Description
	10 Clock is always on 11 Clock is always on except in STOP mode
29–28 CG14	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
27–26 CG13	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
25–24 CG12	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
23–22 CG11	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
21–20 CG10	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
19–18 CG9	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
17–16 CG8	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
15–14 CG7	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode.

Table continues on the next page...

CCM_CCGR0 field descriptions (continued)

Field	Description
	01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
13–12 CG6	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
11–10 CG5	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
9–8 CG4	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
7–6 CG3	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
5–4 CG2	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
3–2 CG1	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
1–0 CG0	Defines clock gating for power reduction of clock see Table 5-55. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

5.4.28 CCM Clock Gating Register 1 (CCM_CCGR1)

The following figure represents CCM Clock Gating Register 1 (CCGR1). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required corresponds to the number of peripherals in the system.

NOTE

Additional information about the bit fields of this register can be found in Table 5-56.

Address: CCM_CCGR1 is 53FD_4000h base + 6Ch offset = 53FD_406Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8		CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1

CCM_CCGR1 field descriptions

Field	Description
31–30 CG15	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
29–28 CG14	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
27–26 CG13	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
25–24 CG12	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

Table continues on the next page...

CCM_CCGR1 field descriptions (continued)

Field	Description
23–22 CG11	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
21–20 CG10	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
19–18 CG9	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
17–16 CG8	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
15–14 CG7	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
13–12 CG6	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
11–10 CG5	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
9–8 CG4	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes

Table continues on the next page...

CCM_CCGR1 field descriptions (continued)

Field	Description
	10 Clock is always on 11 Clock is always on except in STOP mode
7–6 CG3	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
5–4 CG2	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
3–2 CG1	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
1–0 CG0	Defines clock gating for power reduction of clock see Table 5-56. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

5.4.29 CCM Clock Gating Register 2 (CCM_CCGR2)

The following figure represents CCM Clock Gating Register 2 (CCGR2). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required corresponds to the number of peripherals in the system.

NOTE

Additional information about the bit fields of this register can be found in Table 5-57.

Address: CCM_CCGR2 is 53FD_4000h base + 70h offset = 53FD_4070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8		CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CCGR2 field descriptions

Field	Description
31–30 CG15	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
29–28 CG14	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
27–26 CG13	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
25–24 CG12	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
23–22 CG11	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
21–20 CG10	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
19–18 CG9	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
17–16 CG8	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes

Table continues on the next page...

CCM_CCGR2 field descriptions (continued)

Field	Description
	10 Clock is always on 11 Clock is always on except in STOP mode
15–14 CG7	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
13–12 CG6	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
11–10 CG5	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
9–8 CG4	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
7–6 CG3	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
5–4 CG2	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
3–2 CG1	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
1–0 CG0	Defines clock gating for power reduction of clock see Table 5-57. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode.

Table continues on the next page...

CCM_CCGR2 field descriptions (continued)

Field	Description
	01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

5.4.30 CCM Clock Gating Register 3 (CCM_CCGR3)

The following figure represents CCM Clock Gating Register 3 (CCGR3). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required corresponds to the number of peripherals in the system.

NOTE

Additional information about the bit fields of this register can be found in Table 5-58.

Address: CCM_CCGR3 is 53FD_4000h base + 74h offset = 53FD_4074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8		CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

CCM_CCGR3 field descriptions

Field	Description
31–30 CG15	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
29–28 CG14	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
27–26 CG13	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

Table continues on the next page...

CCM_CCGR3 field descriptions (continued)

Field	Description
25–24 CG12	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
23–22 CG11	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
21–20 CG10	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
19–18 CG9	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
17–16 CG8	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
15–14 CG7	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
13–12 CG6	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
11–10 CG5	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes

Table continues on the next page...

CCM_CCGR3 field descriptions (continued)

Field	Description
	10 Clock is always on 11 Clock is always on except in STOP mode
9–8 CG4	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
7–6 CG3	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
5–4 CG2	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
3–2 CG1	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
1–0 CG0	Defines clock gating for power reduction of clock see Table 5-58. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

5.4.31 CCM Clock Gating Register 4 (CCM_CCGR4)

The following figure represents CCM Clock Gating Register 4 (CCGR4). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required corresponds to the number of peripherals in the system.

NOTE

Additional information about the bit fields of this register can be found in Table 5-59.

Address: CCM_CCGR4 is 53FD_4000h base + 78h offset = 53FD_4078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8		CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CCGR4 field descriptions

Field	Description
31–30 CG15	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
29–28 CG14	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
27–26 CG13	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
25–24 CG12	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
23–22 CG11	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
21–20 CG10	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
19–18 CG9	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes

Table continues on the next page...

CCM_CCGR4 field descriptions (continued)

Field	Description
	10 Clock is always on 11 Clock is always on except in STOP mode
17–16 CG8	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
15–14 CG7	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
13–12 CG6	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
11–10 CG5	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
9–8 CG4	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
7–6 CG3	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
5–4 CG2	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
3–2 CG1	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode.

Table continues on the next page...

CCM_CCGR4 field descriptions (continued)

Field	Description
	01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
1–0 CG0	Defines clock gating for power reduction of clock see Table 5-59. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

5.4.32 CCM Clock Gating Register 5 (CCM_CCGR5)

The following figure represents CCM Clock Gating Register 5 (CCGR5). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required corresponds to the number of peripherals in the system.

NOTE

Additional information about the bit fields of this register can be found in Table 5-60.

Address: CCM_CCGR5 is 53FD_4000h base + 7Ch offset = 53FD_407Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8		CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CCGR5 field descriptions

Field	Description
31–30 CG15	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
29–28 CG14	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

Table continues on the next page...

CCM_CCGR5 field descriptions (continued)

Field	Description
27–26 CG13	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
25–24 CG12	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
23–22 CG11	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
21–20 CG10	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
19–18 CG9	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
17–16 CG8	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
15–14 CG7	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
13–12 CG6	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes

Table continues on the next page...

CCM_CCGR5 field descriptions (continued)

Field	Description
	10 Clock is always on 11 Clock is always on except in STOP mode
11–10 CG5	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
9–8 CG4	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
7–6 CG3	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
5–4 CG2	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
3–2 CG1	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
1–0 CG0	Defines clock gating for power reduction of clock see Table 5-60. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

5.4.33 CCM Clock Gating Register 6 (CCM_CCGR6)

The following figure represents CCM Clock Gating Register 6 (CCGR6). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required corresponds to the number of peripherals in the system.

NOTE

Additional information about the bit fields of this register can be found in Table 5-61.

Address: CCM_CCGR6 is 53FD_4000h base + 80h offset = 53FD_4080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8		CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

CCM_CCGR6 field descriptions

Field	Description
31–30 CG15	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
29–28 CG14	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
27–26 CG13	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
25–24 CG12	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
23–22 CG11	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
21–20 CG10	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes

Table continues on the next page...

CCM_CCGR6 field descriptions (continued)

Field	Description
	10 Clock is always on 11 Clock is always on except in STOP mode
19–18 CG9	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
17–16 CG8	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
15–14 CG7	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
13–12 CG6	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
11–10 CG5	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
9–8 CG4	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
7–6 CG3	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
5–4 CG2	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode.

Table continues on the next page...

CCM_CCGR6 field descriptions (continued)

Field	Description
	01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
3–2 CG1	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
1–0 CG0	Defines clock gating for power reduction of clock see Table 5-61. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

5.4.34 CCM Clock Gating Register 7 (CCM_CCGR7)

The following figure represents CCM Clock Gating Register 7 (CCGR7). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required corresponds to the number of peripherals in the system.

NOTE

Additional information about the bit fields of this register can be found in Table 5-62.

Address: CCM_CCGR7 is 53FD_4000h base + 84h offset = 53FD_4084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8		CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CCGR7 field descriptions

Field	Description
31–30 CG15	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

Table continues on the next page...

CCM_CCGR7 field descriptions (continued)

Field	Description
29–28 CG14	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
27–26 CG13	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
25–24 CG12	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
23–22 CG11	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
21–20 CG10	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
19–18 CG9	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
17–16 CG8	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
15–14 CG7	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes

Table continues on the next page...

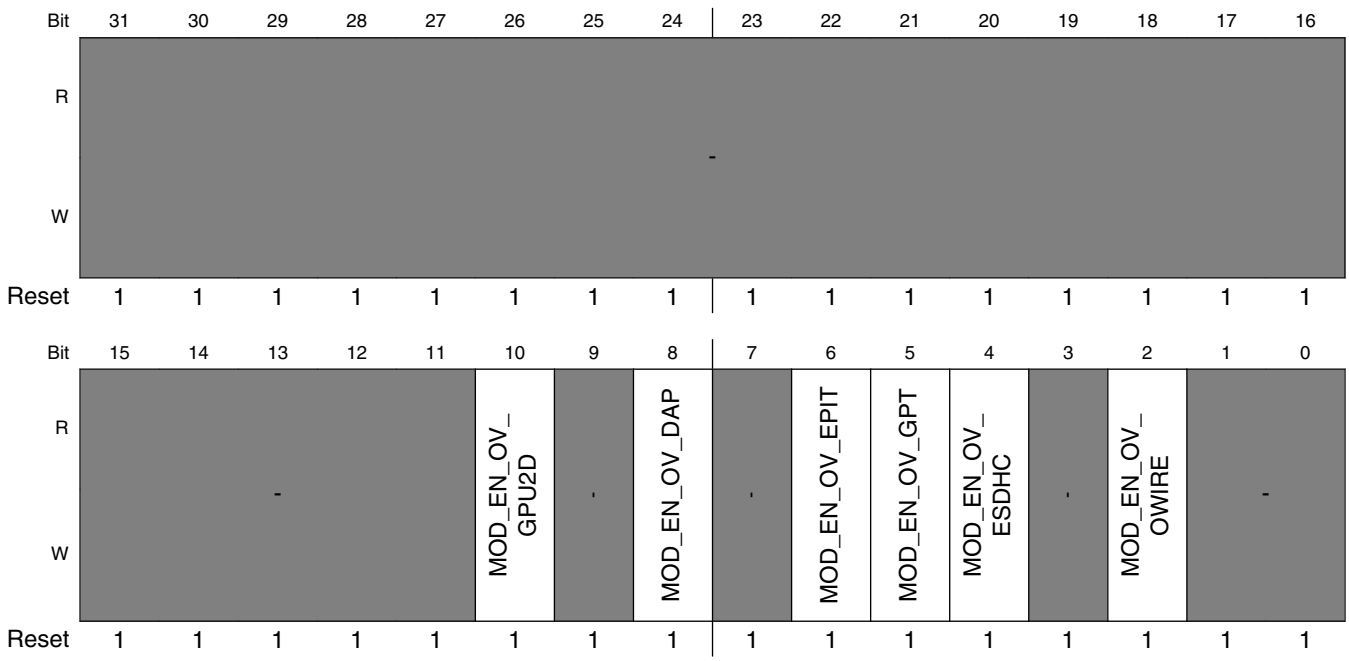
CCM_CCGR7 field descriptions (continued)

Field	Description
	10 Clock is always on 11 Clock is always on except in STOP mode
13–12 CG6	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
11–10 CG5	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
9–8 CG4	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
7–6 CG3	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
5–4 CG2	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
3–2 CG1	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode
1–0 CG0	Defines clock gating for power reduction of clock see Table 5-62. 0 Clock is off during all modes. Hardware handshake is disabled for entering STOP mode. 01 Clock is on in RUN mode, but off in WAIT and STOP modes 10 Clock is always on 11 Clock is always on except in STOP mode

5.4.35 CCM Module Enable Override Register (CCM_CMEOR)

The following figure represents the CCM Module Enable Override Register (CMEOR). The CMEOR register contains bits to override the clock enable signal from the module. This should be used in case it is decided to bypass the clock enable signals from the modules. This bit will be applicable only for the modules for which a clock enable signal is used.

Address: CCM_CMEOR is 53FD_4000h base + 88h offset = 53FD_4088h



CCM_CMEOR field descriptions

Field	Description
31–11 -	Reserved
10 MOD_EN_OV_GPU2D	Overide clock enable signal from gpu2d - clock will not be gated based on gpu2d's signal 'gpu2d_busy' . 0 don't overide module enable signal 1 overide module enable signal
9 -	Reserved
8 MOD_EN_OV_DAP	Overide clock enable signal from dap- clock will not be gated based on dap's signal 'dap_dbgen' . 0 don't overide module enable signal 1 overide module enable signal
7 -	Reserved

Table continues on the next page...

CCM_CMEOR field descriptions (continued)

Field	Description
6 MOD_EN_OV_EPIT	Overide clock enable signal from epit - clock will not be gated based on epit's signal 'ipg_enable_clk' . 0 don't overide module enable signal 1 overide module enable signal
5 MOD_EN_OV_GPT	Overide clock enable signal from gpt - clock will not be gated based on gpt's signal 'ipg_enable_clk' . 0 don't overide module enable signal 1 overide module enable signal
4 MOD_EN_OV_ESDHC	Overide clock enable signal from esdhc - clock will not be gated based on esdhc's signals 'hclk_en', 'ipg_clk_en' and 'ipg_per_clk_en'. 0 don't overide module enable signal 1 overide module enable signal
3 -	Reserved
2 MOD_EN_OV_OWIRE	Overide clock enable signal from owire - clock will not be gated based on owire's signal 'owire_clk_en'. 0 don't overide module enable signal 1 overide module enable signal
1–0 -	Reserved

5.4.36 CCM Control Status Register 2 (CCM_CSR2)

The following figure represents the CCM Control Status Register 2 (CSR2). The CSR2 register contains bits which check the status of various CCM divide and glitchless clock mux changes. When the busy bit for the respective clock divider is high, the divider change is being updated; subsequent updates to the divider should not be performed until the busy bit has cleared.

Address: CCM_CSR2 is 53FD_4000h base + 8Ch offset = 53FD_408Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-		EPDC_ASM_ACTIVE	EPXP_ASM_ACTIVE	ELCDIF_ASM_ACTIVE	SYS_CLK_XTAL_ACTIVE	ELCDIF_PIX_BUSY	EPDC_PIX_BUSY	MSHC_XMSCKI_BUSY	BCH_BUSY	GPMI_BUSY	EPDC_AXI_BUSY	DISPLAY_AXI_BUSY	DDR_CLK_REF_PLL_BUSY	SYS_CLK_REF_PLL_BUSY	SYS_CLK_REF_XTAL_BUSY
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

CCM_CSR2 field descriptions

Field	Description
31–14 -	Reserved
13 EPDC_ASM_ ACTIVE	Indicates Auto Slow Mode is actively slowing down the epdc_axi_clk 1 Auto Slow Mode is Active 0 Auto Slow Mode is disengaged, clock is at full rate
12 EPXP_ASM_ ACTIVE	Indicates Auto Slow Mode is actively slowing down the epxp_axi_clk 1 Auto Slow Mode is Active 0 Auto Slow Mode is disengaged, clock is at full rate
11 ELCDIF_ASM_ ACTIVE	Indicates Auto Slow Mode is actively slowing down the elcdif_axi_clk 1 Auto Slow Mode is Active 0 Auto Slow Mode is disengaged, clock is at full rate
10 SYS_CLK_ XTAL_ACTIVE	Indicates the 24MHz xtal is active clock from the glitchless clock mux between xtal and pll sources. 1 Xtal Active clock 0 PLL/PFD active clock
9 ELCDIF_PIX_ BUSY	Indicates divider change in progress for elcdif_pix_div. 1 Update in progress, Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.
8 EPDC_PIX_ BUSY	Indicates divider change in progress for epdc_pix_div. 1 Update in progress, Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.
7 MSHC_ XMSCKI_BUSY	Indicates divider change in progress for mshc_xmscki_div. 1 Update in progress, Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.
6 BCH_BUSY	Indicates divider change in progress for bch_div. 1 Update in progress, Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.
5 GPMI_BUSY	Indicates divider change in progress for gpmi_div. 1 Update in progress. Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.
4 EPDC_AXI_ BUSY	Indicates divider change in progress for epdc_axi_div. 1 Update in progress. Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.
3 DISPLAY_AXI_ BUSY	Indicates divider change in progress for display_axi_div. 1 Update in progress. Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.

Table continues on the next page...

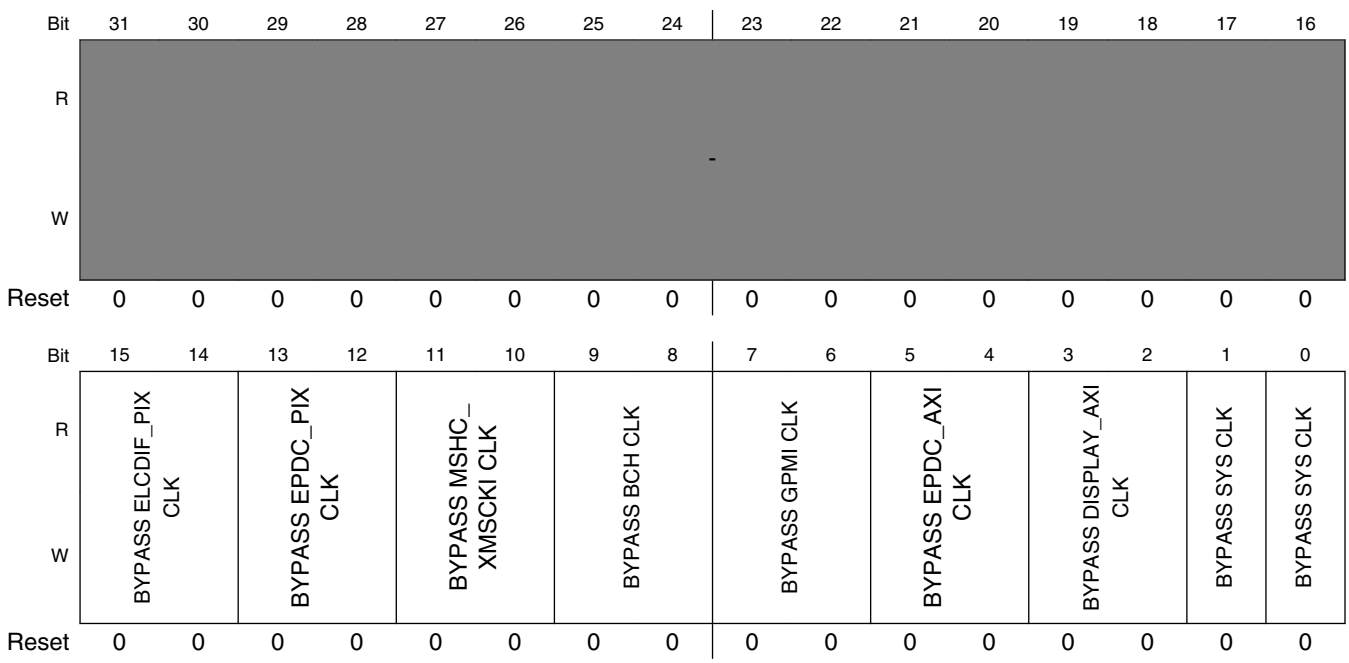
CCM_CSR2 field descriptions (continued)

Field	Description
2 DDR_CLK_ REF_PLL_BUSY	Indicates divider change in progress for ddr_div_pll. 1 Update in progres. Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.
1 SYS_CLK_REF_ PLL_BUSY	Indicates divider change in progress for sys_div_pll. 1 Update in progress. Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.
0 SYS_CLK_REF_ XTAL_BUSY	Indicates divider change in progress for sys_div_xtal. 1 Update in progress. Do not change divider setting at this time. 0 Update completed. Ready to accept new divider value.

5.4.37 CCM Clock Sequence Bypass (CCM_CLKSEQ_BYPASS)

The following figure represents the CCM Clock Sequence Bypass Register (CLKSEQ_BYPASS). The CLKSEQ_BYPASS register contains bits to change between 24MHz xtal and PLL/PFD reference clocks.

Address: CCM_CLKSEQ_BYPASS is 53FD_4000h base + 90h offset = 53FD_4090h



CCM_CLKSEQ_BYPASS field descriptions

Field	Description
31–16 -	Reserved

Table continues on the next page...

CCM_CLKSEQ_BYPASS field descriptions (continued)

Field	Description
15–14 BYPASS ELCDIF_ PIX CLK	Mux select between 24 MHz xtal and PLL/PFD clock sources for elcdif_pix_clk 00 24MHz xtal 01 PFD6 10 PLL1 11 Camp1
13–12 BYPASS EPDC_ PIX CLK	Mux select between 24 MHz xtal and PLL/PFD clock sources for epdc_pix_clk 00 24MHz xtal 01 PFD5 10 PLL1 11 Camp1
11–10 BYPASS MSHC_ XMSCKI CLK	Mux select between 24 MHz xtal and PLL/PFD clock sources for mshc_xmscki 00 24 MHz xtal 01 PFD7 10 PLL1 11 Reserved
9–8 BYPASS BCH CLK	Mux select between 24 MHz xtal and PLL/PFD clock sources for bch_clk 00 24 MHz xtal 01 PFD4 10 PLL1 11 Reserved
7–6 BYPASS GPMI CLK	Mux select between 24 MHz xtal and PLL/PFD clock sources for gpmi_clk 00 24 MHz xtal 01 PFD4 10 PLL1 11 Reserved
5–4 BYPASS EPDC_ AXI CLK	Mux select between 24 MHz xtal and PLL/PFD clock sources for epdc_axi_clk 00 24 MHz xtal 01 PFD3 10 PLL1 11 Reserved
3–2 BYPASS DISPLAY_ AXI CLK	Mux select between 24 MHz xtal and PLL/PFD clock sources for display_axi_clk 00 24 MHz xtal 01 PFD2 10 PLL1 11 Reserved
1 BYPASS SYS CLK	Mux select between PFD1 and PLL clock sources for sys_clk 0 PFD1 1 PLL1

Table continues on the next page...

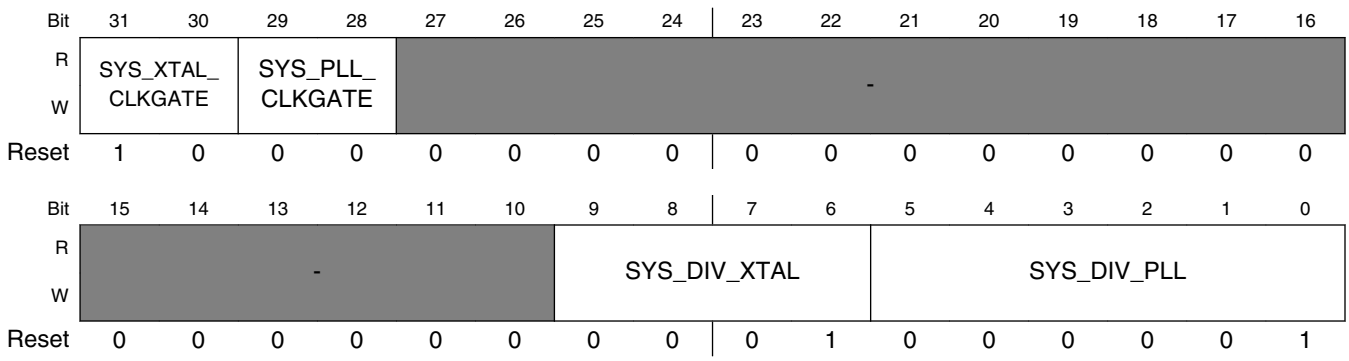
CCM_CLKSEQ_BYPASS field descriptions (continued)

Field	Description
0 BYPASS SYS CLK	Mux select between 24 MHz xtal and PLL/PFD clock sources for sys_clk 0 24 MHz xtal 1 PFD1/PLL (bypass sys clk [1])

5.4.38 CCM System Clock Register (CCM_CLK_SYS)

The following figure represents the CCM System clock register (CLK_SYS). The CLK_SYS register contains bits to program the system clock.

Address: CCM_CLK_SYS is 53FD_4000h base + 94h offset = 53FD_4094h



CCM_CLK_SYS field descriptions

Field	Description
31–30 SYS_XTAL_ CLKGATE	Enable/Disable sys_clk xtal source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–28 SYS_PLL_ CLKGATE	Enable/Disable sys_clk PLL/PFD source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
27–10 -	Reserved
9–6 SYS_DIV_XTAL	Divide field for sys_clk 24MHz xtal clock divider 0000 Gate clock off 0001 Divide by 1 0010 Divide by 2 1111 Divide by (2^4)-1

Table continues on the next page...

CCM_CLK_SYS field descriptions (continued)

Field	Description
5–0 SYS_DIV_PLL	Divide field for sys_clk PLL/PFD clock divider **NOTE:** The sys_pll_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0 000000 Gate clock off 000001 Divide by 1 000010 Divde by 2 111111 Divide by (2^6)-1

5.4.39 CCM DDR Clock Register (CCM_CLK_DDR)

The following figure represents the CCM DDR clock register (CLK_DDR). The CLK_DDR register contains bits to program the system clock.

Address: CCM_CLK_DDR is 53FD_4000h base + 98h offset = 53FD_4098h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DDR_CLKGATE		-													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								DDR_PFD_SEL	DDR_DIV_PLL						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

CCM_CLK_DDR field descriptions

Field	Description
31–30 DDR_CLKGATE	Enable/Disable asynchronous ddr_clk PLL/PFD source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–7 -	Reserved
6 DDR_PFD_SEL	Selects between PLL and PFD reference clocks for the asynchronous ddr clock divider 0 Select PLL 1 Select PFD0

Table continues on the next page...

CCM_CLK_DDR field descriptions (continued)

Field	Description
5–0 DDR_DIV_PLL	Divide field for ddr_clk PLL/PFD clock divider **NOTE:** The ddr_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0. 000000 Gate clock off 000001 Divide by 1 000010 Divde by 2 111111 Divide by (2^6)-1

5.4.40 CCM ELCDIF PIX Clock Serial Divide Register (CCM_ELCDIFPIX)

The following figure represents the CCM ELCDIF Pix Clock Serial Divide Register (ELCDIFPIX). The ELCDIFPIX register contains bits to control the clock generation submodule dividers.

Address: CCM_ELCDIFPIX is 53FD_4000h base + 9Ch offset = 53FD_409Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ELCDIF_PIX_CLKGATE		-													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-		ELCDIF_CLK_PRED[1:0]		ELCDIF_PIX_CLK_PODF[11:0]											
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

CCM_ELCDIFPIX field descriptions

Field	Description
31–30 ELCDIF_PIX_CLKGATE	Enable/Disable asynchronous elcdif_pix_clk source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–14 -	Reserved
13–12 ELCDIF_CLK_PRED[1:0]	Divider for elcdif pix clock pred. **NOTE:** The elcdif_pix_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0.

Table continues on the next page...

CCM_ELCDIFPIX field descriptions (continued)

Field	Description
	00 Clock is gated off 01 divide by 1 10 divide by 2 11 divide by 3
11–0 ELCDIF_PIX_ CLK_ PODF[11:0]	Divider for elcdif clock podf. **NOTE:** The elcdif_pix_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0. 000 Clock is gated off 001 divide by 1 010 divide by 2 011 divide by 3 100 divide by 4 101 divide by 5 110 divide by 6 111 divide by 7 111111111111 divide by (2^12)-1

5.4.41 CCM EPDC PIX Clock Serial Divide Register (CCM_EPDCPIX)

The following figure represents the CCM EPDC Pix Clock Serial Divide Register (EPDCPIX). The EPDCPIX register contains bits to control the clock generation submodule dividers.

NOTE

The ELCDIFPIX and EPDCPIX dividers are serial dividers. If both the pre and podf dividers are written at the same time, the podf divider has higher priority and will take effect before the pre divider.

Address: CCM_EPDCPIX is 53FD_4000h base + A0h offset = 53FD_40A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EPDC_PIX_ CLKGATE		-													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-		EPDC_CLK_ PRED[1:0]		EPDC_PIX_CLK_PODF[11:0]											
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

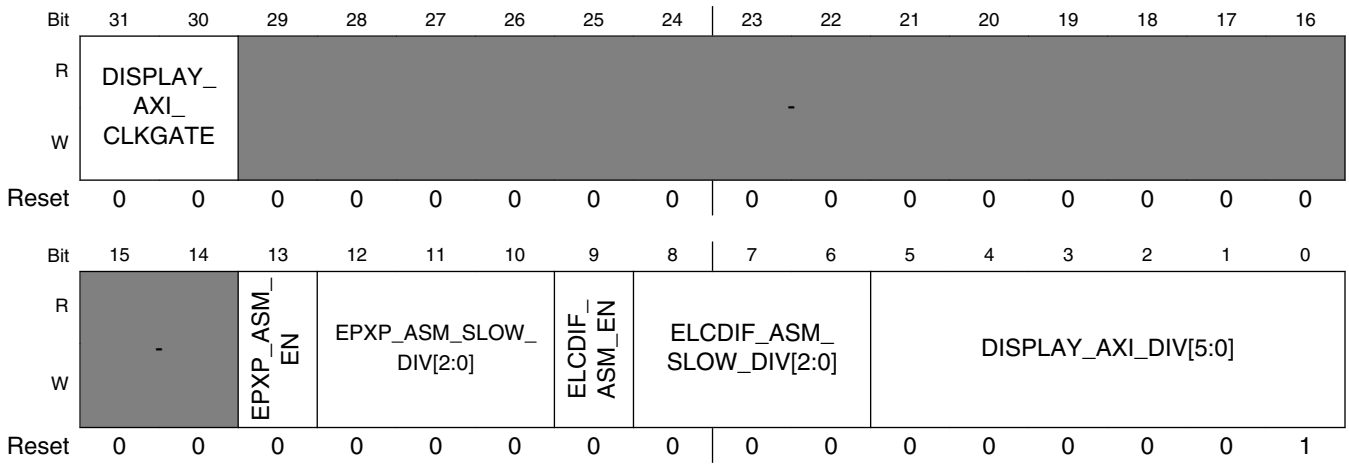
CCM_EPDCPIX field descriptions

Field	Description
31–30 EPDC_PIX_ CLKGATE	Enable/Disable asynchronous epdc_pix_clk source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–14 -	Reserved
13–12 EPDC_CLK_ PRED[1:0]	Divider for epdc_pix clock pred. **NOTE:** The epdc_pix_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0. 00 Clock is gated off 01 divide by 1 10 divide by 2 11 divide by 3
11–0 EPDC_PIX_ CLK_ PODF[11:0]	Divider for epdc_pix clock podf. **NOTE:** The epdc_pix_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0. 000 Clock is gated off 001 divide by 1 010 divide by 2 011 divide by 3 100 divide by 4 101 divide by 5 110 divide by 6 111 divide by 7 111111111111 divide by (2^12)-1

5.4.42 CCM DISPLAY_AXI Clock Divide Register (CCM_DISPLAY_AXI)

The following figure represents the CCM DISPLAY_AXI Clock Divide Register (DISPLAY_AXI). The DISPLAY_AXI register contains bits to control the clock generation submodule dividers.

Address: CCM_DISPLAY_AXI is 53FD_4000h base + A4h offset = 53FD_40A4h



CCM_DISPLAY_AXI field descriptions

Field	Description
31–30 DISPLAY_AXI_ CLKGATE	Enable/Disable asynchronous epdc_axi_clk source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–14 -	Reserved
13 EPXP_ASM_EN	Enable Auto slow Mode for epxp_axi clock 1 Enable Auto Slow Mode 0 Disable Auto Slow Mode
12–10 EPXP_ASM_ SLOW_DIV[2:0]	ePXP auto slow mode divide. The display_axi clock for the epxp_axi branch is divided down when epxp_asm_en and the ePXP is not busy. 000 Clock is not auto slowed 001 Clock is divided by 2 010 Clock is divided by 4 011 Clock is divided by 8 100 Clock is divided by 16 101 Clock is divided by 32 110 - 111 Reserved

Table continues on the next page...

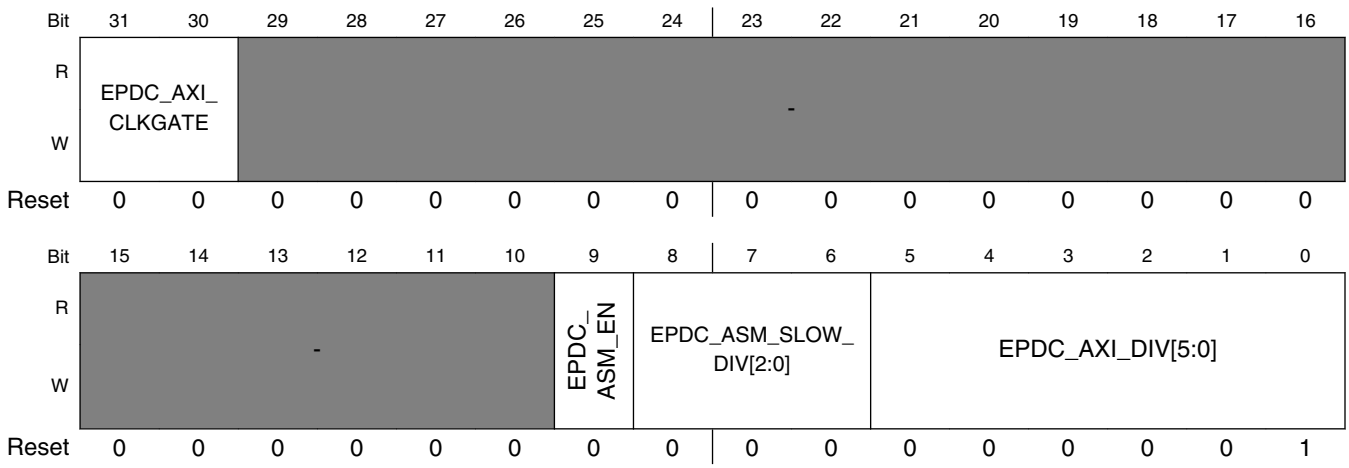
CCM_DISPLAY_AXI field descriptions (continued)

Field	Description
9 ELCDIF_ASM_EN	Enable Auto slow Mode for elcdif_axi clock 1 Enable Auto Slow Mode 0 Disable Auto Slow Mode
8–6 ELCDIF_ASM_SLOW_DIV[2:0]	Elcdif auto slow mode divide. The display_axi clock for the elcdif_axi branch is divided down when elcdif_asm_en and the elcdif is not busy. 000 Clock is not auto slowed 001 Clock is divided by 2 010 Clock is divided by 4 011 Clock is divided by 8 100 Clock is divided by 16 101 Clock is divided by 32 110 - 111 Reserved
5–0 DISPLAY_AXI_DIV[5:0]	Divider for display_axi clock. **NOTE:** The display_axi_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0. 000 Clock is gated off 001 divide by 1 010 divide by 2 011 divide by 3 100 divide by 4 101 divide by 5 110 divide by 6 111 divide by 7 111111 divide by (2^6)-1

5.4.43 CCM EPDC_AXI Clock Divide Register (CCM_EPDC_AXI)

The following figure represents the CCM EPDC_AXI Clock Divide Register (EPDC_AXI). The EPDC_AXI register contains bits to control the clock generation submodule dividers.

Address: CCM_EPDC_AXI is 53FD_4000h base + A8h offset = 53FD_40A8h



CCM_EPDC_AXI field descriptions

Field	Description
31–30 EPDC_AXI_ CLKGATE	Enable/Disable asynchronous epdc_axi_clk source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–10 -	Reserved
9 EPDC_ASM_EN	Enable Auto slow Mode for epdc_axi clock 1 Enable Auto Slow Mode 0 Disable Auto Slow Mode
8–6 EPDC_ASM_ SLOW_DIV[2:0]	Elcdif auto slow mode divide. The epdc_axi clock is divided down when epdc_asm_en and the epdc is not busy. 000 Clock is not auto slowed 001 Clock is divided by 2 010 Clock is divided by 4 011 Clock is divided by 8 100 Clock is divided by 16 101 Clock is divided by 32 110 - 111 Reserved

Table continues on the next page...

CCM_EPDC_AXI field descriptions (continued)

Field	Description
5–0 EPDC_AXI_ DIV[5:0]	Divider for epdc_axi clock. **NOTE:** The epdc_axi_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0. 000 Clock is gated off 001 divide by 1 010 divide by 2 011 divide by 3 100 divide by 4 101 divide by 5 110 divide by 6 111 divide by 7 111111 divide by (2^6)-1

5.4.44 CCM GPMI Clock Divide Register (CCM_GPMI)

The following figure represents the CCM GPMI Clock Divide Register (GPMI). The GPMI register contains bits to control the clock generation submodule dividers.

Address: CCM_GPMI is 53FD_4000h base + ACh offset = 53FD_40ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPMI_ CLKGATE		-													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-										GPMI_DIV[5:0]					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

CCM_GPMI field descriptions

Field	Description
31–30 GPMI_ CLKGATE	Enable/Disable asynchronous gpmi_clk source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–6 -	Reserved
5–0 GPMI_DIV[5:0]	Divider for gpmi clock.

Table continues on the next page...

CCM_GPMI field descriptions (continued)

Field	Description
	NOTE: The gpmi_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0. 000 Clock is gated off 001 divide by 1 010 divide by 2 011 divide by 3 100 divide by 4 101 divide by 5 110 divide by 6 111 divide by 7 111111 divide by (2^6)-1

5.4.45 CCM BCH Clock Divide Register (CCM_BCH)

The following figure represents the CCM BCH Clock Divide Register (BCH). The BCH register contains bits to control the clock generation submodule dividers.

Address: CCM_BCH is 53FD_4000h base + B0h offset = 53FD_40B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BCH_ CLKGATE		-													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-										BCH_DIV[5:0]					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

CCM_BCH field descriptions

Field	Description
31–30 BCH_CLKGATE	Enable/Disable asynchronous bch_clk source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–6 -	Reserved
5–0 BCH_DIV[5:0]	Divider for bch clock. **NOTE:** The bch_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0.

Table continues on the next page...

CCM_BCH field descriptions (continued)

Field	Description	
	000	Clock is gated off
	001	divide by 1
	010	divide by 2
	011	divide by 3
	100	divide by 4
	101	divide by 5
	110	divide by 6
	111	divide by 7
	111111	divide by (2^6)-1

5.4.46 CCM MSHC_XMSCKI Clock Divide Register (CCM_MSHC_XMSCKI)

The following figure represents the CCM MSHC_XMSCKI Clock Divide Register (MSHC_XMSCKI). The MSHC_XMSCKI register contains bits to control the clock generation submodule dividers.

NOTE

The CLK_SYS, CLK_DDR, ELCDIFPIX, EPDCPIX, DISPLAY_AXI, EPDC_AXI, GPMI, BCH and MSHC_XMSCKI dividers load new divide values on the fly without loss of clock cycles. The dividers should not be gated off when the divide value is updated; the new divide value will only take effect if the divider is active.

Address: CCM_MSHC_XMSCKI is 53FD_4000h base + B4h offset = 53FD_40B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MSHC_ XMSCKI_ CLKGATE		-													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-										MSHC_XMSCKI_DIV[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

CCM_MSHC_XMSCKI field descriptions

Field	Description
31–30 MSHC_ XMSCKI_ CLKGATE	Enable/Disable asynchronous mshc_xmscki_clk source clock divider 00 Clock is gated off 01 Clock is on in run mode, but off in wait and stop modes 10 Clock is always on 11 Clock is on during all modes, except stop mode
29–6 -	Reserved
5–0 MSHC_ XMSCKI_ DIV[5:0]	Divider for mshc_xmscki clock. **NOTE:** The mshc_xmscki_clkgate will gate the clock off before the divider and should be used for maximum power savings instead of dividing by 0. 000 Clock is gated off 001 divide by 1 010 divide by 2 011 divide by 3 100 divide by 4 101 divide by 5 110 divide by 6 111 divide by 7 111111 divide by (2^6)-1

5.4.47 Clock Gating bits (CG(i))

These bits are used to turn on/off the clock to each module independently. Table 5-54 details the possible clock activity conditions for each module.

Table 5-54. Clock Gating Bits Description

CGR value	Clock Activity Description
00	Clock is off during all modes.
01	Clock is on in run mode, but off in wait and stop modes
10	Clock is always on
11	Clock is on during all modes, except stop mode.

NOTE

Module should be stopped before the CGR value is set to 00 because clocks to the module will be stopped immediately.

Table 5-55 through Table 5-62 show the register mappings for the gated clocks connected to each target module.

The CCM Root Clock in each table refers to the parent clock output shown in Figure 5-1.

Table 5-55. CCM_CCGR0 Gated Clock Mapping to Target Module

CCGR0 Register Bits	MCG Bit Index	CCM Root Clock	Module
[1:0]	0	ipg_clk_root	ARM Cortex A8
[3:2]	1	Reserved	Reserved
[5:4]	2	ahb_clk_root	ARM Cortex A8
[7:6]	3	ahb_clk_root	TZIC
		ahb_clk_root	NIC301
[9:8]	4	ahb_clk_root	CoreSight Debug Access Port (DAP)
		ahb_clk_root	DAP
		debug_apb_clk_root	DAP
[11:10]	5	debug_apb_clk_root	Trace Port Interface Unit (TPIU)
		debug_apb_clk_root	CTI2
[13:12]	6	debug_apb_clk_root	CTI2
[15:14]	7	debug_apb_clk_root	CTI3
[17:16]	8	ahb_clk_root	AHBMUX1 Dependencies:FEC,SDMA,MSHC,GPU2D,NIC301(Slave5)
[19:18]	9	Reserved	Reserved
[21:20]	10	ahb_clk_root	ROMCP
		ipg_clk_root	ROMCP
		ipg_clk_root	NIC301
[23:22]	11	ahb_clk_root	ROM
[25:24]	12	ahb_clk_root	AIPSTZ1
[27:26]	13	ahb_clk_root	AIPSTZ2
[29:28]	14	ipg_clk_root	AHBMAX
		ahb_clk_root	AHBMAX
		ahb_clk_root	NIC301
[31:30]	15	Reserved	Reserved

Table 5-56. CCM_CCGR1 Gated Clock Mapping to Target Module

CCGR1 Register Bits	MCG Bit Index	CCM Root Clock	Module
[1:0]	0	Reserved	Reserved
[3:2]	1	ahb_clk_root	TMAX2 Dependencies: eSDHCv2_1, eSDHCv2_2
[5:4]	2	Reserved	Reserved

Table continues on the next page...

Table 5-56. CCM_CCGR1 Gated Clock Mapping to Target Module (continued)

CCGR1 Register Bits	MCG Bit Index	CCM Root Clock	Module
[7:6]	3	ipg_clk_root	UART1
[9:8]	4	uart_clk_root	UART1
[11:10]	5	ipg_clk_root	UART2
[13:12]	6	uart_clk_root	UART2
[15:14]	7	ipg_clk_root	UART3
[17:16]	8	uart_clk_root	UART3
[19:18]	9	perclk_root	I2C1
[21:20]	10	perclk_root	I2C2
[23:22]	11	perclk_root	I2C3
[25:24]	12	Reserved	Reserved
[27:26]	13	Reserved	Reserved
[29:28]	14	Reserved	Reserved
[31:30]	15	Reserved	Reserved

Table 5-57. CCM_CCGR2 Gated Clock Mapping to Target Module

CCGR2 Register Bits	MCG Bit Index	CCM Root Clock	Module
[1:0]	0	Reserved	Reserved
[3:2]	1	ipg_clk_root	EPIT1
		ipg_clk_root	CTI3
[5:4]	2	perclk_root	EPIT1
[7:6]	3	Reserved	Reserved
[9:8]	4	Reserved	Reserved
[11:10]	5	ipg_clk_root	PWM1
[13:12]	6	perclk_root	PWM1
[15:14]	7	ipg_clk_root	PWM2
[17:16]	8	perclk_root	PWM2
[19:18]	9	ipg_clk_root	GPT
[21:20]	10	perclk_root	GPT
[23:22]	11	perclk_root	OWIRE
[25:24]	12	ipg_clk_root	FEC
		ahb_clk_root	FEC
		ahb_clk_root	NIC301
[27:26]	13	ahb_clk_root	USBOH1
		ahb_clk_root	NIC301
[29:28]	14	Reserved	Reserved

Table continues on the next page...

Table 5-57. CCM_CCGR2 Gated Clock Mapping to Target Module (continued)

CCGR2 Register Bits	MCG Bit Index	CCM Root Clock	Module
[31:30]	15	Reserved	Reserved

Table 5-58. CCM_CCGR3 Gated Clock Mapping to Target Module

CCGR3 Register Bits	MCG Bit Index	CCM Root Clock	Module
[1:0]	0	ipg_clk_root	eSDHCv2_1
		ahb_clk_root	eSDHCv2_1
[3:2]	1	esdhc1_clk_root	eSDHCv2_1
[5:4]	2	ipg_clk_root	eSDHCv2_2
		ahb_clk_root	eSDHCv2_2
[7:6]	3	esdhc2_clk_root	eSDHCv2_2
[9:8]	4	ipg_clk_root	eSDHCv3_3
		ahb_clk_root	eSDHCv3_3
[11:10]	5	esdhc3_clk_root	eSDHCv3_3
[13:12]	6	ipg_clk_root	eSDHCv2_4
		ahb_clk_root	eSDHCv2_4
[15:14]	7	esdhc4_clk_root	eSDHCv2_4
[17:16]	8	ipg_clk_root	SSI1
[19:18]	9	ssi1_clk_root	SSI1
[21:20]	10	ipg_clk_root	SSI2
[23:22]	11	ssi1_clk_root	SSI2
[25:24]	12	Reserved	Reserved
[27:26]	13	Reserved	Reserved
[29:28]	14	ssi1e_clk_root	OWIRE PAD
[31:30]	15	ssi2e_clk_root	EPIT0 PAD

Table 5-59. CCM_CCGR4 Gated Clock Mapping to Target Module

CCGR4 Register Bits	MCG Bit Index	CCM Root Clock	Module
[1:0]	0	Reserved	Reserved
[3:2]	1	Reserved	Reserved
[5:4]	2	Reserved	Reserved
[7:6]	3	Reserved	Reserved
[9:8]	4	Reserved	Reserved
[11:10]	5	usb_phy_clk_root	USBPHY1
[13:12]	6	usb_phy_clk_root	USBPHY2
[15:14]	7	Reserved	Reserved

Table continues on the next page...

Table 5-59. CCM_CCGR4 Gated Clock Mapping to Target Module (continued)

CCGR4 Register Bits	MCG Bit Index	CCM Root Clock	Module
[17:16]	8	Reserved	Reserved
[19:18]	9	ipg_clk_root	ECSPI1
[21:20]	10	ecspi_clk_root	ECSPI1
[23:22]	11	ipg_clk_root	ECSPI2
[25:24]	12	ecspi_clk_root	ECSPI2
[27:26]	13	ipg_clk_root	CSPI
[29:28]	14	ipg_clk_root	SRTC
[31:30]	15	ahb_clk_root	SDMA
		ipg_clk_root	SDMA
		ipg_clk_root	CTI2
		ahb_clk_root	NIC301

Table 5-60. CCM_CCGR5 Gated Clock Mapping to Target Module

CCGR5 Register Bits	MCG Bit Index	CCM Root Clock	Module
[1:0]	0	ipg_clk_root	SPBA
		ipg_clk_root	CTI2
[3:2]	1	Reserved	Reserved
[5:4]	2	Reserved	Reserved
[7:6]	3	Reserved	Reserved
[9:8]	4	Reserved	Reserved
[11:10]	5	Reserved	Reserved
[13:12]	6	ahb_clk_root	IPMUX1 Dependency: AIPSTZ1
		ipg_clk_root	IPMUX1 Dependencies: CCM, EPIT1, GPC, GPIO(1-6), GPT, I2C3 IOMUXC, KPP, MSHC, PWM1, PWM2, RNGB, SPBA, SRC, SRTC, UART1, UART2, UART4, USBOH1, WDOG1
[15:14]	7	Reserved	Reserved
[17:16]	8	weim_clk_root	EIM
		weim_clk_root	NIC301
[19:18]	9	ipg_clk_root	EIM
[21:20]	10	Reserved	Reserved
[23:22]	11	Reserved	Reserved
[25:24]	12	ipg_clk_root	GPC

Table continues on the next page...

Table 5-60. CCM_CCGR5 Gated Clock Mapping to Target Module (continued)

CCGR5 Register Bits	MCG Bit Index	CCM Root Clock	Module
[27:26]	13	Reserved	Reserved
[29:28]	14	Reserved	Reserved
[31:30]	15	Reserved	Reserved

Table 5-61. CCM_CCGR6 Gated Clock Mapping to Target Module

CCGR6 Register Bits	MCG Bit Index	CCM Root Clock	Module
[1:0]	0	ahb_clk_root	IPMUX2 Dependency: AIPSTZ2
		ipg_clk_root	IPMUX2 Dependencies: ARM CORTEX A8, AHBMAX, AUDMUX, DPLLIP(1-3), CSPI, ECSPI2, EIM, FEC, I2C(1-2), OWIRE, ROMCP, SDMA, SSI1, UART5
[3:2]	1	sys_clk_root	OCRAM (CTRL)
		sys_clk_root	OCRAM (RAM)
		sys_clk_root	NIC301
[5:4]	2	Reserved	Reserved
[7:6]	3	Reserved	Reserved
[9:8]	4	mshc_xmscki_clk_root	MSHC
[11:10]	5	epdc_pix_clk_root	EPDC
[13:12]	6	elcdif_pix_clk_root	LCDIF
[15:14]	7	gpu2d_clk_root	GPU2D
		wrck_clk_root	GPU2D
		ahb_clk_root	CSYNC-GPU2D
		gpu2d_clk_root	CSYNC-GPU2D
		gpu2d_clk_root	NIC301
[17:16]	8	ahb_clk_root	EPDC
		epdc_axi_clk_root	EPDC
		epdc_axi_clk_root	NIC301
[19:18]	9	ahb_clk_root	PXP
		display_axi_clk_root	PXP
		display_axi_clk_root	NIC301
[21:20]	10	ahb_clk_root	LCDIF
		display_axi_clk_root	LCDIF
		display_axi_clk_root	NIC301
[23:22]	11	Reserved	Reserved

Table continues on the next page...

Table 5-61. CCM_CCGR6 Gated Clock Mapping to Target Module (continued)

CCGR6 Register Bits	MCG Bit Index	CCM Root Clock	Module
[25:24]	12		
[27:26]	13		
[29:28]	14	ahb_clk_root	CCM Analog
[31:30]	15	ahb_clk_root	DRAM MC

Table 5-62. CCM_CCGR7 Gated Clock Mapping to Target Module

CCGR7 Register Bits	MCG Bit Index	CCM Root Clock	Module
[1:0]	0	bch_clk_root	BCH
[3:2]	1	ipg_clk_root	RNGB
[5:4]	2	ahb_clk_root	AHB2AXI Dependency: MSHC
[7:6]	3	Reserved	Reserved
[9:8]	4	ipg_clk_root	UART4
[11:10]	5	uart_clk_root	UART4
[13:12]	6	ipg_clk_root	UART5
[15:14]	7	uart_clk_root	UART5
[17:16]	8	ahb_clk_root	GPMI
[19:18]	9	gpmi_clk_root	GPMI
[21:20]	10	ahb_clk_root	APBH-BRIDGE-DMA
[23:22]	11	ahb_clk_root	DCP
		ahb_clk_root	NIC301
[25:24]	12	ahb_clk_root	BCH
		ahb_clk_root	NIC301
[27:26]	13	ahb_clk_root	OCOTP
[29:28]	14	ahb_clk_root	MSHC
		ipg_clk_root	MSHC
[31:30]	15	ahb_clk_root	DIGCTRL

5.5 CCM Analog Memory Map/Register Definition

This section defines the control registers for PLL and PFD clocks.

CCM_ANALOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_8010	Fractional Clock Control Register 0 (CCM_ANALOG_FRAC0)	32	R/W	9292_9292h	5.5.1/406
4101_8014	Fractional Clock Control Register 0 (CCM_ANALOG_FRAC0_SET)	32	R/W	9292_9292h	5.5.1/406
4101_8018	Fractional Clock Control Register 0 (CCM_ANALOG_FRAC0_CLR)	32	R/W	9292_9292h	5.5.1/406
4101_801C	Fractional Clock Control Register 0 (CCM_ANALOG_FRAC0_TOG)	32	R/W	9292_9292h	5.5.1/406
4101_8020	Fractional Clock Control Register 1 (CCM_ANALOG_FRAC1)	32	R/W	9292_9292h	5.5.2/408
4101_8024	Fractional Clock Control Register 1 (CCM_ANALOG_FRAC1_SET)	32	R/W	9292_9292h	5.5.2/408
4101_8028	Fractional Clock Control Register 1 (CCM_ANALOG_FRAC1_CLR)	32	R/W	9292_9292h	5.5.2/408
4101_802C	Fractional Clock Control Register 1 (CCM_ANALOG_FRAC1_TOG)	32	R/W	9292_9292h	5.5.2/408
4101_8060	Miscellaneous Register Description (CCM_ANALOG_MISC)	32	R/W	0000_0080h	5.5.3/410
4101_8064	Miscellaneous Register Description (CCM_ANALOG_MISC_SET)	32	R/W	0000_0080h	5.5.3/410
4101_8068	Miscellaneous Register Description (CCM_ANALOG_MISC_CLR)	32	R/W	0000_0080h	5.5.3/410
4101_806C	Miscellaneous Register Description (CCM_ANALOG_MISC_TOG)	32	R/W	0000_0080h	5.5.3/410
4101_8070	PLL Control Register (CCM_ANALOG_PLLCTRL)	32	R/W	00FF_0000h	5.5.4/411
4101_8074	PLL Control Register (CCM_ANALOG_PLLCTRL_SET)	32	R/W	00FF_0000h	5.5.4/411
4101_8078	PLL Control Register (CCM_ANALOG_PLLCTRL_CLR)	32	R/W	00FF_0000h	5.5.4/411
4101_807C	PLL Control Register (CCM_ANALOG_PLLCTRL_TOG)	32	R/W	00FF_0000h	5.5.4/411

5.5.1 Fractional Clock Control Register 0 (CCM_ANALOG_FRAC0*n*)

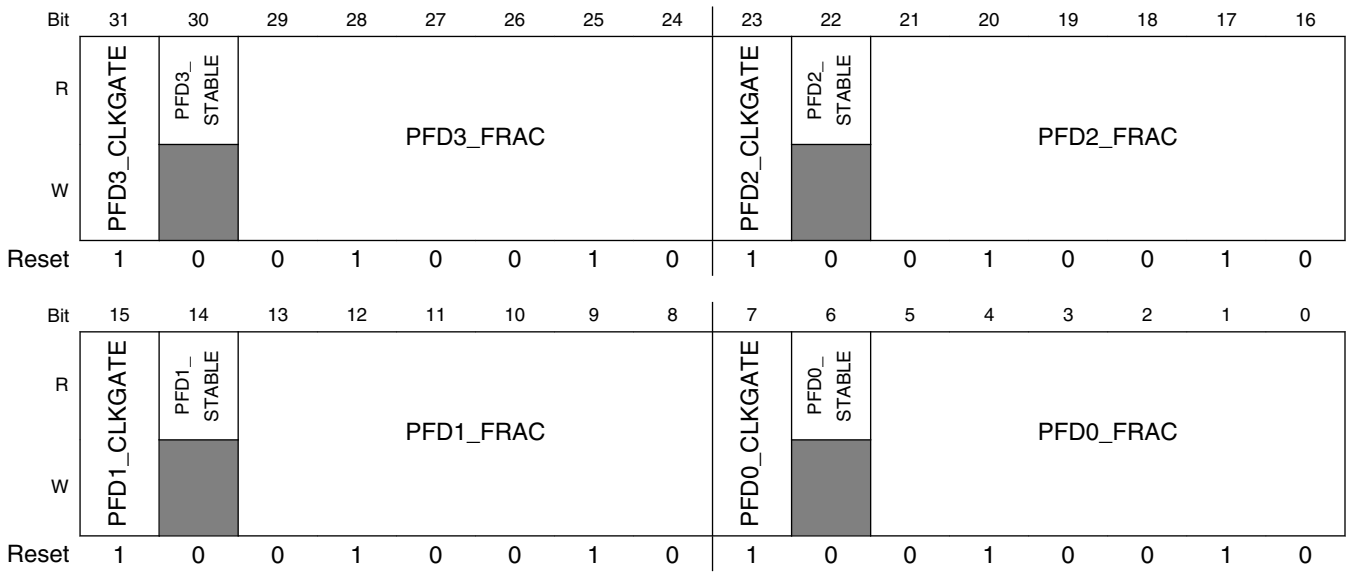
The FRAC0 control register provides control for PFD clock generation. This register controls the 4-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

Addresses: CCM_ANALOG_FRAC0 is 4101_8000h base + 10h offset = 4101_8010h

CCM_ANALOG_FRAC0_SET is 4101_8000h base + 14h offset = 4101_8014h

CCM_ANALOG_FRAC0_CLR is 4101_8000h base + 18h offset = 4101_8018h

CCM_ANALOG_FRAC0_TOG is 4101_8000h base + 1Ch offset = 4101_801Ch



CCM_ANALOG_FRAC0*n* field descriptions

Field	Description
31 PFD3_ CLKGATE	IO Clock Gate. 0 **ref_pfd3 fractional divider clock is enabled** — 1 **The 3rd fractional divider clock (reference ref_pfd3) is off (power saving)** —
30 PFD3_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. **NOTE:** The value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 PFD3_FRAC	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/ PFD3_FRAC) where PFD3_FRAC = 18-35.
23 PFD2_ CLKGATE	IO Clock Gate. 0 **ref_pfd2 fractional divider clock is enabled** — 1 **The 2nd fractional divider clock (reference ref_pfd2) is off (power saving)** —

Table continues on the next page...

CCM_ANALOG_FRAC0*n* field descriptions (continued)

Field	Description
22 PFD2_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. **NOTE:** The value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 PFD2_FRAC	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/PFD2_FRAC) where PFD2_FRAC = 18-35.
15 PFD1_ CLKGATE	IO Clock Gate. 0 **ref_pfd1 fractional divider clock is enabled** — 1 **The 1st fractional divider clock (reference ref_pfd1) is off (power saving)** —
14 PFD1_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. **NOTE:** The value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD1_FRAC	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/PFD1_FRAC) where PFD1_FRAC = 18-35.
7 PFD0_ CLKGATE	IO Clock Gate. 0 **ref_pfd0 fractional divider clock is enabled** — 1 **The 0th fractional divider clock (reference ref_pfd0) is off (power saving)** —
6 PFD0_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. **NOTE:** The value will not invert when the fractional divider is taken out of or placed into clock-gated state.
5–0 PFD0_FRAC	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/PFD0_FRAC) where PFD0_FRAC = 18-35.

5.5.2 Fractional Clock Control Register 1 (CCM_ANALOG_FRAC1*n*)

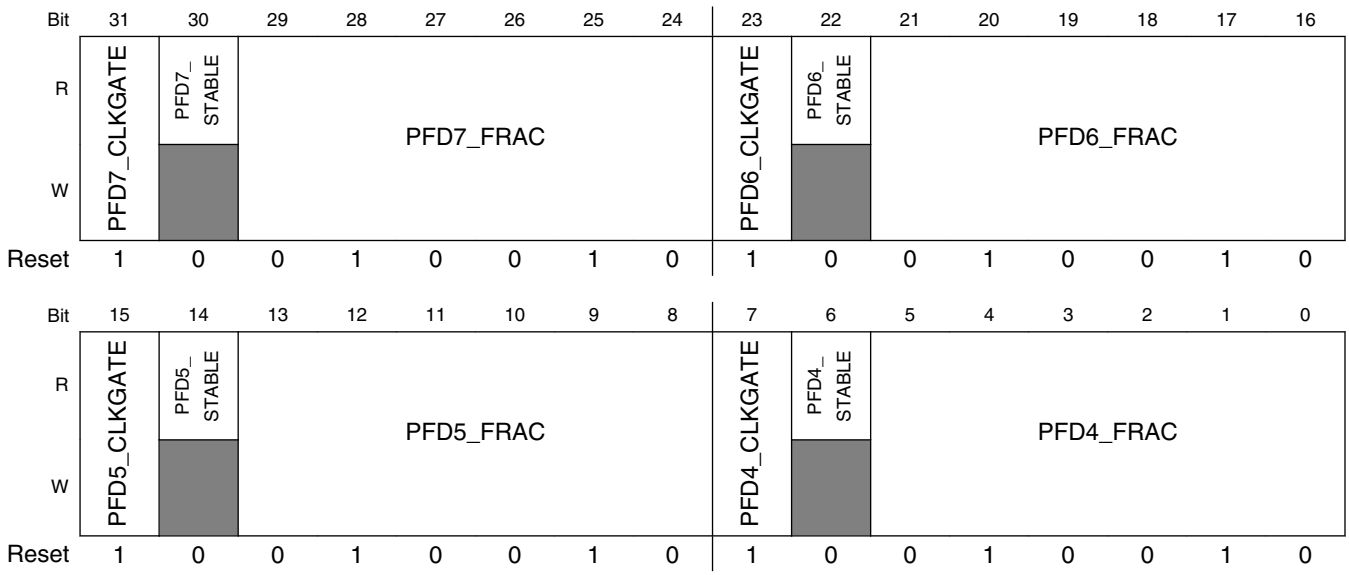
The FRAC1 control register provides control for PFD clock generation. This register controls the 4-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

Addresses: CCM_ANALOG_FRAC1 is 4101_8000h base + 20h offset = 4101_8020h

CCM_ANALOG_FRAC1_SET is 4101_8000h base + 24h offset = 4101_8024h

CCM_ANALOG_FRAC1_CLR is 4101_8000h base + 28h offset = 4101_8028h

CCM_ANALOG_FRAC1_TOG is 4101_8000h base + 2Ch offset = 4101_802Ch



CCM_ANALOG_FRAC1*n* field descriptions

Field	Description
31 PFD7_ CLKGATE	IO Clock Gate. 0 **ref_pfd7 fractional divider clock is enabled** — 1 **The 7th fractional divider clock (reference ref_pfd7) is off (power saving)** —
30 PFD7_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. **NOTE:** The value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 PFD7_FRAC	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/ PFD7_FRAC) where PFD7_FRAC = 18-35.
23 PFD6_ CLKGATE	IO Clock Gate. 0 **ref_pfd6 fractional divider clock is enabled** — 1 **The 6th fractional divider clock (reference ref_pfd6) is off (power saving)** —

Table continues on the next page...

CCM_ANALOG_FRAC1*n* field descriptions (continued)

Field	Description
22 PFD6_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. **NOTE:** The value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 PFD6_FRAC	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/PFD6_FRAC) where PFD6_FRAC = 18-35.
15 PFD5_ CLKGATE	IO Clock Gate. 0 **ref_pfd5 fractional divider clock is enabled** — 1 **The 5th fractional divider clock (reference ref_pfd5) is off (power saving)** —
14 PFD5_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. **NOTE:** The value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD5_FRAC	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/PFD5_FRAC) where PFD5_FRAC = 18-35.
7 PFD4_ CLKGATE	IO Clock Gate. 0 **ref_pfd4 fractional divider clock is enabled** — 1 **The 4th fractional divider clock (reference ref_pfd4) is off (power saving)** —
6 PFD4_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. **NOTE:** The value will not invert when the fractional divider is taken out of or placed into clock-gated state.
5–0 PFD4_FRAC	This field controls the IO clocks fractional divider. The resulting frequency shall be 480 * (18/PFD4_FRAC) where PFD4_FRAC = 18-35.

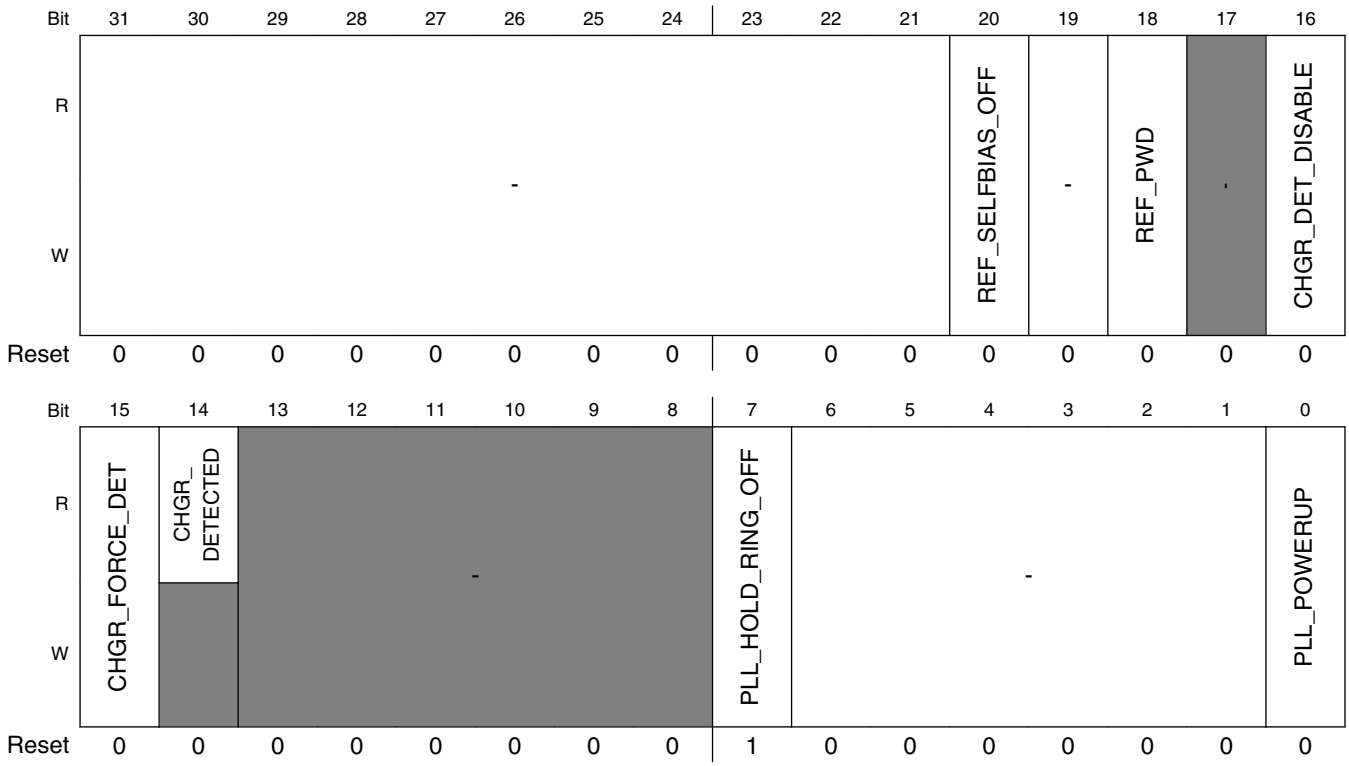
5.5.3 Miscellaneous Register Description (CCM_ANALOG_MISC*n*)

Addresses: CCM_ANALOG_MISC is 4101_8000h base + 60h offset = 4101_8060h

CCM_ANALOG_MISC_SET is 4101_8000h base + 64h offset = 4101_8064h

CCM_ANALOG_MISC_CLR is 4101_8000h base + 68h offset = 4101_8068h

CCM_ANALOG_MISC_TOG is 4101_8000h base + 6Ch offset = 4101_806Ch



CCM_ANALOG_MISC*n* field descriptions

Field	Description
31–21 -	Reserved.
20 REF_ SELFBIAS_OFF	This bit controls the source of the reference bias currents. The bit needs to wait at least 10 us after bandgap is powered up before setting and needs to be set high by software or the tester very early in the flow. The bit needs to be cleared before the bandgap is powered down.
19 -	Reserved.
18 REF_PWD	Powers down the bandgap reference.
17 -	Reserved.
16 CHGR_DET_ DISABLE	This bit disables the charget detect function. The CHGR_FORCE_DET bit should be set if external open drain output needs to be kept asserted.

Table continues on the next page...

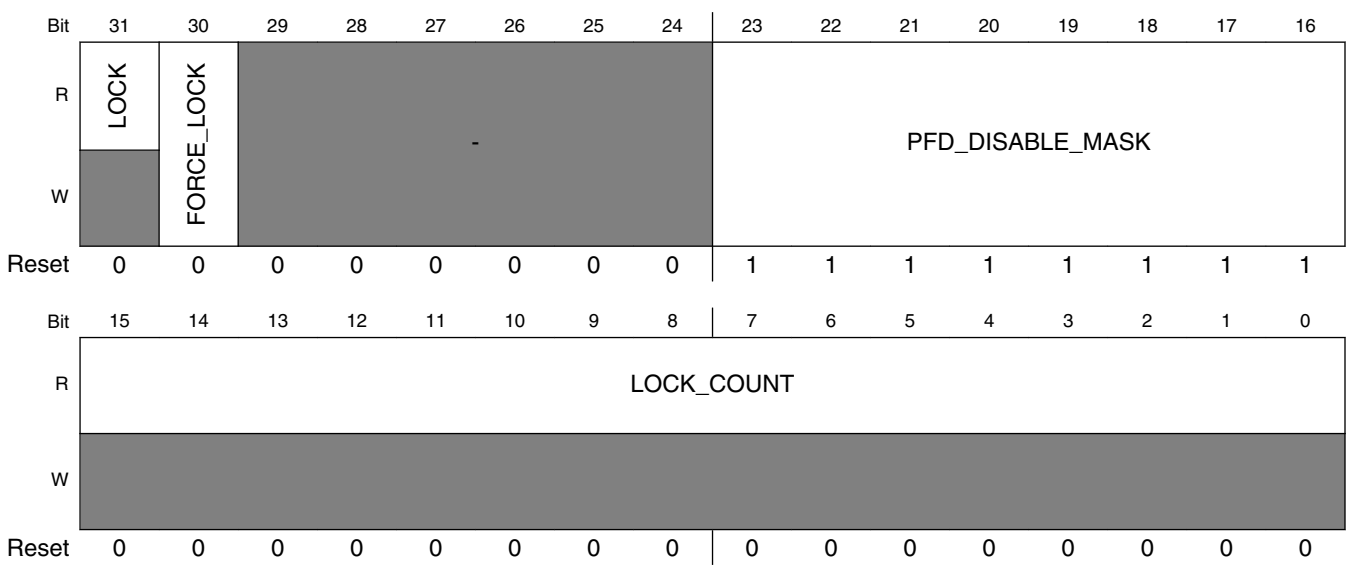
CCM_ANALOG_MISC*n* field descriptions (continued)

Field	Description
15 CHGR_FORCE_DET	This bit asserts the open drain CHGR_DET_B pin regardless of the detector output (enabled or disabled). **NOTE:** This function will not work unless the VBUS power for the OTG USB is present.
14 CHGR_DETECTED	Status bit reflecting the state CHGR_DET_B pin. 1 = asserted.
13–8 -	Reserved.
7 PLL_HOLD_RING_OFF	This hold ring off bit is essential when starting the APLL. This bit should initially be set before powering up the APLL and then cleared more than 10 us later. Additional 10 us wait should occur after clearing the bit before using the output clock either in the PFD's or elsewhere in the system.
6–1 -	Reserved.
0 PLL_POWERUP	PLL Power On. Allow 10 us after turning the PLL0 on before using the PLL0 as a clock source. This is the time the PLL0 takes to lock to 480 MHz. **NOTE:** The POWER bit must be set to on to ungate the reference xtal to all of the PLLs. 0 PLL Off. 1 PLL On.

5.5.4 PLL Control Register (CCM_ANALOG_PLLCTRL*n*)

PLL0 Lock Control Register

Addresses: CCM_ANALOG_PLLCTRL is 4101_8000h base + 70h offset = 4101_8070h
CCM_ANALOG_PLLCTRL_SET is 4101_8000h base + 74h offset = 4101_8074h
CCM_ANALOG_PLLCTRL_CLR is 4101_8000h base + 78h offset = 4101_8078h
CCM_ANALOG_PLLCTRL_TOG is 4101_8000h base + 7Ch offset = 4101_807Ch



CCM_ANALOG_PLLCTRL*n* field descriptions

Field	Description
31 LOCK	PLL0 Lock bit. 0 PLL0 Unlocked. 1 PLL0 Locked.
30 FORCE_LOCK	Force the PLL0 Lock sequence to start. 1= Enable Force Lock. This bit is not self clearing.
29–24 -	Reserved.
23–16 PFD_DISABLE_MASK	This mask determines which PFD's are disabled when the disable_pfds_n is asserted. A 1 indicates the PFD is disabled. The PLL will only be disabled when this mask is set to 0xFF.
15–0 LOCK_COUNT	Status of the PLL0 lock count. The PLL0 lock bit will assert when the count reaches 0x4B0. The lock count is driven off of xtal, so it should be asserted after 50us.

Chapter 6
System Boot

6.1 Introduction

The i.MX50 boot process begins at Power On Reset (POR) where the hardware reset logic forces the ARM core to begin execution starting from the on-chip boot ROM. The boot ROM logic reads hardware inputs such as the boot pins on the IC, OCOTP bits and/or GPIO settings to determine the boot flow behavior of the i.MX50.

The main features of the ROM include:

- Support for booting from various boot devices
- USB Downloader support Device Configuration Data (DCD)
- Digital signature based High Assurance Boot (HAB)

The i.MX50 boot ROM supports the following boot devices:

- Parallel NOR Flash
- NAND Flash
- ONFI 2.x BA-NAND
- Toggle-Mode High speed NAND
- OneNAND Flash
- SD/eSD/MMC/eMMC
- Serial ROM devices, including SPI NOR/EEPROM, I^2C EEPROM

The boot ROM is configured using a combination of external boot pins and OCOTP bits. For development purposes a few key OCOTP bits may be overridden using GPIO pin inputs. The combination of the boot pin and OCOTP bit (or GPIO pin) settings dictate the boot device from which the i.MX50 attempts to boot and the flow within the boot ROM.

The boot ROM logic also allows the downloading of programs to be run on the i.MX50. An example is a provisioning program that can make further use of the serial connection to provision a boot device with a new image. Typically the provisioning program is

downloaded to internal RAM which facilitates programming boot devices such as a NAND Flash. The ROM Serial Downloader uses a high speed USB in non-stream mode connection.

The Device Configuration Data (DCD) feature allows the ROM to obtain IC configuration data from an external Program Image residing on the boot device. As an example, DCD can be used to program the DRAM MC for optimal settings improving the boot performance. DCD is restricted to memory areas and peripheral addresses that are considered essential for boot purposes.

A key feature of the i.MX50 ROM is the ability to perform a secure or High Assurance Boot (HAB). This is supported by the HAB security library which is a subcomponent of the i.MX50 ROM. The HAB uses a combination of hardware and software together with a Public Key Infrastructure (PKI) protocol to protect the system from executing unauthorized programs. Before the HAB allows a user's image to execute, the image must be signed. The signing process is done during the image build process by the private key holder and the signatures are then included as part of the final Program Image. If configured to do so, the i.MX50 verifies the signatures using the public keys included in the Program Image. A secure boot with HAB can be performed on all boot devices supported on i.MX50 in addition to the Serial Downloader. The HAB library in the i.MX50 boot ROM also provides API functions, allowing additional boot chain components (bootloaders) to extend the secure boot chain. The out-of-fab setting for HAB is the open (engineering) configuration mode in which the ROM/HAB performs image authentication but all authentication errors are ignored and the image is still allowed to execute.

The remainder of this chapter provides the details on how to configure and use the boot features of the i.MX50.

6.2 Boot Modes

6.2.1 Boot Mode Pin Settings

The i.MX50 has four boot modes (one is reserved for FSL internal use) that are selected by the two boot mode pins on the IC package (BOOT_MODE[1:0]). The settings of these two pins are sampled out of reset and stored in the BOOT_MODE[1:0] field of the SRC Boot Mode Register (SBMR) in the SRC (System Reset Controller) module. The boot modes are: Internal, Reserved , Internal Boot - Fuses Only, and USB Downloader.

Table 6-1. Boot Mode Pin Settings

BOOT_MODE[1:0]	Boot Type
00	Internal Boot
01	Reserved
10	Internal Boot-Fuses Only
11	USB Downloader

6.2.2 High Level Boot Sequence

Figure 6-1 shows the High Level processor ROM code flow followed in the i.MX50.

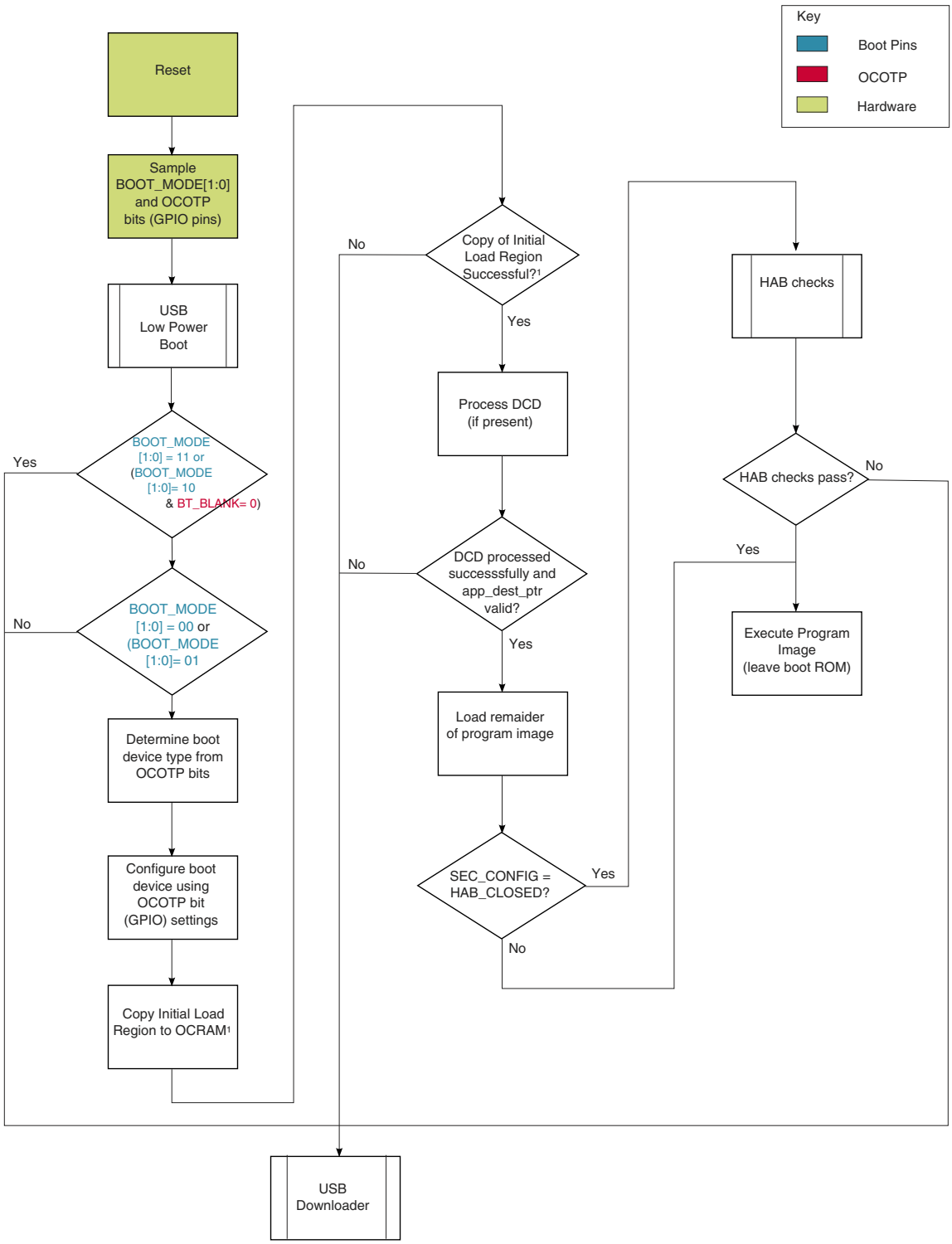


Figure 6-1. i.MX50 ROM Boot Flow

6.2.3 Overview of USB Low Power Boot

The i.MX50 supports a mode where a USB connection can be used to power the device when either the battery level is too low or a battery is not connected to the device. This USB Low Power Boot (LPB) mode feature is enabled by default and can be disabled by blowing the BT_DISABLE_LPB OCOTP bit field.

When a low battery condition is detected, the i.MX50 ROM minimizes the power consumption during the boot process by decreasing the ARM core frequency from 800 MHz/400 MHz to a much lower frequency to meet the power limit constraints of the USB connection. An additional OCOTP bit field BT_LPB_FREQ[1:0] allows users to define the ARM core frequency used in the LPB mode. The lower ARM core frequencies in LPB mode means that the i.MX50 boot times increase but this ensures the constraints of the USB power limits (100 mA) are met. See USB Low-Power Boot, for more details.

6.2.4 Internal Boot (BOOT_MODE [1:0] = 00)

Internal boot mode is selected by driving a value of 00 on the BOOT_MODE[1:0] pins at device power up. In this mode, the processor boots from internal ROM. The boot ROM code performs HW initialization, loads the program image from the chosen boot device, performs image validation using the HAB library (see Boot Security Settings), and then jumps to an address derived from the program image. If any error occurs during internal boot, the boot ROM code jumps to the USB Downloader (see USB Downloader (BOOT_MODE[1:0] = 11)). A secure boot using the HAB on the i.MX50 is possible in all the three boot modes (BOOT_MODE[1:0] = 00, BOOT_MODE[1:0] = 10, and BOOT_MODE[1:0] = 11).

When set to Internal Boot (BOOT_MODE[1:0] = 00) the boot flow may be controlled by a combination of OCOTP bit settings with an option of overriding the fuse setting using GPIO pins. Whether the ROM uses GPIO inputs for a select number of configuration parameters or OCOTP bits is determined by the GPIO Boot Select (BT_FUSE_SEL) OCOTP bit.

- If BT_FUSE_SEL = 1, all boot options are controlled by the OCOTP bits described in Table 6-2.
- If BT_FUSE_SEL = 0, a select number of boot configuration parameters may be set using GPIO pins rather than OCOTP bits. The fuses that can be overridden when in GPIO are indicated in the GPIO column of Table 6-2. Table 6-3 provides the details on the GPIO pins.

The use of GPIO overrides is intended for development since these pads are used for other purposes in deployed products. Freescale recommends controlling the boot configuration by OCOTP bits (BT_FUSE_SEL fuse blown) in deployed products and reserve the use of the GPIO mode (BT_FUSE_SEL fuse intact) for development and testing purposes only.

6.2.5 Internal Boot-Fuses Only (BOOT_MODE [1:0] = 10)

Internal boot-Fuses Only mode is selected by driving a value of 10 on the BOOT_MODE[1:0] pins at device power up. This mode is similar to the Internal boot BOOT_MODE[1:0]=00, with one difference being the GPIO boot override pins are ignored regardless of the BT_FUSE_SEL setting. The boot ROM code uses the boot OCOTP bit settings only. This mode also supports a secure boot using HAB.

If set to Internal Boot - Fuses Only mode, the boot flow is controlled by the BT_BLANK OCOTP bit value. If BT_BLANK = 0, indicating that the boot device (Flash, SD/MMC, and so on.) has not yet been programmed, the boot flow jumps directly to the Serial Downloader. If BT_BLANK = 1, then the normal boot flow is followed, where the ROM attempts to boot from the selected boot device.

The first time a board is used, the default OCOTP bits may be configured incorrectly for the hardware on the platform. In such a case, the ROM code may try to boot from a boot device that does not exist. This may cause an electrical/logic violation on some pads. Using Internal Mode-Fuses Only (BOOT_MODE[1:0] = 10) mode solves this problem. The first time the BT_BLANK OCOTP bit is encountered, it is not blown and forces the ROM to jump directly to the Serial Downloader. This allows a bootloader to be downloaded which can then provision the boot device with a Program Image and blow the BT_BLANK and the other boot configuration OCOTP bits. After a reset, the ROM determines that BT_BLANK is blown and the ROM code performs internal boot according to the new OCOTP bit settings. This allows a user to set BOOT_MODE[1:0]=10 on a production device and burn fuses on the same device (by forcing entry to the Serial Downloader), without changing the value of BOOT_MODE[1:0] or pull-ups/pull-downs on the boot pins.

6.2.6 USB Downloader (BOOT_MODE [1:0] = 11)

The USB Downloader is invoked if the external boot device is not programmed, when a failure is encountered during the boot flow process or any of the following conditions are met:

- BOOT_MODE[1:0] = 11 USB Downloader mode

- BOOT_MODE[1:0] = 10 Internal boot-Fuses Only and the OCOTP bit BT_BLANK = 0
- BOOT_MODE[1:0] = 00 Internal Boot or BOOT_MODE[1:0]=10 Internal Boot-Fuses Only and valid image is not found on the boot device
- Internal failure
- Runtime exception occurs
- Error returned by the HAB functions while SEC_CONFIG = HAB_CLOSED configuration. (Errors are ignored in SEC_CONFIG = HAB_OPENED configuration). See Boot Security Settings.

See USB Downloader (BOOT_MODE[1:0] = 11) for details on the USB Downloader.

6.2.7 Boot Security Settings

All internal boot modes use one of following two secure boot configurations:

- **SEC_CONFIG = HAB_CLOSED** This configuration is intended for use with shipping secure products. All HAB functions are executed and security hardware is initialized, DCD is processed if present, and software in the boot device or software downloaded to RAM is authenticated by HAB prior to its execution. The first error detected is logged, and the boot flow aborted with control passing to the USB Downloader. In this configuration execution does not leave the internal ROM unless the target Program Image has been successfully authenticated.
- **SEC_CONFIG = HAB_OPENED** (default). This configuration is intended for use in non-secure products or during the development phases of a secure product. All HAB functions are executed as for a SEC_CONFIG=HAB_CLOSED configuration: security hardware is initialized, DCD is processed if present, and a Program Image in a boot device or that has been downloaded through the USB Downloader to RAM is authenticated by HAB prior to its execution. The first failure detected is logged, but has no influence and the boot flow continues as if the failure did not occur.

See the SEC_CONFIG entry in Table 6-2 for the specific OCOTP bit field settings.

6.3 Device Configuration

This section describes the external inputs defined that control the behavior of the i.MX50 ROM. This includes boot device selection (NAND, NOR, MMC/SD, SPI, I2C,Toggle-mode NAND, OneNAND), boot device configuration (NAND address cycles), and so on.

In general, the source for this configuration comes from OCOTP bits embedded inside the i.MX50. However, select configuration parameters can be sourced from GPIO pins allowing further flexibility during the development process.

6.3.1 Boot OCOTP bit Descriptions

Table 6-2 shows a comprehensive list of the configuration parameters that the i.MX50 ROM uses.

Table 6-2. Boot OCOTP Bit Descriptions

Fuse	Config	Definition	GPIO[1]	Settings[2]
DIR_BT_DIS	OEM	Reserved FSL test mode disabled.	NA	0 -Reserved FSL test boot allowed 1 - Reserved FSL test boot not allowed
OSC_FREQ_SEL	OEM	CKIH Frequency Select. It is used by boot code for PLL configuration.	Yes	0 - 24Mhz fixed 1 - 26/19.2/27/24 Mhz auto calibration
BT_FUSE_SEL	OEM	GPIO Boot Select. Determines, whether the boot settings indicated by a Yes in the GPIO column are controlled by GPIO pins or OCOTP bit settings in the IIM: **NOTE:** This feature requires BOOT_MODE[1:0] = 00 to operate.	NA	If BOOT_MODE="00" (Development) 0=Boot mode configuration is taken from GPIOs. 1=Boot mode configuration is taken from fuses. If BOOT_MODE="10" (Production) 0 - Boot using Serila Loader (UART/USB) 1- Boot mode configuration is taken from fuses.
BT_BLANK	OEM	Indicates that the boot area has not yet been programmed. **NOTE:** This fuse is only read when BOOT_MODE[1:0]=1(	NA	0 BOOT area is not yet programmed. Boot flow jumps to serial downloader. 1 BOOT area is programmed. Regular boot flow is performed.

Table continues on the next page...

Table 6-2. Boot OCOTP Bit Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Settings[2]
SEC_CONFIG[1:0]	OEM	Security mode configurations. See Boot Security Settings, for definitions.	YES	00 - HAB_FAB Configuration, out of fab configuration. 01 - HAB_OPENED Configuration (allows any Program Image to be executed, even if it has no valid signature) Others - HAB_CLOSED Configuration (Program Images must have valid signatures)
SRK_HASH[255:0]	OEM	256-bit hash value of super root key (SRK_HASH).	NA	Settings vary - used by HAB
DIE-X-CORDINATE[7:0] DIE-Y-CORDINATE[7:0] WAFER_NO[4:0] LOT_NO_ENC[42:40] LOT_NO_ENC[39:32] LOT_NO_ENC[31:24] LOT_NO_ENC[23:16] LOT_NO_ENC[15:8] LOT_NO_ENC[7:0]	Freescale	Device Unique ID, 64-bit UID.	NA	Settings vary - used by HAB
BT_FREQ	OEM	OCOTP bit used to select ARM core frequency at normal boot mode	Yes	0 800Mhz 1 400Mhz
BT_LPB_FREQ[1:0]	OEM	ARM core frequency at Low-Power Boot Mode (LPB).	Yes	00 - 192 MHz (Default[3] - out of reset), 01 - 133 MHz 10 - 55.33 MHz 11 - 220 MHz
BT_MMU_DISABLE	OEM	MMU/D-Cache disable bit used by boot ROM to control HAB processing speed (faster with MMU/D-Cache enabled)	Yes	0 - MMU/D-Cache is enabled by ROM during the boot 1 - MMU/D-Cache is disabled by ROM during the boot
BT_DISABLE_LPB	OEM	Bit used to disable Low-Power Boot Mode (LPB).	No	0 LPB Enabled in default 1 LPB is disabled
BT_DEVICE_1.8V	OEM	OCOTP bit used to enable iomux configuration for 1.8v boot media devices	Yes	0 Boot device is 3.3v(default) 1 Boot device is 1.8v
BT_DISABLE_ROM_REDUNDANT_BOOT	OEM	Disable redundant boot functionality	YES	0 - Redundant boot enabled 1 - Redundant boot disabled

Table continues on the next page...

Table 6-2. Boot OCOTP Bit Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Settings[2]
AXI/DDR_FREQ	OEM	Select PLL2 frequency	YES	0 - PLL2 400 MHz 1 - PLL2 333 MHz
BT_USB_LEGACY	Freescale	Bit used to disable USB HID mode. If blown device will connect to usb host in legacy mode and host will need JUNGO USB driver installed on host machine to enumerate the device	No	0 Device enumerates on host as USB HID device 1 Device enumerates on host in legacy mode if JUNGO driver is installed on host machine
ENABLE_WDOG_TIMER_RESET	Freescale	Bit used to enable wdog timer reset in USB mode	No	0 In USB mode wdog reset timer is not used 1 In USB mode wdog reset timer is used.
DISABLE_WDOG_BOOT	Freescale	Bit used to disable wdog boot feature newly added into i.MX50 ROM	No	0 Watch dog boot enabled 1 Watch dog boot disabled
USB_VID	OEM	16 bits used for specifying USB vendor id	No	default value if not blown is 0x15A2
USB_PID	OEM	16 bits used for specifying USB product id	No	default value if not blown is 0x0052
DISABLE_MMC_FAST_BOOT_ACK at fuse BOOT_CFG3[6]	OEM	By default ROM will enable BOOT_ACK mode of MMC boot register for eSDHCv3. This fuse can be use to disable BOOT_ACK mode.	Yes	0 - ROM will set 1 to BOOT_ACK bit of MMC_BOOT_REGISTER 1 - ROM will set 0 to BOOT_ACK bit of MMC_BOOT_REGISTER
DISABLE_EMMC44_CMD0_RESET at BOOT_CFG3[26]	OEM	For fast boot, by default ROM will send CMD0+0xF0F0F0F0 to reset eMMC card. For earlier versions of eMMC like 4.3 this sequence is not supported so this fuse can be used to force ROM not to send reset cmd to the card.	Yes	0 - fast boot from eMMC4.4 card, CMD0+0xF0F0F0F0 will be sent before fast boot 1 - fast boot from eMMC4.3 card if blown, since (CMD0+F0F0F0F0) reset is not supported by eMMC4.3
ENABLE_MAX_DRIVE_STRENGTH at CFG6[16]	OEM	By default ROM will enable high drive strength for SD pins. If this fuse is set then ROM will set max drive strength for SD pins	No	0 - Not enabled (default to high strength) 1 - Enabled
DLL_SLAVE_OVERRIDE at CFG6[17]	OEM	If this bit is set then ROM will enable DLL SLAVE and DLL will not be enabled otherwise ROM will enable the DLL during eMMC init	No	0 - Use DLL SLAVE 1 - DLL OVERRIDE

Table continues on the next page...

Table 6-2. Boot OCOTP Bit Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Settings[2]
NUMBER_OF_NANDS at CFG6[15:13]	OEM	These fuse bits are used to indicate to the ROM number of NAND chip enables present on the device. This fuse value is useful when enabling internal pull-ups on NAND CE and RDY pins. If there are 8 chip selects then an external pull-up should be added on board. ROM can only enable internal pull-ups for 7 nand chip selects, this is due to a bug in ROM code.	No	000 - 1 001 - 1 010 - 2 011 - 3 100 - 4 101 - 5 110 - 6 111 - 7
ENABLE_INTERNAL_NAND_PULL_UP at CFG6[8]	OEM	If this bit is set then ROM will enable internal pull-ups for NAND chip-enable and ready-busy pins.	No	0 - Not enable internal pull-ups for NAND CE and RDY pins. 1- Enable internal pull-ups for NAND CE and RDY pins
DISABLE_LEVEL1_ICACHE at CFG6[4]	Freescale	By default ROM will enable ICache at startup and this fuse can be used to force ROM to not enable I-Cache.	No	0 - I-Cache will be enabled 1 - I-Cache will not be enabled

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 6-3 for corresponding GPIO pin.
2. 0 = intact otp bit and 1= blown fuse
3. For details refer to Clock Initialization.

6.3.2 GPIO Boot Overrides

Table 6-3 provides a list of GPIO boot overrides. These input pins are sampled at boot, and can be used to override corresponding OCOTP bit values, depending on the setting of the BT_FUSE_SEL fuse. The GPIO override feature is only available when BT_FUSE_SEL is 0 (fuse is unblown) and BOOT_MODE[1:0] = 00.

Table 6-3. GPIO Override Contact Assignments

Contact	Boot Configuration Bits	Details
BOOT_MODE0	BOOT_MODE[0]	
EIM_DA0	BOOT_CFG1[0]	BT_MMU_DISABLE
EIM_DA1	BOOT_CFG1[1]	BT_FREQ
EIM_DA2	BOOT_CFG1[2]	BT_1.8V_BOOT
EIM_DA3	BOOT_CFG1[3]	-
EIM_DA4	BOOT_CFG1[4]	-
EIM_DA5	BOOT_CFG1[5]	-

Table continues on the next page...

Table 6-3. GPIO Override Contact Assignments (continued)

Contact	Boot Configuration Bits	Details
EIM_DA6	BOOT_CFG1[6]	-
EIM_DA7	BOOT_CFG1[7]	-
EIM_DA8	BOOT_CFG2[0]	SEC_CONFG[0]
EIM_DA9	BOOT_CFG2[1]	SEC_CONFIG[1]
EIM_DA10	BOOT_CFG2[2]	DISABLE_REDUNDANT_BOOT
EIM_DA11	BOOT_CFG2[3]	OSC_FREQ_SEL
EIM_DA12	BOOT_CFG2[4]	BT_DDR_FREQ
SSI_TXFS	CFG4[10]	BT_LPB_FREQ[0]
SSI_TXC	CFG4[11]	BT_LPB_FREQ[1]

6.3.3 Device Configuration Data

See Device Configuration Data (DCD), for more details on Device Configuration Data.

6.4 Device Initialization

This section describes the details on the i.MX50 ROM and the provides initialization details. This includes details on:

- The OCROM Memory Map
- The OCRAM Memory Map
- On-chip modules that the ROM should make use of or change POR register default values
- Clock initialization
- Enabling the MMU/L2 cache
- Exception handling, error logging, and interrupt handling

6.4.1 Memory Map

Figure 6-2 shows the OCROM and OCRAM memory map for the i.MX50 Memory Map.

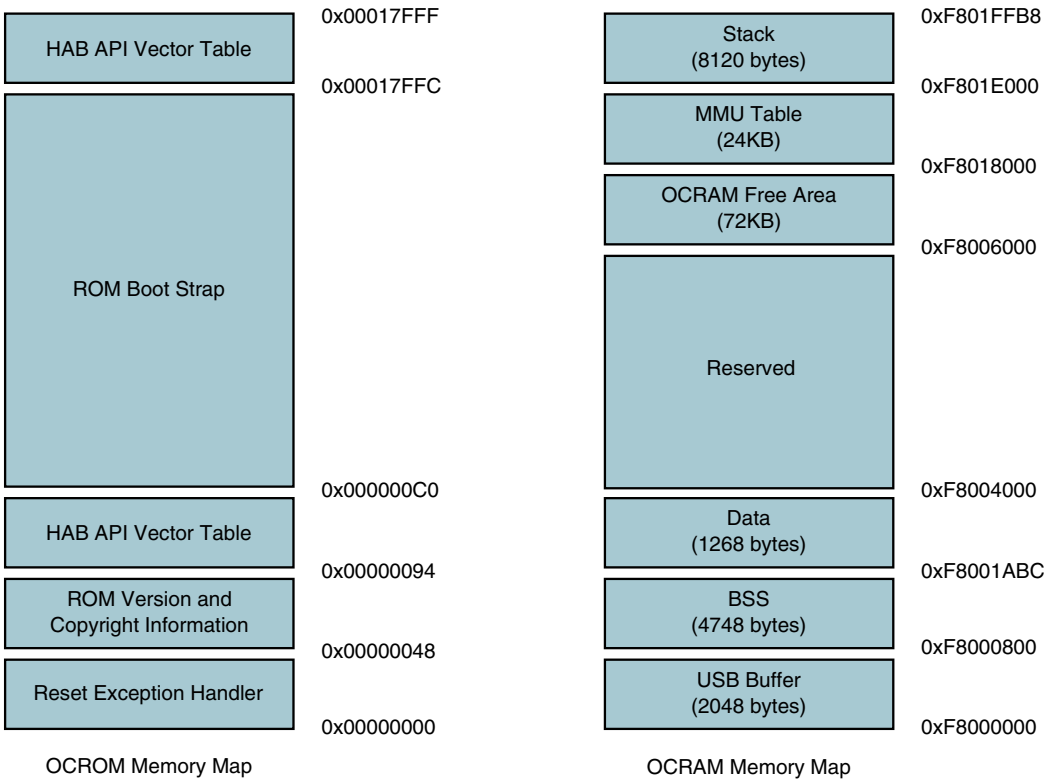


Figure 6-2. OCROM and OCRAM Memory Map

The OCRAM free space occupies the address from 0xF8006000~0xF8017FFF occupying a total of 72 Kbytes. The following 24 Kbytes of OCRAM is reserved for MMU page tables when the BT_MMU_DISABLE OCOTP bit is not blown. When the BT_MMU_DISABLE OCOTP bit is blown the OCRAM space from 0xF8006000-0xF801DFFF may be treated as OCRAM free space. See Enabling MMU/ Caches, for additional information on enabling the MMU.

NOTE

When the BT_MMU_DISABLE OCOTP bit is not blown, don't write any data to the MMU Page Table area in OCRAM. Doing so will cause the i.MX50 to fail boot.

6.4.2 Boot Module Activation

The i.MX50 boot ROM affects a number of different hardware modules which are activated and play a vital role in the boot flow. The i.MX50 ROM configures and uses the following modules (listed in alphabetical order) during the boot process. Note that the modules actually used depend on the boot mode and boot device selection:

- CCM-Clock Control Module

- CSPI/eCSPI-Configurable Serial Peripheral Interface and enhanced Configurable Serial Peripheral Interface
- DCP-Data Co-Processor
- DRAM MC(EIM)-DRAM Memory Controller. Only used for booting from an external memory device using the EIM.
- eSDHC/v3-enhanced Secure Digital Host Controller
- I2C-Inter IC
- IOMUX- I/O Multiplexor allows using the GPIO to override OCOTP bit boot settings.
- OCOTP-On Chip One Time Programming bits
- PLL- Phase Locked Loop. Also called Digital Phase Locked Loop (DPLL)
- SRC-System Reset Controller
- GPT-General Purpose Timer
- SRTC-Secure Real Time Clock
- USB-Used for serial download of a boot device provisioning program
- WDOG-Watchdog Timer
- BCH-32-bit error correction HW engine with AXI bus master and private connection to GPMI.
- GPMI-NAND controller pin interface.
- APBH_DMA-DMA engine to drive the GPMI module.

6.4.3 Clocks at Boot Time

The clocks that are enabled (and their frequencies) when ROM begins execution are shown in Table 6-5. This table also describes the peripheral clock sources.

6.4.4 Clock Initialization

On reset, the processor has access to all shared peripherals and by default all peripheral clocks are enabled and the boot ROM does not change any of the peripheral clock enables. The clocks are initialized depending on whether a normal boot or a USB low power boot is performed.

6.4.4.1 NORMAL Mode

In normal mode the following frequencies are available as input clock to PLL1-3 modules. Selection of these frequencies is accomplished as follows:

1. The CLKSS is read and if OSC is selected then further input clock selection is based on the OCOTP bit OSC_FREQ_SEL (Refer to Table 6-4).
2. If the CLKSS is FPM, then 32 kHz is selected as input to PLL1-3 module.

Table 6-4. OSC_FREQ_SEL Frequencies

OSC_FREQ_SEL	Oscillator Frequency	LPB Frequency BT_LPB_FREQ = 0
0	24 MHz	192 MHz
1	26/19.2/27/24 MHz Auto Calibration	208/153.2/216/192 MHz

NOTE: Although ROM code can calibrate the frequency of oscillator on board when OSC_FREQ_SEL is blown, it is not recommended to do so, because auto calibration of frequency will take 10ms to calculate, furthermore 24 MHz oscillator is recommended.

Table 6-5 shows PLL1-PLL3 with their associated frequencies.

Table 6-5. PLL Frequencies

Clock	Frequency (MHz)
PLL1	800/400
PLL2	400/333
PLL3	216

Table 6-6 shows the various clocks used and the clock sources used.

Table 6-6. Normal Clock Configuration

	CCM Signal	Source	Freq [MHz] Fuse AXI/DDR Freq = 0 and BT_FREQ=0	Freq [MHz] Fuse AXI/DDR Freq = 0 and BT_FREQ=1	Freq [MHz] Fuse AXI/DDR Freq = 1 and BT_FREQ=0	Freq [MHz] Fuse AXI/DDR Freq = 1 and BT_FREQ=1
ARM Core	arm_clk_root	PLL1	800	400	800	400
GPMI rawnand	gpmi_clk_root	OSC	24	24	24	24
GPMI Togglemode NAND	gpmi_clk_root	PFD0 or PLL1	33/40/66	33/40/66	33/40/66	33/40/66
DRAM MC slow	emi_slow_clk_root	PLL2	100	100	83.25	83.25
AHB	ahb_clk_root	PLL2	133	133	110	110
IPG	ipg_clk_root	PLL2	66.5	66.5	55	55
AXI_A	axi_a	PLL2	400	400	133	133
AXI_B	axi_b	PLL2	200	200	166.5	166.5

Table continues on the next page...

Table 6-6. Normal Clock Configuration (continued)

	CCM Signal	Source	Freq [MHz] Fuse AXI/DDR Freq = 0 and BT_FREQ=0	Freq [MHz] Fuse AXI/DDR Freq = 0 and BT_FREQ=1	Freq [MHz] Fuse AXI/DDR Freq = 1 and BT_FREQ=0	Freq [MHz] Fuse AXI/DDR Freq = 1 and BT_FREQ=1
eSDHC1	esdhc1_clk_root	PLL2	80	80	66.6	66.6
eSDHC2	esdhc2_clk_root	PLL2	80	80	66.6	66.6
eSDHC3	esdhc3_clk_root	PLL2	80	80	66.6	66.6
eSDHC4	esdhc4_clk_root	PLL2	80	80	66.6	66.6
eCSPI	eCSPI_clk_root	PLL3	54	54	54	54
CSPI	ipg_clk_root	IPG_CLK	66.5	66.5	55	55
I2C	per_clk_clk	PER_CLK	20	20	16.65	16.65
USBOH1_PHY	Oscillator	OSC	26/19.2/27/24 See Table 6-4	26/19.2/27/24 See Table 6-4	26/19.2/27/224 See Table 6-4	26/19.2/27/24 See Table 6-4

In addition to these clocks, the PERCLK_ROOT clock is generated by dividing PLL2 as a source to I^2C module and the CSPI_CLOCK_ROOT using IPG_ROOT (66.5 MHz).

6.4.4.2 USB Low-Power Boot Mode (LPB)

In MX50, LPB support is implemented by only decreasing the ARM core frequency, and the frequency of other modules will be the same with that of normal boot mode. The ROM code configures the ARM core frequency value based on the setting of the OCOTP bit BT_LPB_FREQ[1:0]. The LPB ARM core frequency defaults to 192 MHz out of reset. The LPB ARM core frequency can change based on the BT_LPB_FREQ[1:0] as shown in Table 6-7.

Table 6-7 lists the various LPB frequencies used by different clocks based on the value of BT_LPB_FREQ[1:0].

Table 6-7. ARM Frequency of Low Power Boot Mode

BT_LPB_FREQ[1:0]	00	01	10	11
PLL1	192	266	166	220
ARM core	192 Mhz	133 Mhz	55.3 Mhz	220 Mhz

6.4.5 Enabling MMU/Caches

Boot ROM includes a feature of enabling the MMU and caches to improve boot speed when performing image authentication using HAB library (High Assurance Boot (HAB). L1 instruction cache is enabled at the start of image download. L1 data cache, L2 cache and MMU are enabled during image authentication. The MMU and caches are enabled by default. If the BT_MMU_DISABLE fuse bit is blown, then MMU and caches are disabled during boot.

6.4.6 Error Logging

The HAB status is logged to a single 32-bit word. The value can be obtained in USB HID mode using the SDP command ERROR_STATUS. For details on HAB status codes please contact your local Freescale representative.

6.4.7 Exception Handling

The exception vectors located at the start of OCROM are used to map all the ARM exceptions (except the reset exception) to a duplicate exception vector table in internal RAM. Copying the exception vectors to OCRAM allows the vectors to be changed by later stages in the boot chain.

During the boot phase, the OCRAM vectors point to the USB Downloader in OCROM. After boot the OS loader can overwrite the vectors as required. The OCRAM exception table is filled with the address of the USB Downloader entry point such that any exception or interrupt results in execution of the USB Downloader. The code shown is used to map the ROM exception vector table to the duplicate one in OCRAM.

Mapping ROM Exception Vector Table

```
;; Define linker area for OCROM exception vector table
AREA IROM_VECTORS, CODE, READONLY


LDR   PC, Reset_Addr
LDR   PC, Undefined_Addr
LDR   PC, SWI_Addr
LDR   PC, Prefetch_Addr
LDR   PC, Abort_Addr
NOP         ; Reserved vector
LDR   PC, IRQ_Addr
LDR   PC, FIQ_Addr
LDR   PC, SW_monitor_addr

;; Define exception vector table
Reset_Addr      DCD     start_address
Undefined_Addr  DCD     OCRAM_Undefined_Handler
SWI_Addr        DCD     OCRAM_SWI_Handler
Prefetch_Addr   DCD     OCRAM_Prefetch_Handler
```

```
Abort_Addr      DCD     OCRAM_Abort_Handler
DCD  0                ; Reserved vector
IRQ_Addr        DCD     OCRAM_IRQ_Handler
FIQ_Addr        DCD     OCRAM_FIQ_Handler
SW_monitor_add  DCD   SW monitor exception

start_address   DCD start ;reset handler vector
```

6.4.8 Interrupt Handling During Boot

No special interrupt handling routines are required during the boot process. Interrupts are disabled during boot ROM execution and may be enabled in a later boot stage.

6.5 Boot Devices (Internal Boot)

The i.MX50 supports the following boot devices:

- NOR flash with EIM Interface, located on CS0, with only 16-bits bus width supported.
- OneNAND flash with EIM interface, located on CS0 with only 16-bit bus width supported.
- Raw NAND (MLC and SLC), and Toggle-mode NAND flash through GPMI-2 interface. Page sizes of 512 bytes, 2 Kbyte, 4 Kbyte and 8 Kbyte. Bus widths of 8-bit with 2 through 32-bit BCH Hardware ECC (Error Correction) supported.
- SD/MMC/eMMC and eSD through eSDHC interface, supporting high capacity cards. eMMC-4.3/4.4 boot mode (FAST BOOT) through eSDHCv3 port. The eMMC4.4 DDR mode is also supported.
- eSD FAST BOOT mode through all the eSDHC ports.
- Serial ROM boot via SPI (serial flash or EEPROM) and I^2C EEPROM.

The selection of boot device type is controlled by BT_MEM_CTL[1:0] and BT_MEM_TYPE[1:0] OCOTP bits. See Table 6-2 for the specific OCOTP bit settings.

6.5.1 NOR/OneNAND Boot from EIM Interface

The NOR flash interface works in the asynchronous mode and supports either muxed Address/Data or non-muxed schemes based on OCOTP bit settings, see Table 6-8 for details.

Table 6-8. EIM Boot OCOTP Bits Descriptions

Fuse	Config	Definition	GPIO[1]	Settings
BOOT_CFG1[7:4]	OEM	Boot Device Selection	Yes	0000 - Boot from EIM Interface
BOOT_CFG1[3]	OEM	NOR/OneNand Selection	Yes	0 - NOR 1 - OneNand
BOOT_CFG2[7:6]	OEM	Muxing Scheme	Yes	00 - Muxed, 16-bit data (low half) interface 01 - Not muxed, 16-bit data (high half) interface 10 - Reserved 11 - Reserved

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 6-3 for corresponding GPIO pin.

6.5.1.1 IOMUX Configuration for EIM

Because all i.MX50 pads are configured as GPIO input out of reset, IOMUX configuration is required even for EIM interface.

Table 6-9. IOMUX Configuration of EIM Interface

A0	EIM_DA0.ALT0	A26	DISP_RS.ALT3	D8	EPDC_D8.ALT2
A1	EIM_DA1.ALT0	A27	DISP_CS.ALT3	D9	EPDC_D9.ALT2
A2	EIM_DA2.ALT0	CS3	DISP_CS.ALT4	D10	EPDC_D10.ALT2
A3	EIM_DA3.ALT0	CS2	EIM_CS2.ALT2	D11	EPDC_D11.ALT2
A4	EIM_DA4.ALT0	CS1	EIM_CS2.ALT2	D12	EPDC_D12.ALT2
A5	EIM_DA5.ALT0	CS0	EIM_CS2.ALT2	D13	EPDC_D13.ALT2
A6	EIM_DA6.ALT0	EB0	EIM_EB0.ALT0	D14	EPDC_D14.ALT2
A7	EIM_DA7.ALT0	EB1	EIM_EB1.ALT0	D15	EPDC_D15.ALT2
A8	EIM_DA8.ALT0	EB2	EIM_PWRCTRL3.ALT2	D16	EPDC_GDCLK.ALT2
A9	EIM_DA9.ALT0	EB3	EIM_VCOMO.ALT2	D17	EPDC_GDSP.ALT2
A10	EIM_DA10.ALT0	WAIT	EIM_WAIT.ALT0	D18	EPDC_GDO2.ALT2
A12	EIM_DA12.ALT0	RDY	EIM_RDY.ALT0	D20	EPDC_SDCLK.ALT2
A13	EIM_DA13.ALT0	OE	EIM_OE.ALT0	D21	EPDC_SDOEZ.ALT2
A14	EIM_DA14.ALT0	RW	EIM_RW.ALT0	D22	EPDC_SDOED.ALT2
A15	EIM_DA15.ALT0	LBA	EIM_LBA.ALT0	D23	EPDC_SDOE.ALT2
A16	DISP_D0.ALT3	CRE	EIM_CRE.ALT0	D24	EPDC_SDLE.ALT2
A17	DISP_D0.ALT3	DTACK_B	EIM_WAIT.ALT2	D25	EPDC_SDSP.ALT2
A18	DISP_D0.ALT3	D0	EPDC_D0.ALT2	D26	EPDC_SDSHR.ALT2

Table continues on the next page...

Table 6-9. IOMUX Configuration of EIM Interface (continued)

A19	DISP_D0.ALT3	D1	EPDC_D1.ALT2	D27	EPDC_PWRCOM.ALT2
A20	DISP_D0.ALT3	D2	EPDC_D2.ALT2	D28	EPDC_PWRSTAT.ALT2
A21	DISP_D0.ALT3	D3	EPDC_D3.ALT2	D29	EPDC_PWRCTRL0.ALT2
A22	DISP_D0.ALT3	D4	EPDC_D4.ALT2	D30	EPDC_PWRCTRL1.ALT2
A23	DISP_D0.ALT3	D5	EPDC_D5.ALT2	D31	EPDC_PWRCTRL2.ALT2
A24	DISP_WR.ALT3	D6	EPDC_D6.ALT2		

[1] The shadowed cells are the pins which are not configured in ROM code.

[2] D[31:16] will be configured for non-mux addressing mode.

6.5.1.2 NOR Flash Boot Operation

Booting from the NOR Flash is supported via EIM interface. The ROM reads Image Vector Table and Boot Data structures to determine if the image can be executed directly from EIM address space or should be copied to other memory. The start field of Boot Data Structure specifies the final location of the image.

6.5.1.3 OneNAND Flash Boot Operation

The OneNAND flash devices are only available with a 16-bit interface. The ROM OneNAND Flash driver reads the device page size by issuing a software command to device and collecting its response.

At system power-up, OneNAND automatically copies 1 Kbyte data from start of Flash array (sector 0 and sector 1, page 0, block 0) to its Boot RAM, (oneNAND's internal RAM). For example, the Boot RAM contains the OneNAND Image Vector Table. Boot ROM copies the 1 Kbyte OneNAND Boot RAM contents to the application destination pointer (located in the boot_data structure of Image Vector Table (IVT) and decrements the length of the image to be read from OneNAND by 1k. Because the Boot RAM area is memory mapped, the copy operation is a simple memory copying operation. The length of image to be read from the OneNAND device is specified in Image Vector Table structure described in Figure 6-15. Failure loading data from OneNAND flash for any reason forces the processor Boot ROM to switch to serial download. Due to limited 1Kbyte size of OneNAND Boot RAM both IVT and DCD must fit in first 1Kbyte. This limits the max size of DCD for OneNAND device to 736 bytes as specified in Table 6-22. ROM reads rest of image and copies it to destination address during HAB authentication process.

At system power-up, the voltage detector in the device detects the rising edge of Vcc and releases internal power-up reset signal which triggers bootcode loading. Bootcode loading means that the boot loader in the OneNAND copies the designated sized data (1Kbyte) from the beginning of memory to the BootRAM. The Bootcode copy operation starts 400 µs later than POR activation and the 1Kbyte Bootcode copy takes 70 µs (estimated) from sector0 and sector1/page0/block0 of NAND Flash array to the BootRAM. The INT bit of Interrupt status register goes from 'Low' to 'High' on the condition of 'Bootcode-copy done' and RP rising edge.

Boot ROM implements OneNAND initialization in two stages:

- Use the GPT timer to introduce a delay of around 500 µS.
- Wait for INT bit on the Interrupt status register to go High. Once INT bit of interrupt status register is set to High, Boot ROM proceeds with OneNAND initialization.

NOTE

Image header, DCD and boot data must reside in the initial load region (1K) of bootable image. Image length must be specified in the boot data structure pointed to by the image header boot data element, please see Figure 6-15 for image vector table structure

6.5.2 Raw NAND Boot from GPMI-2 Interface

A number of MLC/SLC NAND flash devices, from various vendors, are supported by the i.MX50 ROM. The BCH32 ECC module is used for error detection and correction.

6.5.2.1 NAND eFUSE/GPIO Configuration

The i.MX50 ROM determines the configuration of an external NAND Flash by the following parameters, provided either by eFUSE or GPIO pins sampled at power up.

To control the external NAND device parameters, use the BOOT_CFG1, BOOT_CFG2 and BOOT_CFG3 and the CFG4 eFuses.

See Table 6-10 for the specific eFUSE settings.

Table 6-10. NAND Boot Device Fuse Settings

Fuse	Config	Definition	GPIO[1]	Settings
BOOT_CFG1[7]	OEM	Boot Device Selection	Yes	1 - Boot from Nand Interface

Table continues on the next page...

Table 6-10. NAND Boot Device Fuse Settings (continued)

Fuse	Config	Definition	GPIO[1]	Settings
BOOT_CFG1[6]	OEM	GPMI_CLK_SRC_PLL1	Yes	Only used if BT_TOGGLEMODE fuse is blown. 0 - GPMI and BCH CLK sourced from PFD4 1 - GPMI and BCH CLK sourced from PLL1 If BT_TOGGLEMODE fuse is not blown then ROM uses 24 MHz OSC as source for deriving GPMI and BCH CLKs
BOOT_CFG1[5]	OEM	BT_TOGGLEMODE	Yes	0 - Boot from legacy nand flash 1 - Boot from Samsung's togglemode DDR nand flash
BOOT_CFG1[4:3]	OEM	GPMI cle/ale/ce/ready/dqs MUX on	Yes	00 - Kpp 01 - Eim 10 - Elcdif 11 - Reserved
BOOT_CFG2[7:5]	OEM	Toggle Mode 33 MHz Preamble Delay, Read Latency	Yes	3 bits used to specify toggle mode NAND timing parameters preamble delay and read latency, default if not programmed will be a value of 2. Same value is used for both parameters. ROM will overwrite these with values read from FCBi
BOOT_CFG3[7:6]	OEM	Nand_Row_address_bytes	Yes	00 - 3 address cycles 01 - 2 address cycles 10 - 4 address cycles 11 - 5 address cycles
BOOT_CFG3[5:4]	OEM	BOOT_SEARCH_COUNT	Yes	00 - 2 01 - 2 10 - 4 11 - 8
BOOT_CFG3[3:2]	OEM	Pages In Block	Yes	00 - 128 01 - 64 10 - 32 11 - Reserved
CFG4[23:16]	OEM	NAND_READ_CMD_CODE1	No	Command for read action in the first address cycle
CFG4[15:8]	OEM	NAND_READ_CMD_CODE2	No	Command for read action in the second address cycle

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 6-3 for corresponding GPIO pin.

6.5.2.2 NAND Flash Boot Flow and Boot Control Blocks(BCB)

There are two BCB data structures: FCB and DBBT. As part of the NAND media initialization, the ROM driver uses safe NAND timings to search for a Firmware Configuration Block (FCB) that contains the optimum NAND timings, page address of Discovered Bad Block Table (DBBT) Search Area and start page address of primary and secondary firmware.

The hardware ECC level to use is embedded inside FCB block. The FCB data structure is itself protected

using software ECC (SEC-DED Hamming Codes). Driver reads raw 2112 bytes of first sector and runs through software ECC engine that determines whether FCB data is valid or not.

If the FCB is found, the optimum NAND timings are loaded for further reads. If the ECC fails, or the fingerprints do not match, the Block Search state machine increments page number to Search Stride number of pages to read for the next BCB until SearchCount pages have been read.

If search fails to find a valid FCB, the NAND driver responds with an error and the boot ROM enters into serial download mode.

The FCB contains the page address of DBBT Search Area, and the page address for primary and secondary boot images. DBBT is searched in DBBT Search Area just like how FCB is searched. After the FCB is read, the DBBT is loaded, and the primary or secondary boot image is loaded using starting page address from FCB.

See the flow of FCB search in Figure 6-3.

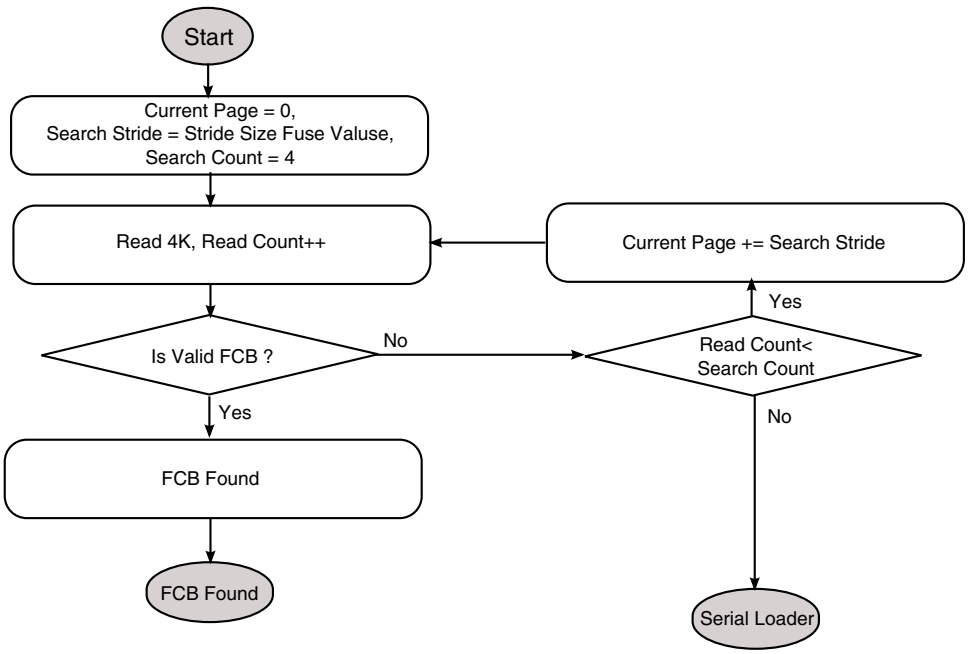


Figure 6-3. FCB Search Flow

Once FCB found the boot ROM starts searching for Discovered Bad Blocks Table (DBBT). If DBBT Search Area is 0 in FCB, then ROM assumes that there are no bad blocks on NAND device. See Figure 6-4 for DBBT search flow.

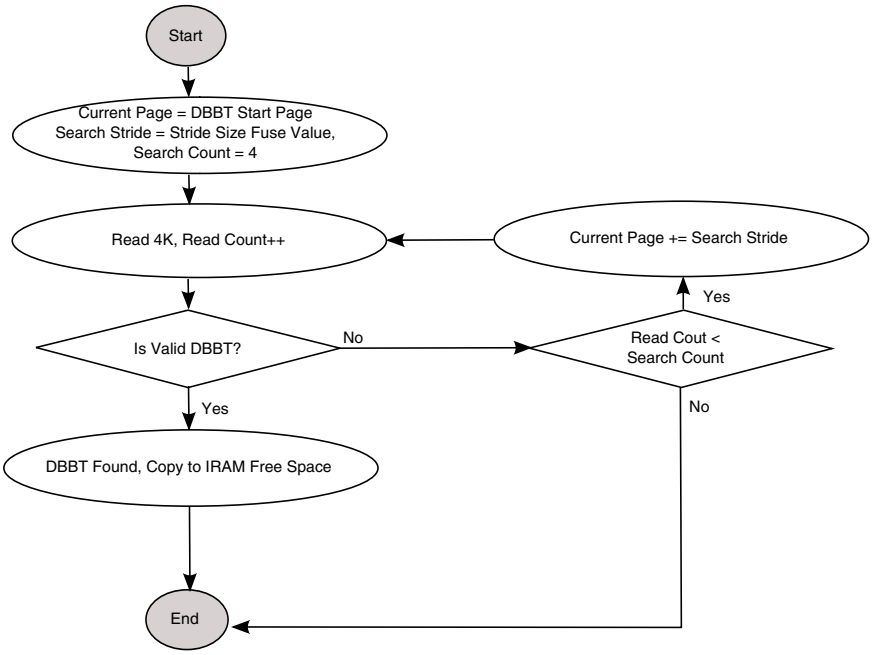


Figure 6-4. DBBT Search Flow

If during primary image read there was a page with number of errors higher than ECC can correct, the boot

ROM will turn on PERSIST_SECONDARY_BOOT bit and perform SW reset. (After SW reset secondary image will be used).

If during secondary image read there was a page with number of errors higher than ECC can correct, the boot ROM will go to serial loader.

6.5.2.3 Firmware Configuration Block

The FCB is the first sector in the first good block. The FCB should be present at each search stride of the search area. The search area contains copies of the FCB at each stride distance, so in case the first NAND block becomes corrupted, the ROM will find its copy in the next NAND block. The search area should span over at least two NAND blocks. The location information for DBBT search area, FW1, and FW2 are all specified in the FCB. Flash Control Block Structure is as shown in Table 6-11.

Table 6-11. Flash Control Block Structure

Name	Start Byte	Size in Bytes	Description
Reserved	0	4	Reserved for Fingerprint #1(Checksum)

Table continues on the next page...

Table 6-11. Flash Control Block Structure (continued)

Name	Start Byte	Size in Bytes	Description
FingerPrint	4	4	32 bit word with a value of 0x4E434220, in ascii "FCB"
Version	8	4	32-bit version number; this version of FCB is 0x00000001
m_NANDTiming	12	4	8 bytes of data for 8 NAND Timing Parameters from NAND datasheet. The 8 parameters are: data_setup, data_hold, address_setup, dsample_time, nand_timing_state, REA, RLOH, RHOH
PageDataSize	20	4	Number of bytes of data in a page. Typically, this is 2048 bytes for 2112 bytes page size or 4096 bytes for 4314 bytes page size
TotalPageSize	24	4	Total number of bytes in page. Typically, 2112 for 2 K page or 4224 or 4314 for 4 K page.
SectorsPerBlock	28	4	Number of pages per block. Typically 64 or 128 or depending on NAND device type.
NumberOfNANDs	32	4	Not used by ROM
TotalInternalDie	36	4	Not used by ROM
CellType	40	4	Not used by ROM
EccBlockNEccType	44	4	Value from 0 to 16 used to set BCH Error Corrrection level 0, 2, 4, .. or 32 for Block B0 of ECC page, used in configuring BCH32 page layout registers
EccBlock0Size	48	4	Size of block B0, used in configuring BCH32 page layout registers
EccBlockNSize	52	4	Size of block BN, used in configuring BCH32 page layout registers
EccBlock0EccType	56	4	Value from 0 to 16 used to set BCH Error Corrrection level 0, 2, 4, .. or 32 for Block BN of ECC page, used in configuring BCH32 page layout registers
MetadataBytes	60	4	Size of metadata bytes used in configuring BCH32 page layout registers
NumEccBlocksPerPage	64	4	Number of ECC blocks BN not including B0. This value is used in configuring BCH32 page layout registers
EccBlockNEccLevelSDK	68	4	Not used by ROM

Table continues on the next page...

Table 6-11. Flash Control Block Structure (continued)

Name	Start Byte	Size in Bytes	Description
EccBlock0SizeSDK	72	4	Not used by ROM
EccBlockNSizeSDK	76	4	Not used by ROM
EccBlock0EccLevelSDK	80	4	Not used by ROM
NumEccBlocksPerPageSDK	84	4	Not used by ROM
MetadataBytesSDK	88	4	Not used by ROM
EraseThreshold	92	4	Not used by ROM
Firmware1_startingPage	104	4	Page number address where first copy of bootable firmware is located
Firmware2_startingPage	108	4	Page number address where second copy of bootable firmware is located
PagesInFirmware1	112	4	Size of first copy of firmware in pages
PagesInFirmware2	116	4	Size of second copy of firmware in pages
DBBTSearchAreaStartAddress	120	4	Page address for bad block table search area
BadBlockMarkerByte	124	4	This is an input offset in BCH page for ROM to swap with first byte of metadata after reading a page using BCH32. ROM supports restoration of manufacturer marked bad block markers in the page and this offset is the bad block marker offset location
BadBlockMarkerStartBit	128	4	This is an input bit offset in BadBlockMarkerByte for ROM to use when swapping 8 bits with fiirst byte of metadata.
BBMarkerPhysicalOffset	132	4	This is the offset where manufacturer leaves bad block marker on a page
BCHType	136	4	0 for BCH20 and 1 for BCH32. MX50 is backward compatible to BCH20 and this field tell ROM to use BCH20 or BCH32 block
TMTiming2_ReadLatency	140	4	Toggle mode NAND timing parameter read latency, ROM use this vlue to configure timing2 register of GPMI
TMTiming2_PreambleDelay	144	4	Toggle mode NAND timing parameter Preamble Delay. ROM use this value to configure timing2 register of GPMI
TMTiming2_CEDelay	148	4	Toggle mode NAND timing parameter CE Delay. ROM use this value to configure timing2 register of GPMI
TMTiming2_PostambleDelay	152	4	Toggle mode NAND timing parameter Postamble Delay. ROM use this value to configure timing2 register of GPMI

Table continues on the next page...

Table 6-11. Flash Control Block Structure (continued)

Name	Start Byte	Size in Bytes	Description
TMTiming2_CmdAddPause	156	4	Toggle mode NAND timing parameter Cmd Add Pause. ROM use this value to configure timing2 register of GPMI
TMTiming2_DataPause	160	4	Toggle mode NAND timing parameter Data Pause. ROM use this value to configure timing2 register of GPMI
TMSpeed	164	4	This is the toggle mode speed for ROM to configure gpmi clock at. 0 for 33 MHz, 1 for 40 MHz and 2 for 66 MHz
TMTiming1_BusyTimeout	168	4	Toggle mode NAND timing parameter Busy Timeout. ROM use this value to configure timing1 register of GPMI
DISBBM	172	4	If 0 ROM will swap BadBlockMarkerByte with metadata[0] after reading a page using BCH32. If the value set is 1 then ROM will not do swapping
BBMark_Spare_Offset	176	4	This is the offset for the metadata area. It stores the data marked by the bad block marker.

6.5.2.4 Discovered Bad Block Table

See Table 6-12 for DBBT format.

Table 6-12. DBBT Structure

Name	Start Byte	Size in Bytes	Description
reserved	0	4	-
FingerPrint	4	4	32-bit word with a value of 0x44424254,in ascii "DBBT"
Version	8	4	32-bit version number; this version of DBBT is 0x00000001
reserved	12	4	-
DBBT_NUM_OF_PAGES	16	4	Size of DBBT in pages
reserved	20	4*PageSize-20	-
reserved	4*PageSize	4	-
Number of Entries	4*PageSize + 4	4	Number of bad blocks
Bad Block Number	4*PageSize + 8	4	First bad block number
Bad Block Number	4*PageSize + 12	4	Second bad block number
...-	-	-	...next bad block number

Table continues on the next page...

Table 6-12. DBBT Structure (continued)

Name	Start Byte	Size in Bytes	Description
...-	-	-	...-
Last bad block number	-	-	last bad block number

6.5.2.5 Bad Block Handling in the ROM

During firmware boot, at the block boundary, the Bad Block table is searched for a match to the next block. If no match is found, the next block can be loaded. If a match is found, the block must be skipped and the next block checked.

If Bad Block table start page is null, check the manufactory made Bad Block marker. The location of Bad Block maker is at the first 3 or last 3 pages in every block of the nand flash. Nand manufacturers normally use one byte in the spare area of certain pages within a block to mark a block is bad or not. 0xFF means good block, non FF means bad block.

In order to preserve the BI (bad block information), flash updater or gang programmer applications need to swap Bad Block Information (BI) data to byte 0 of metadata area for every page before programming nand flash. ROM when loading firmware copies back the value at metadata[0] to BI offset in page data. Figure 6-5 shows how the factory bad block marker is preserved.

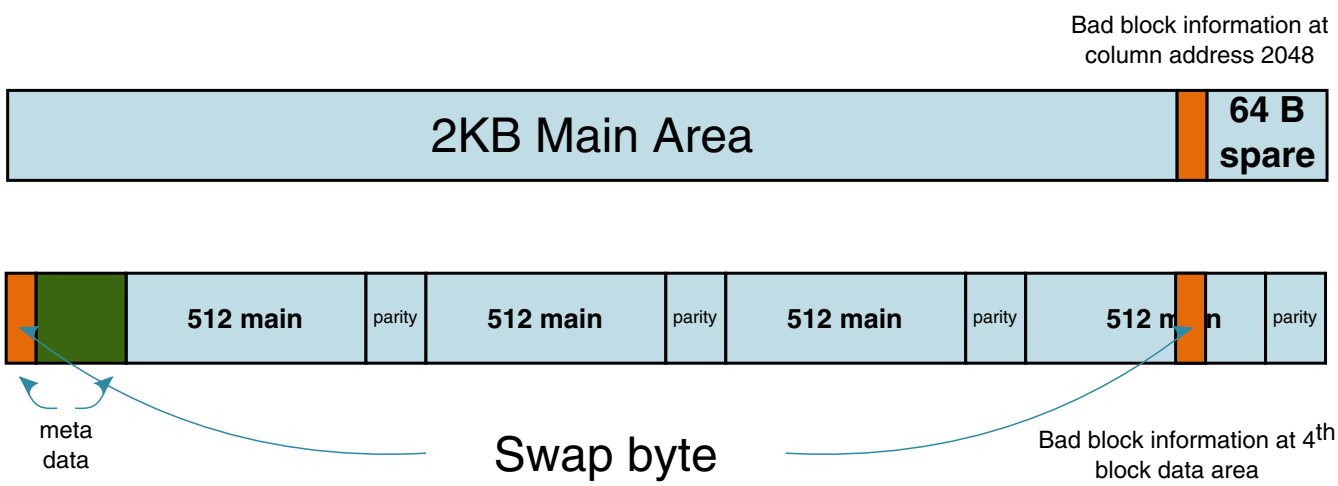


Figure 6-5. Factory Bad Block Marker Preservation

In the FCB structure, there are two elements BadBlockMarkerByte and BadBlockMarkerStartBit to indicate the byte and bit place in the page data, that manufacturer marked the bad block marker.

6.5.2.6 Toggle Mode DDR NAND Boot

If BT_TOGGLEMODE efuse is blown then ROM does the following to boot from Samsung's toggle mode DDR NAND.

6.5.2.6.1 Configures GPMI and BCH Clocks to Run at 33 MHz

if GPMI_CLK_SRC_PLL1 efuse is set then ROM will derive gpmi and bch clocks from PLL1 source otherwise will derive from 480 MHz PFD4 PLL. ROM sets the dividers in CCM registers to run GPMI CLK and BCH CLK at default 33 MHz. ROM sets default values to timing0, timing1 and timing2 gpmi registers for 33 MHz clock speed. It uses fuse BOOT_CFG2[7:5] to configure GPMI timing2 register paramters preamble delay and read latency, the default value for these parameters is 2 when fuses are not blown.

Default timing parameter values used by ROM for toggle-mode device:

- Timing0.ADDRESS_SETUP = 5
- Timing0.DATA_SETUP = 10
- Timing0.DATA_HOLD = 10
- Timing1.DEVICE_BUSY_TIMEOUT = 0 x 500
- Timing2.READ_LATENCY = BOOT_CFG2[7:5] if blown, otherwise 2
- Timing2.CE_DELAY = 2
- Timing2.PREAMBLE_DELAY = BOOT_CFG2[7:5] if blown, otherwise 2
- Timing2.POSTAMBLE_DELAY = 3
- Timing2.CMDADD_PAUSE = 4
- Timing2.DATA_PAUSE = 6

6.5.2.6.2 Setup DMA for DDR Transfers

In DMA descriptors GPMI is configured to read page data at double data rate, the word length is set to 16 and transfer count to half of page size.

6.5.2.6.3 Reconfigure Timing and Speed Using Values in FCB

After reading FCB page with GPMI set to default timings and speed 33 MHz, ROM reconfigures CCM dividers to run gpmi/bch clks to desired speed specified in FCB for rest of boot process. The GPMI timing registers are also reconfigured to values specified in FCB.

The GPMI can be configured to run at 33 MHz/40 MHz/66 MHz using FCB parameter TMSpeed, 0 for 33, 1 for 40 and 2 for 66 MHz speed.

The GPMI timing0 register fields data_setup, data_hold and address_setup are set to values specified for data_setup, data_hold and address_setup in FCB member m_NANDTiming.

The GPMI timing1.DEVICE_BUSY_TIMEOUT is set to value specified in FCB member TMTiming1_BusyTimeout.

The GPMI timing2 register values are set using FCB members TMTiming2.READ_LATENCY, CE_DELAY, PREAMBLE_DELAY, POSTAMBLE_DELAY, CMDADD_PAUSE and DATA_PAUSE

6.5.2.7 Typical NAND Page Organization

6.5.2.7.1 BCH ECC Page Organization

The first data block is called block 0 and the rest of the blocks are called block N. Separate ECC levelscan be used for block 0 and block N. The metadata bytes should be located at the beginning of a page, starting at byte 0, followed by data block 0, followed by ECC bytes for data block 0, followed by block 1 and its ECC bytes, and so on until N data blocks. The ECC level for block 0 can be different from the ECC level of rest of the blocks.

For NAND boot, with page size restrictions and data block size restricted to 512 bytes, only few combinations of ECC for block 0 and block N are possible.

Figure 6-6 shows the valid layout for 2112 byte sized page.

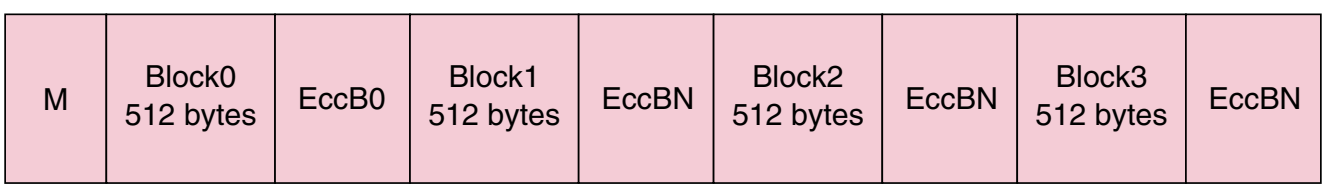


Figure 6-6. Valid Layout for 2112 bytes Sized Page

The number of ECC bits required for a data block is calculated using (ECC_Correction_Level * 13) bits.

In the above layout the ECC size for EccB0 and EccBN should be selected to not exceed a total page size of 2112 bytes. EccB0 and EccBN can be one of 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 bits ECC correction level. The total bytes would then be:

[M + (data_block_size x 4) + ([EccB0 + (EccBN x 3)] x 13) / 8] <= 2112;

M = metadata bytes and data_block_size is 512.

There are 4 data blocks of 512 bytes each in a page of 2k page sized NAND. The values of EccB0 and EccBN should be such that the above calculation would not result in a value greater than 2112 bytes.

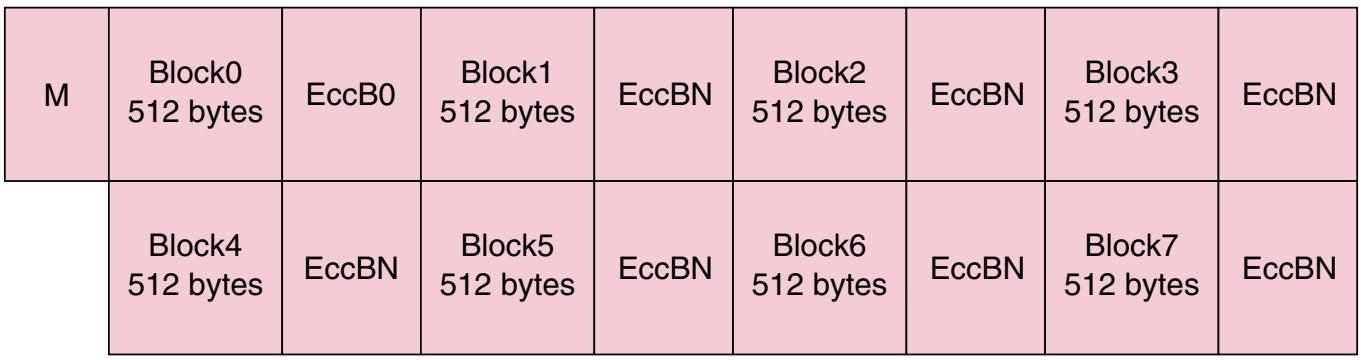


Figure 6-7. Valid Layout for 4 Kbytes Sized Page

Different NAND manufacturers have different sizes for a 4K page, 4314 bytes is typical.

[M + (data_block_size x 8) + ([EccB0 + (EccBN x 7)] x 13) / 8] <= 4314;

M=metadata bytes and data_block_size is 512.

There are 8 data blocks of 512 bytes each in a page of a 4k page sized NAND. The values of EccB0 and EccBN should be such that above calculation should not result in a value greater than the size of a page in a 4k page NAND.

6.5.2.7.2 Metadata

The number of bytes used for metadata are specified in FCB. Metadata for BCH encoded pages will be placed at the beginning of a page. ROM only cares about the first byte of metadata to swap it with bad block marker byte in page data after each page read. It is therefore important to have at least one byte for metadata bytes field in FCB data structure.

6.5.2.8 IOMUX Configuration for NAND Flash

The following table shows the RawNand IOMUX pin configuration.

Table 6-13. RawNand IOMUX Pin Configuration

	KPP	EIM	ELCDIF
CLE	KEY_COL0	EIM_DA8	DISP_D8
ALE	KEY_ROW0	EIM_DA9	DISP_D9

Table continues on the next page...

Table 6-13. RawNand IOMUX Pin Configuration (continued)

	KPP	EIM	ELCDIF
CEN0	KEY_COL1	EIM_DA10	DISP_D10
READY0	KEY_COL3	EIM_DA14	DISP_D14
DQS	KEY)ROW3	EIM_DA15	DISP_D15
WRN	SD3_CMD		
RDN	SD3_CLK		
D[4]	SD3_D0		
D[5]	SD3_D1		
D[6]	SD3_D2		
D[7]	SD3_D3		
D[0]	SD3_D4		
D[1]	SD3_D5		
D[2]	SD3_D6		
D[3]	SD3_D7		
RESET N	SD3_WP		

NOTE

Image header, DCD and boot data must reside in the initial load region of bootable image (4K). Image length must be specified in the boot data structure pointed to by the image header boot data element, please see Figure 6-15 for image vector table structure

6.5.3 Expansion Device Support

The i.MX50 ROM supports to boot from MMC/eMMC and SD/eSD compliant devices. The SD/MMC/eSD/eMMC boot can be performed at either eSDHC-1, eSDHC-2, eSDHC-3, or eSDHC-4 ports, based on setting of the BOOT_CFG3[5:4] (Port Select) fuse or its associated GPIO input value at boot. For eMMC4.3 and eMMC4.4 with BOOT ACK support boot mode, only eSDHC-3 can be used. Refer to Table 6-14 for details. The boot flow for expansion devices is shown in Figure 6-8.

Table 6-14. ESDHC Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Settings
BOOT_CFG1[7:6]	OEM	Boot Device Selection	Yes	01 - Boot from ESDHC Interfaces
BOOT_CFG1[5]	OEM	SD/MMC Selection	Yes	0 - SD/eSD 1 - MMC/eMMC

Table continues on the next page...

Table 6-14. ESDHC Boot eFUSE Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Settings
BOOT_CFG1[4]	OEM	Fast Boot Support	Yes	0 - Normal Boot 1 - Fast Boot
BOOT_CFG1[3]	OEM	SD/MMC Speed Mode	Yes	0 - High Speed Mode (SCLK@40 MHz) 1 - Normal Speed Mode(SCLK@20 MHz)
BOOT_CFG2[7:5]	OEM	Bus Width	Yes	SD/eSD (BOOT_CFG1[5]=0) xx0 - 4-bit xx1 - 1-bit MMC/eMMC (BOOT_CFG1[5]=1) 000 - 4-bit 001 - 1-bit 010 - 8-bit 100 - 4-bit DDR (MMC 4.4) 110 - 8-bit DDR (MMC 4.4) Else - reserved.
BOOT_CFG3[5:4]	OEM	Port Select	Yes	00 - eSDHC1 01 - eSDHC2 10 - eSDHC3 11 - eSDHC4
CFG6[3]	OEM	DISABLE_MMC_DLL_DLY	No	0 - Boot ROM default. 1 - Apply value per fuse field MMC_DLL_DLY[3:0]
CFG6[23:18]	OEM	MMC_DLL_DLY	No	DLL Delay cycle number

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 6-3 for corresponding GPIO pin.

Boot code supports following standards.

- MMCv4.4 or less
- eMMCv4.4 or less
- SDv2.0 or less
- eSDv2.10 rev-0.9, with or without FAST_BOOT.

MMC/SD/eSD/eMMC can be connected to any of eSDHC1-4 module and booting can be done by copying 2 Kbyte of data from MMC/SD/eSD/eMMC device to internal RAM. After checking the Image Vector Table header value (0xD1) from boot image, the ROM code performs a DCD check. After successful DCD extraction, the Rom code extracts the destination pointer and length of image to be copied to RAM device from where code execution occurs.

Table 6-15. SD/MMC Frequencies

Frequency	AXI/DDR Freq=0	AXI/DDR Freq=1
Identification (KHz)	357.143	297.321
Normal Speed Mode (MHz)	20	16.6
High Speed Mode (MHz)	40	33.3

NOTE

Image header, DCD and boot data must reside in the initial load region of bootable image (4K). Image length must be specified in the boot data structure pointed to by the image header boot data element, please see Figure 6-15 for image vector table structure

6.5.3.1 IOMUX Configuration of eSDHC

eSDHC2 and eSDHC3 modules have a full set of dedicated signals. eSDHC1 only supports 4 bits in default; the higher 4 bits are muxed on UART signals. eSDHC4 singlas are totally muxed on UART signals or DISP signals. The lower 4 bits of eSDHC4 data signals have two different mux modes. CFG6.bit[10] is used to select the mux mode.

Table 6-16. IOMUX Configuration Table of eSDHC

	eSDHC1	eSDHC2	eSDHC3/uSDHC	eSDHC4	
				Group 0	Group 1 (enabled by fuse CFG6[10])
CLK	SD1_CLK.ALT0	SD2_CLK.ALT0	SD3_CLK.ALT0	UART2_RTS.ALT4	DISP_D9.ALT4
CMD	SD1_CMD.ALT0	SD2_CMD.ALT0	SD3_CMD.ALT0	UART2_CTS.ALT4	DISP_D8.ALT4
DATA0	SD1_DO.ALT0	SD2_D0.ALT0	SD3_D0.ALT0	UART3_TXD.ALT4	DISP_D10.ALT4
DATA1	SD1_D1.ALT0	SD2_D1.ALT0	SD3_D1.ALT0	UART3_RXD.ALT4	DISP_D11.ALT4
DATA2	SD1_D2.ALT0	SD2_D2.ALT0	SD3_D2.ALT0	UART4_TXD.ALT4	DISP_D12.ALT4
DATA3	SD1_D3.ALT0	SD2_D3.ALT0	SD3_D3.ALT0	UART4_RXD.ALT4	DISP_D13.ALT4
DATA4	UART3_TXD.ALT3	SD2_D4.ALT0	SD3_D4.ALT0	UART1_CTS.ALT4	
DATA5	UART3_RXD.ALT3	SD2_D5.ALT0	SD3_D5.ALT0	UART1_RTS.ALT4	
DATA6	UART4_TXD.ALT3	SD2_D6.ALT0	SD3_D6.ALT0	UART2_TXD.ALT4	
DATA7	UART4_RXD.ALT3	SD2_D7.ALT0	SD3_D7.ALT0	UART2_RXD.ALT4	

6.5.3.2 MMC and eMMC Boot

During initialization (normal boot mode) the MMC frequency is set to 357.143 KHz. When the MMC card enters the identification portion of the initialization, voltage validation is performed and the ROM boot code checks high voltage settings and card capacity. The ROM boot code supports both high capacity and low capacity MMC/ eMMC cards. After initialization phase is complete, the ROM boot code switches to a higher frequency (20 MHz in Normal boot mode or 40 MHz in High Speed mode). eMMC is also interfaced through eSDHC and follows same flow as does by MMC.

The boot partition can be selected for an MMC4.x card after the card initialization is complete. The ROM code reads the BOOT_PARTITION_ENABLE field in the Ext_CSD[179] to get the boot partition to be set. If there is no boot partition mentioned in BOOT_PARTITION_ENABLE field or the user partition has been mentioned, ROM boots from the user partition.

If using an eMMC4.3 or eMMC4.4 device supporting special "Boot mode", it can be initiated by pulling the CMD line low. If BOOT ACK is enabled, the eMMC4.3/ eMMC4.4 device sends the BOOT ACK via DATA lines and ROM can read the BOOT ACK [S010E] to identify the eMMC4.3/eMMC4.4 device. eMMC4.3/eMMC4.4 device with "Boot mode" feature can only be supported via eSDHC-3 and with BOOT ACK enabled. ROM waits for 50 ms to get the BOOT ACK and if BOOT ACK is received by ROM, then only eMMC4.3/eMMC4.4 is booted in "Boot mode", otherwise eMMC4.3/ eMMC4.4 boots as a normal MMC card from the selected boot partition. This boot mode can be selected by BOOT_CFG1[4] (Fast Boot) fuse.

If using eMMC4.4 device, Double Data Rate (DDR) mode can be used. This mode can be selected by BOOT_CFG2[7:5] (Bus Width) fuse.

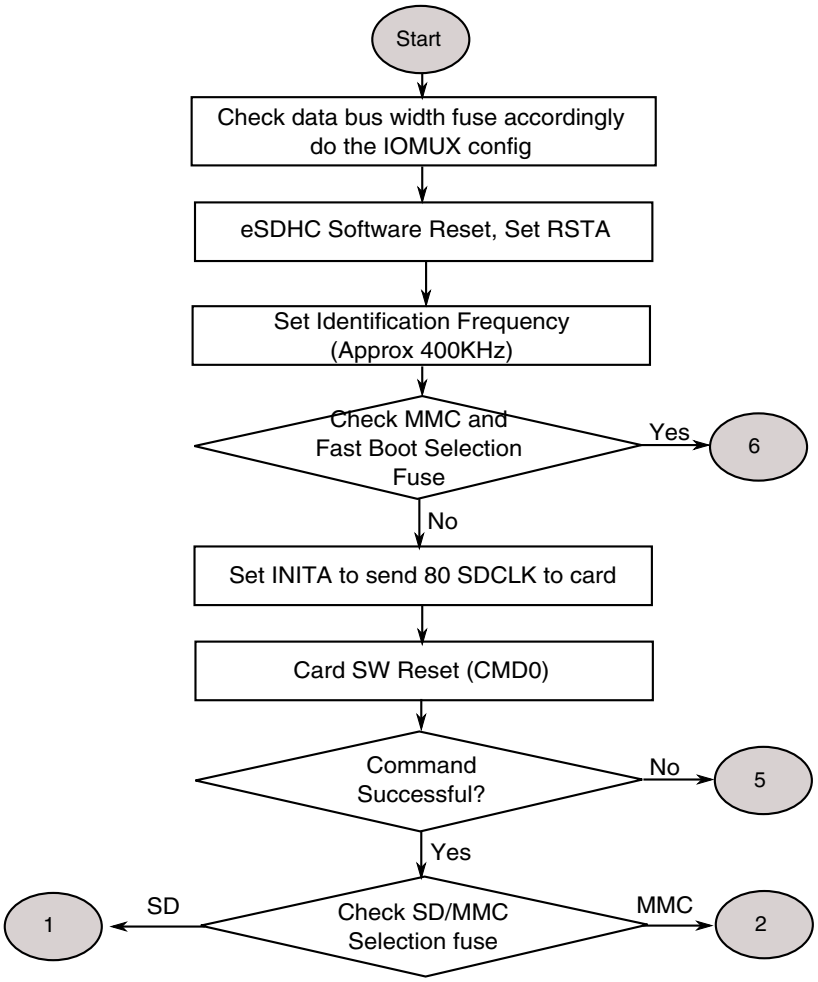
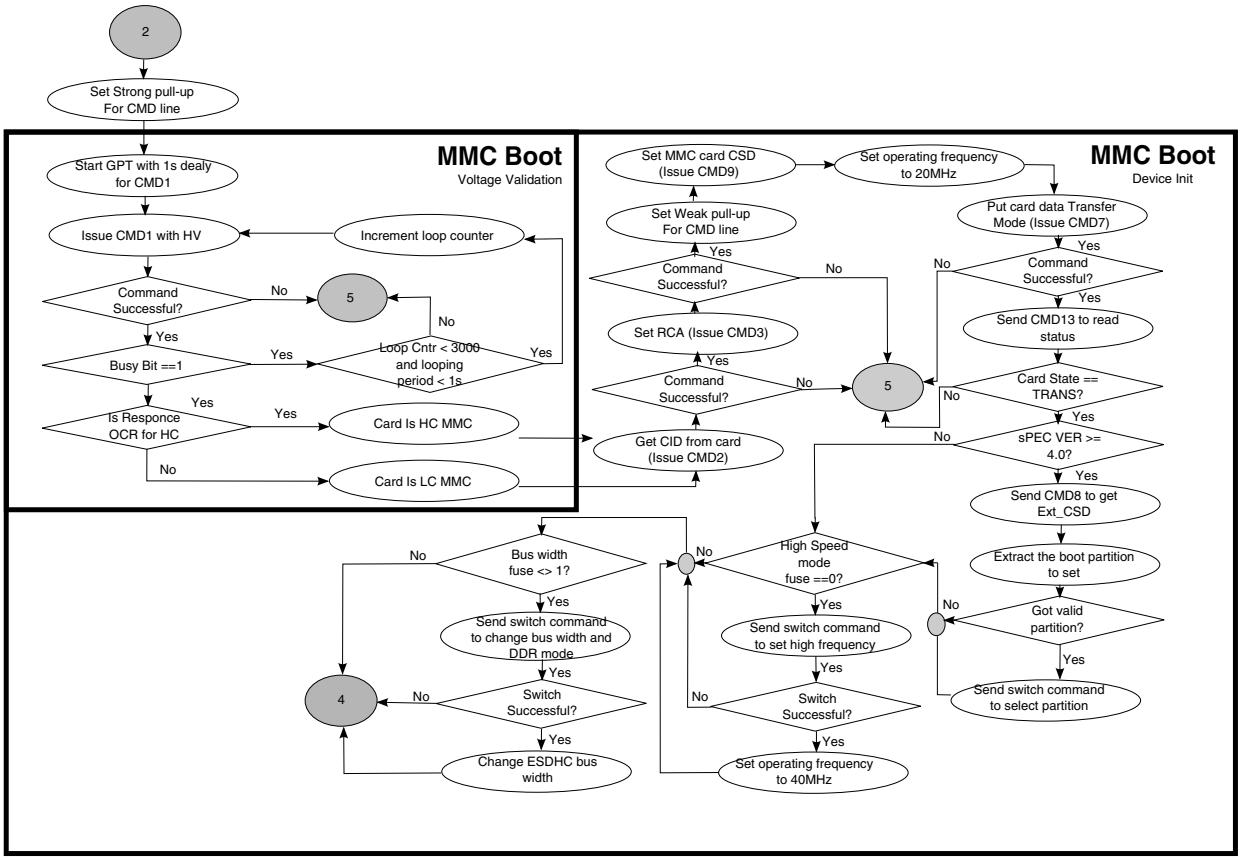


Figure 6-8. Expansion Device Boot Flow (1 of 5)



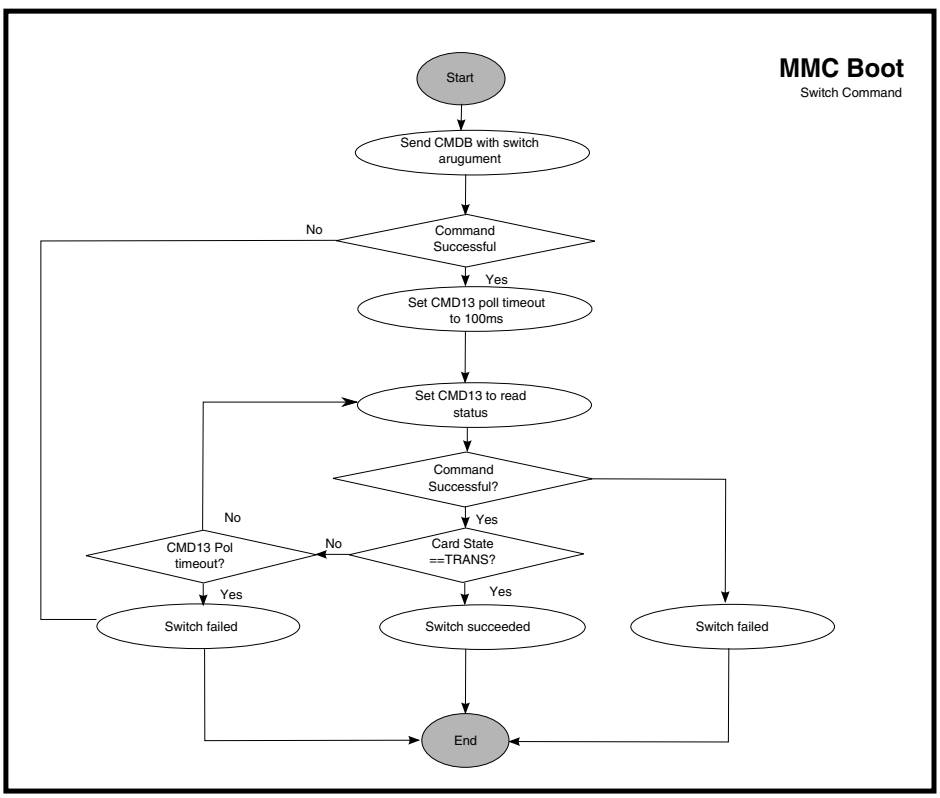


Figure 6-9. Expansion Device (MMC) Boot Flow (2 of 5)

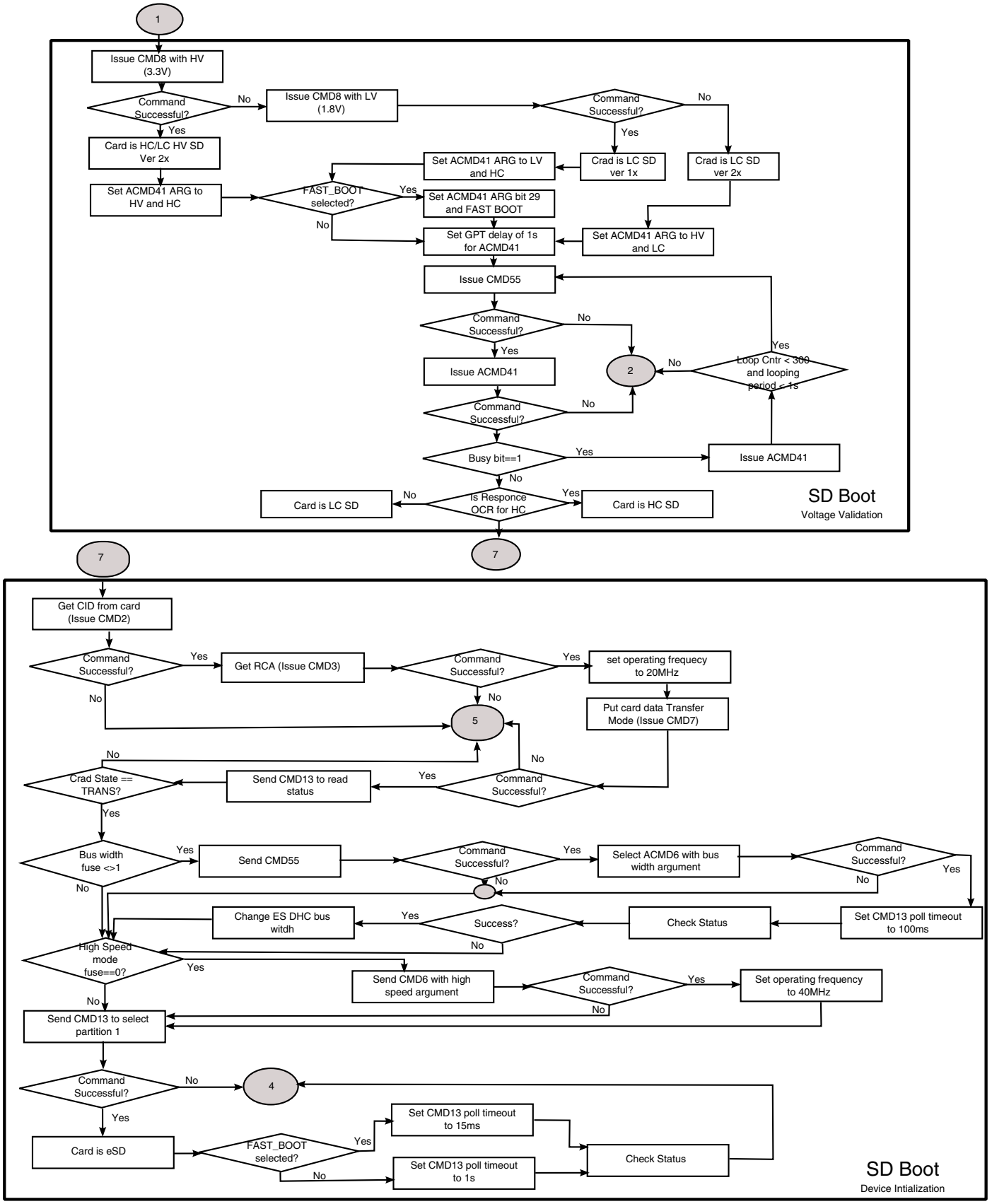


Figure 6-10. Expansion Device (SD/eSD) Boot Flow (3 of 5)

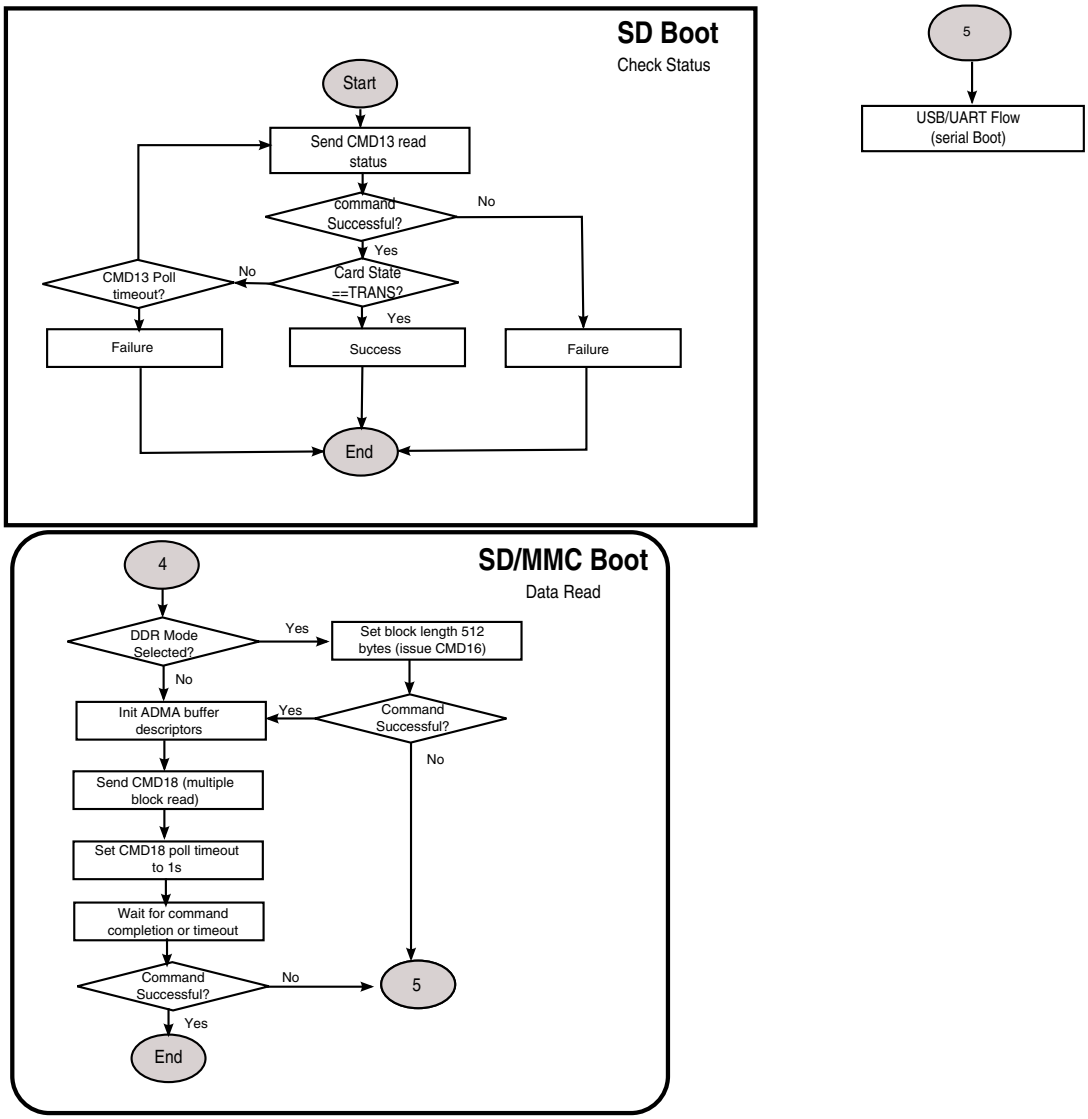


Figure 6-11. Expansion Device Boot Flow (4 of 5)

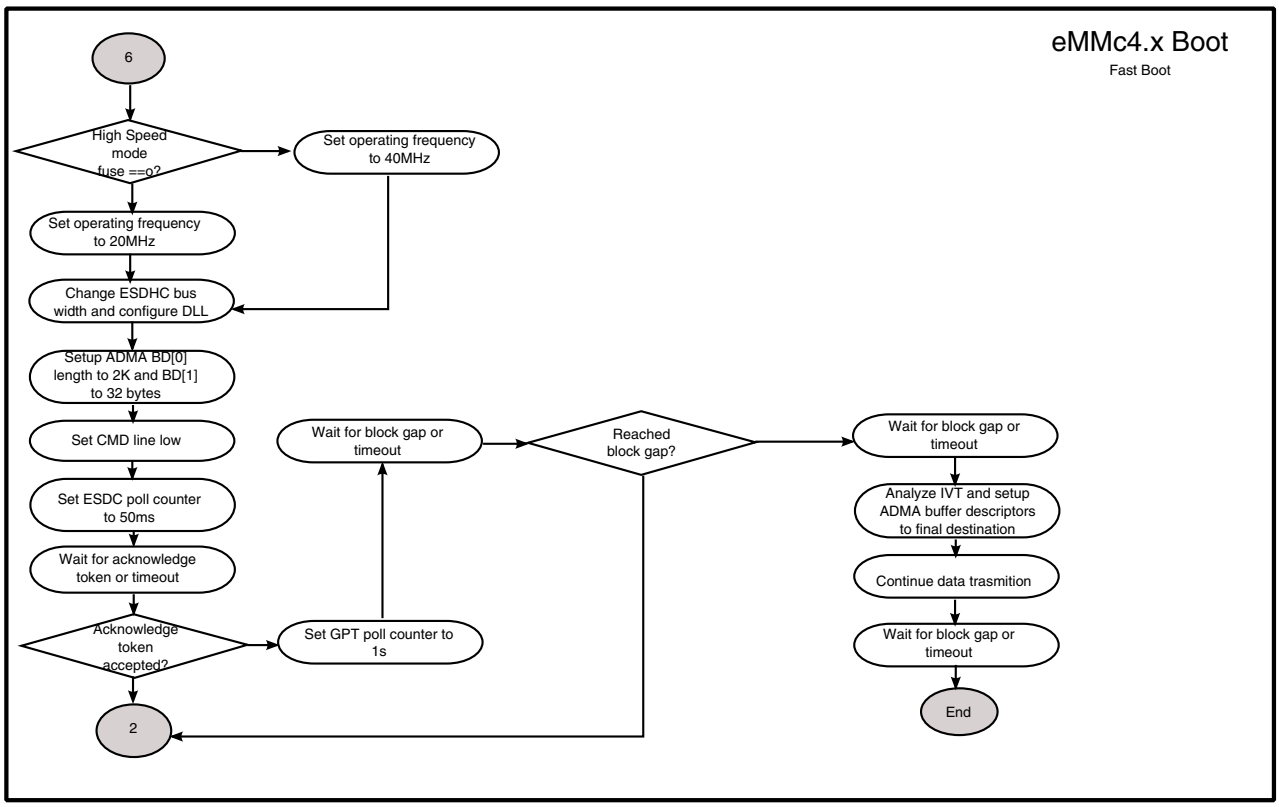


Figure 6-12. Expansion Device Boot Flow (5 of 5)

6.5.4 SD and eSD

After the normal boot mode initialization begins, the SD and eSD frequency is set to 357.143KHz. During the identification phase SD and eSD card voltage validation is performed. During voltage validation, boot code first checks with high voltage settings and if it fails it checks with low voltage settings. Capacity of card is also checked. Boot code supports high capacity and low capacity SD and eSD cards. After voltage validation card initialization is done.

During card initialization the ROM boot code attempts to set the boot partition (both SD and eSD devices). If this fails, the boot code assumes card is a normal SD card otherwise as eSD card. After initialisation phase is over, boot code switches to a higher frequency (20 MHz in Normal Speed mode or 40MHz in High Speed Mode). ROM also supports FAST_BOOT mode booting from eSD card. This mode can be selected by BOOT_CFG1[4] (Fast Boot) fuse described in Table 6-14.

NOTE

Image header, DCD and boot data must reside in the initial load region of bootable image (4K). Image length must be specified

in the boot data structure pointed to by the image header boot data element, see Figure 6-15 for image vector table structure

The table Table 6-17 list values for internal pull-ups on SD pads configured by ROM.

Table 6-17. ROM pull-up values for SD pads

SD1, SD2, SD3, SD4	SD boot modes	MMC boot modes	eMMC boot modes (SD3 only)
CMD	100K PU	22K PU11	100K PU
CLK	47K PU	47K PU	47K PU
DATA[7:0]	47K PU	47K PU	47K PU

6.5.5 Serial ROM Support via SPI and I²C

The i.MX50 supports boot from serial memory devices, such as EEPROM, and Serial Flash using the SPI (CSPI, eCSPI1, eCSPI2), and I²C (I²C1, I²C2 and I²C3) interfaces.

The boot ROM code determines the type of device using the following parameters, either provided by OCOTP bit settings or sampled on the I/O pinsat reset (See Table 6-18 for details).

Table 6-18. Serial ROM Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Settings
BOOT_CFG1[7:4]	OEM	Boot Device Selection	Yes	0011 - Boot from Serial ROM
BOOT_CFG1[3]	OEM	Serial ROM select	Yes	0 - I2C 1- SPI
BOOT_CFG2[5]	OEM	SPI Addressing	Yes	0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)
BOOT_CFG3[5:4]	OEM	Port Select	Yes	00 - I2C1 / eCSPI1 01 - I2C2 / eCSPI2 10 - I2C3 / CSPI 11 - Reserved
BOOT_CFG3[3:2]	OEM	CS select (SPI only)	Yes	00 - CS#0 01 - CS#1 10 - CS#2 11 - CS#3

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 6-3 for corresponding GPIO pin.

The I2C-1/I2C-2/I2C-3 module can be used as boot device when using I²C interface serial ROM boot. The I2C interface is configured to operate at 384 Kbps.

6.5.5.1 I²C EEPROM Boot

The boot flow when booting from an I²C EEPROM is shown in Figure 6-13. The boot ROM code reads the fuses BOOT_CFG1[7:4] (Boot Device Selection) and BOOT_CFG1[3] (Serial ROM select) to detect EEPROM device type. The ROM program copies 4K data from the EEPROM device to internal RAM. The boot ROM code next copies the initial 2 Kbyte of data as well as rest of image directly to application destination extracted from application image.

The i.MX50 uses the Device Select Code/Device Address in Table 6-19 to boot from an EEPROM

Table 6-19. EEPROM via I²C Device Select Code

Bits	Device Type Identifier				Chip Enable Address[1]			R/W
	7	6	5	4	3	2	1	0
Device Select Code	1	0	1	0	0	0	0	R/W

1. These address bits, should be configured at the memory device, to match this '000' value.

Table 6-20. IOMUX Configuration of I²C Modules

	I2C1	I2C2	I2C3
SCL	I2C1_SCL_ALT0	I2C2_SCL.ALT0	I2C2_SCL.ALT0
SDA	I2C1_SDA_ALT0	I2C2_SDA.ALT0	I2C3_SDA.ALT0

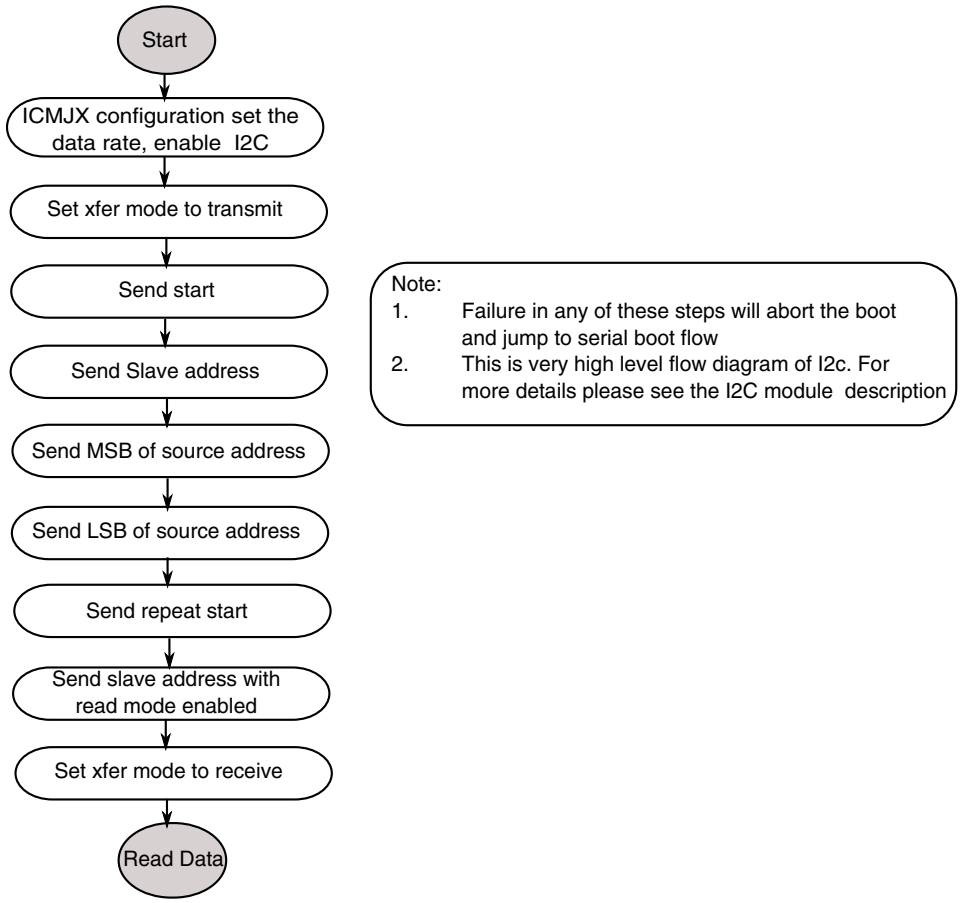


Figure 6-13. I²C Flow Chart

6.5.5.2 SPI EEPROM/Flash Boot

The CSPI/eCSPI1/eCSPI2 interface is configured in Master mode and the SPI memory device is connected to CSPI/eCSPI interfaces as slave. The boot ROM code copies 2 Kbyte data from SPImemory device to the internal RAM. If DCD verification is successful, the ROM code copies the initial 2 Kbyte data as well as the rest of image directly to application destination extracted from application image. The CSPI/ eCSPImaster can read data from theSPI memory device using 2 or 3 byte addressing and its burst length is 32 bytes.

NOTE

The Serial ROM Chip Select Number is determined by BOOT_CFG3[3:2] (Chip Select) fuse.

When using the SPI as boot device, the i.MX50 supports booting from both Serial EEPROMS and Serial Flash devices. The boot code determines which device is being used by reading the appropriate OCOTP bit or GPIO values at boot (See Table 6-18 for details).

Table 6-21. IOMUX Configuration of CSPI and eCSPI

	CSPI	eCSPI1	eCSPI2
SCLK	Cspi.SCLK.Alt0	Ecspi1.SCLK.Alt0	Ecspi2.SCLK.Alt0
MOSI	Cspi.MOSI.Alt0	Ecspi1.MOSI.Alt0	Ecspi2.MOSI.Alt0
MISO	Cspi.MISO.Alt0	Ecspi1.MISO.Alt0	Ecspi2.MISO.Alt0
SS0	Cspi.SS0.Alt0	Ecspi1.SS0.Alt0	Ecspi2.SS0.Alt0
RDY	Ecspi1.SCLK.Alt2	Ecspi2.SCLK.Alt3	Ecspi1.SCLK.Alt3
SS1	Ecspi1.MOSI.Alt2	Ecspi2.MOSI.Alt3	Ecspi1.MOSI.Alt3
SS2	Ecspi1.MISO.Alt2	Ecspi2.MISO.Alt3	Ecspi1.MISO.Alt3
SS3	Ecspi1.SS0.Alt2	Ecspi2.SS0.Alt3	Ecspi1.SS0.Alt3

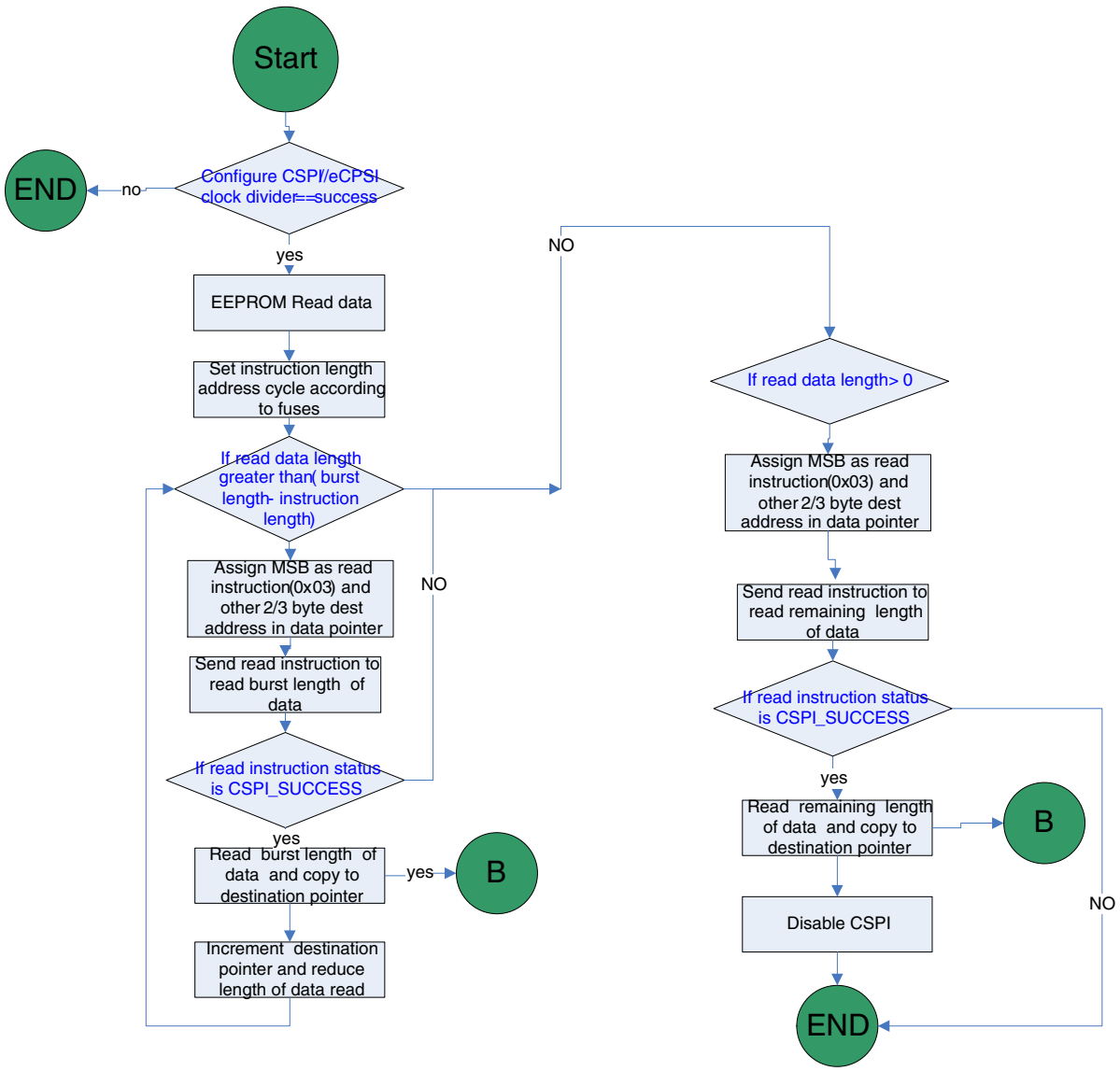


Figure 6-14. CSPI/eCSPI Boot Flow

6.6 Program Image

This section describes the data structures that are required to be included in a user's Program Image. A Program Image consists of:

- Image Vector Table-A list of pointers located at a fixed address that the ROM searches for to determine where other components of the Program Image are located.

- Boot Data-An additional header indicating the Image Location, Program Image size in bytes and plugin flag. It must be located in initial program image part (see Figure 6-1).
- Device Configuration Data-IC configuration data. It must be located in initial program image part (see Figure 6-1).
- Data used by HAB-CSF command data, signatures and certificates
- User code and data

6.6.1 Image Vector Table and Boot Data

The Image Vector Table (IVT) is the data structure that the ROM reads from the boot device supplying the external image containing the required data components to perform a successful boot. The IVT includes the image entry point, a pointer to Device Configuration Data (DCD) and other pointers used by the ROM during the boot process. The ROM locates the IVT at a fixed address that is determined by the boot device connected to i.MX50. The IVT offset from the base address and initial program image size for each boot device type is defined in Table 6-22. The location of the IVT is the only fixed requirement by the ROM. The remainder or the image memory map is flexible and is determined by the contents of the IVT.

Table 6-22. Image Vector Table Offset and Initial Image Size

Flash Type	Base Address Offset	Initial Image Size	DCD Max Size Limitation
NOR	4 Kbyte = 0x1000 bytes	Entire Image Size	1768 bytes
NAND 2KB Page Size	1 Kbyte = 0x400 bytes	2 Kbyte	1024 bytes
NAND 4KB Page Size	1 Kbyte = 0x400 bytes	4 Kbytes	1768 bytes
OneNAND	256 bytes = 0x100 bytes	1 Kbyte	736 bytes
SD/MMC/eSD/eMMC	1 Kbyte = 0x400 bytes	2 Kbyte	1024 bytes
I2C/HS-I2C/SPI EEPROM	1 Kbyte = 0x400 bytes	2 Kbyte	1024 bytes

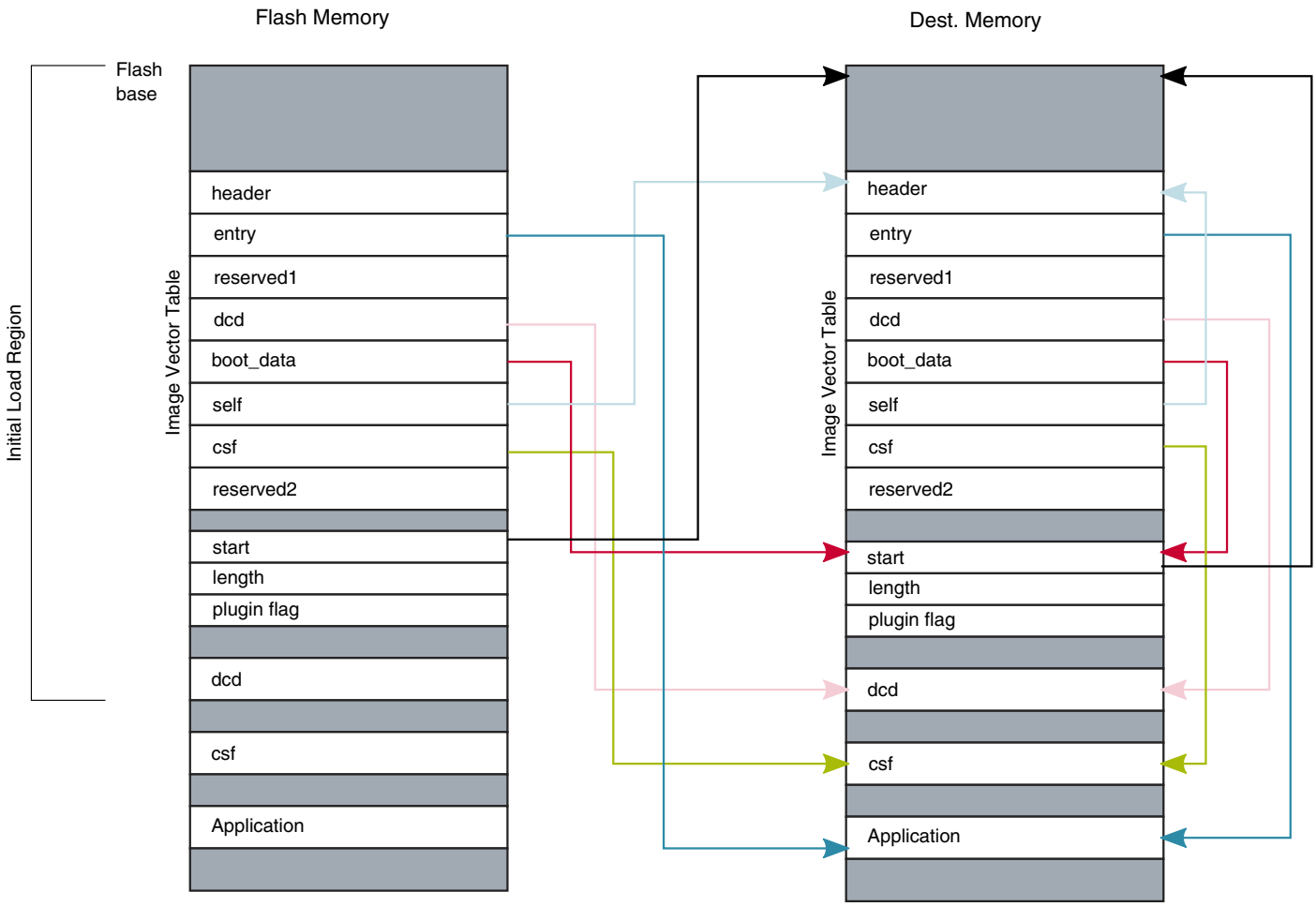


Figure 6-15. Image Vector Table Data Structure

6.6.1.1 Image Vector Table Structure

The IVT has the following format where each entry is a 32-bit word:

Table 6-23. i.MX50 IVT Format

header
entry
reserved1
dcd
boot data
self
csf
reserved2

- header: The IVT header has the following format:

Table 6-24. IVT Header Format

Tag	Length	version

where:

```
Tag: A single byte field set to 0xD1
Length: a two byte field in big endian format containing the overall length of the IVT,
in bytes, including the header. (the length is fixed and must have a value of
32 bytes)
Version: A single byte field set to 0x40
```

- entry: Absolute address of the first instruction to execute from the image.
- reserved1: Reserved and should be NULL.
- dcd: Absolute address of the image DCD. The DCD is optional so this field may be set to NULL if no DCD is required. See Device Configuration Data (DCD) for further details on DCD.
- boot data: Absolute address of the Boot Data, see Boot Data Structure.
- self: Absolute address of the IVT. Used internally by the ROM.
- csf: Absolute address of Command Sequence File (CSF) used by the HAB library. See High Assurance Boot (HAB) for details on secure boot using HAB. This field must be set to NULL when not performing a secure boot.
- reserved2: Reserved and should be NULL.

6.6.1.2 Boot Data Structure

The Boot Data must follow the format defined in Table 6-25. This data structure is pointed to by boot data element in IVT structure.

Table 6-25. i i.MX50 Boot Data Format

start
size
plugin

Where:

- start: Absolute address of the image
- size: Size of the image
- plugin: Plugin flag

6.6.2 Device Configuration Data (DCD)

Upon reset the i.MX50 uses the default register values for all peripherals in the system. The i.MX50 ROM changes some of these defaults such as clock frequencies for boot purposes. However, these settings typically are not ideal for achieving optimal system performance and there are even some peripherals that must configured before they can be used. The DCD is configuration information contained in a Program Image, external to the ROM, that the ROM interprets to configure various peripherals on i.MX50. Some components such as SDRAM require some sequence of register programming as part of configuration before it is ready to be used. The DCD feature can be used to program the EIM registers and DRAM MC registers to the optimal settings.

The ROM determines the location of the DCD table based on information located in the Image Vector Table (IVT). See Image Vector Table and Boot Data for more details. The DCD table shown in Table 6-26 is a big endian byte array of the allowable DCD commands. The maximum size of the DCD limited to 1768 bytes by the HAB. However, for some boot devices this is further restricted as indicated by Table 6-22.

Table 6-26. DCD Data format

Header
[CMD]
[CMD]
...

The DCD header is 4 bytes with the following format:

Table 6-27. DCD Header

Tag	Length	Version

where:

```
Tag: A single byte field set to 0xD2
Length: a two byte field in big endian format containing the overall length of the DCD,
in bytes, including the header
Version: A single byte field set to 0x40
```

DCD Table Structure

```
#define HWC_LINES 63
#define HWC_WORDS (HWC_LINES*2+2)
#define HWC_VALUE (HWC_LINES*3)
#define HWC_BYTES (HWC_VALUE*4)
#define LE_2_BE_32(word) \
    ( ((word&0x000000FF) << 24)|        \
      ((word&0x0000FF00) << 8) |       \
      ((word&0x00FF0000) >> 8) |       \
      ((word&0xFF000000) >> 24)  )
#define LE_2_BE_16(word) \
    ( ((word&0x00FF) << 8) |           \
      ((word&0xFF00) >> 8) )
```

```
typedef unsigned int UINT32;
const unsigned int hwc[HWC_WORDS] = {
    (UINT32)0xD2<<0 |
    LE_2_BE_16(HWC_WORDS*4)<<8 |
    (UINT32)4<<28 |
    (UINT32)0<<24,
    (UINT32)0xCC<<0 | LE_2_BE_16(HWC_WORDS*4-4)<<8 | 4<<24,
    /* DDR2 IOMUX configuration */
    LE_2_BE_32(0x53fa8554), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM3 */
    LE_2_BE_32(0x53fa8558), LE_2_BE_32(0x00380040),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS3 */
    LE_2_BE_32(0x53fa8560), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM2 */
    LE_2_BE_32(0x53fa8564), LE_2_BE_32(0x00380040),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_SDODT1 */
    LE_2_BE_32(0x53fa8568), LE_2_BE_32(0x00380040),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS2 */
    LE_2_BE_32(0x53fa8570), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK_1 */
    LE_2_BE_32(0x53fa8574), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_CAS */
    LE_2_BE_32(0x53fa8578), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK_0 */
    LE_2_BE_32(0x53fa857c), LE_2_BE_32(0x00380040),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS0 */
    LE_2_BE_32(0x53fa8580), LE_2_BE_32(0x00380040),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_SDODT0 */
    LE_2_BE_32(0x53fa8584), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM0 */
    LE_2_BE_32(0x53fa8588), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_RAS */
    LE_2_BE_32(0x53fa8590), LE_2_BE_32(0x00380040),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS1 */
    LE_2_BE_32(0x53fa8594), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM1 */
    LE_2_BE_32(0x53fa86f0), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_GRP_ADDDS */
    LE_2_BE_32(0x53fa86f4), LE_2_BE_32(0x00000200),  /* IOMUXC_SW_PAD_CTL_GRP_DDRMODE_CTL */
    LE_2_BE_32(0x53fa86fc), LE_2_BE_32(0x00000000),  /* IOMUXC_SW_PAD_CTL_GRP_DDRPKE */
    LE_2_BE_32(0x53fa8714), LE_2_BE_32(0x00000000),  /* IOMUXC_SW_PAD_CTL_GRP_DDRMODE - CMOS
mode */
    LE_2_BE_32(0x53fa8718), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_GRP_B0DS */
    LE_2_BE_32(0x53fa871c), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_GRP_B1DS */
    LE_2_BE_32(0x53fa8720), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_GRP_CTLDS */
    LE_2_BE_32(0x53fa8724), LE_2_BE_32(0x02000000),  /* IOMUXC_SW_PAD_CTL_GRP_DDR_TYPE -
DDR_SEL0=  */
    LE_2_BE_32(0x53fa8728), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_GRP_B2DS */
    LE_2_BE_32(0x53fa872c), LE_2_BE_32(0x00380000),  /* IOMUXC_SW_PAD_CTL_GRP_B3DS */

    /* Initialize DDR2 memory */
    LE_2_BE_32(0x14000088), LE_2_BE_32(0x2d313331), /* DATABAHN */
    LE_2_BE_32(0x14000090), LE_2_BE_32(0x393b3836), /* DATABAHN */
    LE_2_BE_32(0x140000F8), LE_2_BE_32(0x00000800), /* DATABAHN */
    LE_2_BE_32(0x1400007c), LE_2_BE_32(0x020c0211), /* DATABAHN */
    LE_2_BE_32(0x14000080), LE_2_BE_32(0x014c0155), /* DATABAHN */
    LE_2_BE_32(0x14000018), LE_2_BE_32(0x000016d0), /* DATABAHN */
    LE_2_BE_32(0x14000000), LE_2_BE_32(0xc4110000), /* DATABAHN */
    LE_2_BE_32(0x1400000C), LE_2_BE_32(0x4d5122d2), /* DATABAHN */
    LE_2_BE_32(0x14000010), LE_2_BE_32(0x92d18a22), /* DATABAHN */
    LE_2_BE_32(0x14000014), LE_2_BE_32(0x00c70092), /* DATABAHN */
    LE_2_BE_32(0x1400002c), LE_2_BE_32(0x000026d2), /* DATABAHN */
    LE_2_BE_32(0x14000030), LE_2_BE_32(0x009f000e), /* DATABAHN */
    LE_2_BE_32(0x14000008), LE_2_BE_32(0x12272000), /* DATABAHN */
    LE_2_BE_32(0x14000004), LE_2_BE_32(0x00030012), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x04008010), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00008032), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00008033), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00008031), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x0b5280b0), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x04008010), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00008020), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00008020), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x0a528030), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x03c68031), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00468031), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x04008018), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x0000803a), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x0000803b), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00008039), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x0b528138), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x04008018), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00008028), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00008028), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x0a528038), /* DATABAHN */
```

```
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x03c68039), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00468039), /* DATABAHN */
    LE_2_BE_32(0x14000020), LE_2_BE_32(0x00005800), /* DATABAHN */
    LE_2_BE_32(0x14000058), LE_2_BE_32(0x00033337), /* DATABAHN */
    LE_2_BE_32(0x1400001c), LE_2_BE_32(0x00000000)  /* DATABAHN */
};
```

6.6.2.1 Write Data Command

The Write Data Command is used to write a list of given 1, 2 or 4-byte values or bitmasks to a corresponding list of target addresses. The format of Write Data Command, again a big endian byte array, is shown in Table 6-28.

Table 6-28. Write Data Command Format

Tag	Length	Parameter
Address		
Value/Mask		
[Address]		
[Value/Mask]		
...		
[Address]		
[Value/Mask]		

where:

```
Tag: A single byte field se to 0xCC
Length: A two byte field in big endian format containing the length of the Write Data
Command, in bytes, including the header
Address: target address to which data should be written
Value/Mask: data value or bitmask to be written to preceding address
```

The Parameter field is a single byte divided into bitfields as follows:

Table 6-29. Write Data Command Parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: width of target locations in bytes. Either 1, 2 or 4
flags: control flags for command behavior.
Data Mask = bit 3: if set, only specific bits may be overwritten at target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, bits at the target address overwritten with this flag
(otherwise it is ignored)
```

One or more target address and value/bitmask pairs can be specified. The same bytes and flags parameters apply to all locations in the command.

When successful, this command writes to each target address in accordance with the flags as follows:

Table 6-30. Interpretation of Write Data Command Flags

"Mask"	"Set"	Action	Interpretation
0	0	*address = val_msk	Write value
0	1	*address = val_msk	Write value
1	0	*address &= ~val_msk	Clear bitmask
1	1	*address |= val_msk	Set bitmask

Notes:

If any of the target addresses does not have the same alignment as the data width indicated in the parameter field, none of the values are written.

If any of the values is larger or any of the bitmasks is wider than permitted by the data width indicated in the parameter field, none of the values are written.

If any of the target addresses do not lie within an allowed region, none of the values are written. The list of allowable modules and target addresses fori.MX50 are given below.

Table 6-31. Valid DCD Address Ranges

Address range	Start address	Length (in bytes)
CCM register set	0x53FD4000	0x53FD7FFF
EIM register set	0x63FDA000	0x63FDAFFF
IOMUXC registers	0x53FA8000	0x53FABFFF
PLL1 registers	0x63F80000	0x63F83FFF
PLL2 registers	0x63F84000	0x63F87FFF
PLL3 registers	0x63F88000	0x63F8BFFF
DRAM MC registers	0x14000000	0x1400043b
DIGCTL registers	0x41004000	0x41005FFF

6.6.2.2 Check Data Command

The Check Data Command is used to test for a given 1, 2 or 4-byte bitmasks from a source address. The Check Data Command is a big endian byte array with format shown in Table 6-32.

Table 6-32. Check Data Command Format

Tag	Length	Parameter

Table continues on the next page...

Table 6-32. Check Data Command Format (continued)

Address
Mask
[Count]

where:

```
Tag: A single byte field se to 0xCF
Length: A two byte field in big endian format containing the length of the Check Data
Command, in bytes, including the header
Address: source address to test
Mask: bit mask to test
Count: optional poll count. If count is not specified this command will poll indefinitely
until the exit condition is met. If count = 0, this command behaves as for NOP.
```

The Parameter field is a single byte divided into bitfields as follows:

Table 6-33. Check Data Command Parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: width of target locations in bytes. Either 1, 2 or 4
flags: control flags for command behavior.
Data Mask = bit 3: if set, only specific bits may be overwritten at target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, bits at the target address overwritten with this flag
(otherwise it is ignored)
```

This command polls the source address until either the exit condition is satisfied, or the poll count is reached. The exit condition is determined by the flags as follows:

Table 6-34. Interpretation of Check Data Command Flags

"Mask"	"Set"	Action	Interpretation
0	0	(*address & mask) == 0	All bits clear
0	1	(*address & mask) == mask	All bits set
1	0	(*address & mask) != mask	Any bit clear
1	1	(*address & mask) != 0	Any bit set

Notes:

If the source address does not have the same alignment as the data width indicated in the parameter field, the value is not read.

If the bitmask is wider than permitted by the data width indicated in the parameter field, the value is not read.

6.6.2.3 NOP Command

This command has no effect. The format of Check Data Command is a little endian four byte array as shown in Table 6-35.

Table 6-35. NOP Command Format

Tag	Length	Undefined

where:

```
Tag: A single byte field se to 0xC0
Length: A two byte field containing the length of the NOP Command in bytes. Fixed to a
value of 4.
Undefined: HAB ignores this byte and can be set to any value.
```

6.7 Plugin Image

i.MX50 ROM supports limited number of boot modes. For using other devices as boot source (e.g. ethernet, CDROM, USB Stick and etc...) supported boot device must be used (typically serial ROM) for firmware with the missing boot drivers.

Boot ROM will detect the image type by plugin flag of boot data structure (see Boot Data Structure). If the plugin flag is 1, then ROM will use the image as plugin function. The function must initialize boot device and copy the image to final location. At the end, the plugin function must return with the parameters of application image.

The boot ROM will authenticate the plugin image prior running plugin function and then will authenticate the application image.

The plugin function must follow the API described below:

typedef BOOLEAN(*) plugin_download_f(void **start, size_t *bytes, UINT32 *ivt_offset)

ARGUMENTS PASSED:

- start - Image load address on exit.
- bytes - Image size on exit.
- ivt_offset - Offset in bytes of the IVT from the image start address on exit.

RETURN VALUE:

- TRUE - on success
- FALSE - on failure

6.8 USB Low-Power Boot

The i.MX50 supports a Low-Power Boot (LPB) mode where a production level device can attempt to boot with a depleted or disconnected battery. This mode is independent of the boot modes derived from the BOOT_MODE[1:0] pins and allows the i.MX50 to boot using USB as the only power supply. The i.MX50 ROM code involved in LPB is minimal, with most configuration operations performed by the Power Management IC (PMIC) and USB drivers included in a Program Image.

The platform current consumption during LPB ROM stage is targeted to be less than 100 mA at VUSB. For this purpose, DRAM module is disabled and the Program Image is downloaded to OCRAM.

Low power boot is supported by default, but it can be disabled by blowing the OCOTP bit BT_DISABLE_LPB, which is located in CFG4.Bit[4].

Table 6-36. OCOTP Bits for USB Low Power Boot

Fuse	Config	Definition	GPIO	Settings
BT_DISABLE_LPB	OEM	Low power boot mode disable or enable	No	0 - LPB Enabled 1 - LPB Disabled
LOW_BATT_GPIO_LEVEL	OEM	Determines whether the LOW_BATT_GPIO (GPIO6_16) is active high or active low.	No	0 - LOW_BATT_GPIO is Active High 1 - LOW_BATT_GPIO is Active Low
BT_LPB_FREQ	OEM	The ARM core frequency at Low power boot mode	Yes	00 - 192MHz at 192MHz PLL1 01 - 133MHz at 266MHz PLL1 10 - 55.3MHz at 166MHz PLL1 11 - 220MHz at 220MHz PLL1

The i.MX50 ROM code use GPIO6_16 as the LOW_BATT_GPIO to indicate the low battery state. The board designer should connect the i.MX50 GPIO6_16 to the interrupt output pad of the PMIC. After finding the low battery state, the ROM code will check the USB OTG PHY register for the existence of USB charger or Wall charger. If a charger is found, it means the system will get enough current, and the ROM code will continue to run on the normal boot flow. Otherwise, the ROM code will have to decrease the power consumption for the 100 mA limitation of USB host. For more details of USB low power boot, refer to *USB Battery Charging Specification*.

The polarity of the LOW_BATT_GPIO signal is determined by the LOW_BATT_GPIO_LEVEL, which is located in CFG6.Bit[9].

The frequency of Low power boot is selected by BT_LPB_FREQ[1:0] illustrated in Table 6-24.

6.8.1 LPB Mode Flow

The maximum LPB ARM core frequency is selected and stored in fuses BT_LPB_FREQ[2:0]. See **USB Low-Power Boot Mode (LPB)**, for details on setting frequency selection OCOTP bits for LPB. Even though the ROM boot code supports all primary boot devices in LPB mode in almost all cases USB are not be used in LPB mode for downloading a Program Image.

The LPB flowchart is shown in Figure 6-16 and has the following steps:

1. Check if the LPB feature is disabled by reading BT_DISABLE_LPB OCOTP bit.
2. If LPB feature is disabled, boot normally
3. If LPB feature is notdisabled, execute remaining steps
4. Check LOW_BATT_GPIO_LEVEL OCOTP bit. If LOW_BATT_GPIO_LEVEL is low, and GPIO6_16 is high, the AAPL-like PMIC reports low battery state; If LOW_BATT_GPIO_LEVEL is high, and GPIO6_16 is low, the non-AAPL-like PMIC reports low battery state.Otherwise,boot normally. Check if a charger is connected to the UTMI PHY of the i.MX50 USB OTG.
5. If a charger is connected, the system can get enough current, boot normally. If no charger is found, go to step 7
6. start to configure the clocks in low power boot mode. ARM core frequency will be decreased to a lower value based on BT_LPB_FREQ[1:0].
7. After configuring the clocks, the ROM code will start the similar flow with normal boot to load boot image from the boot media.

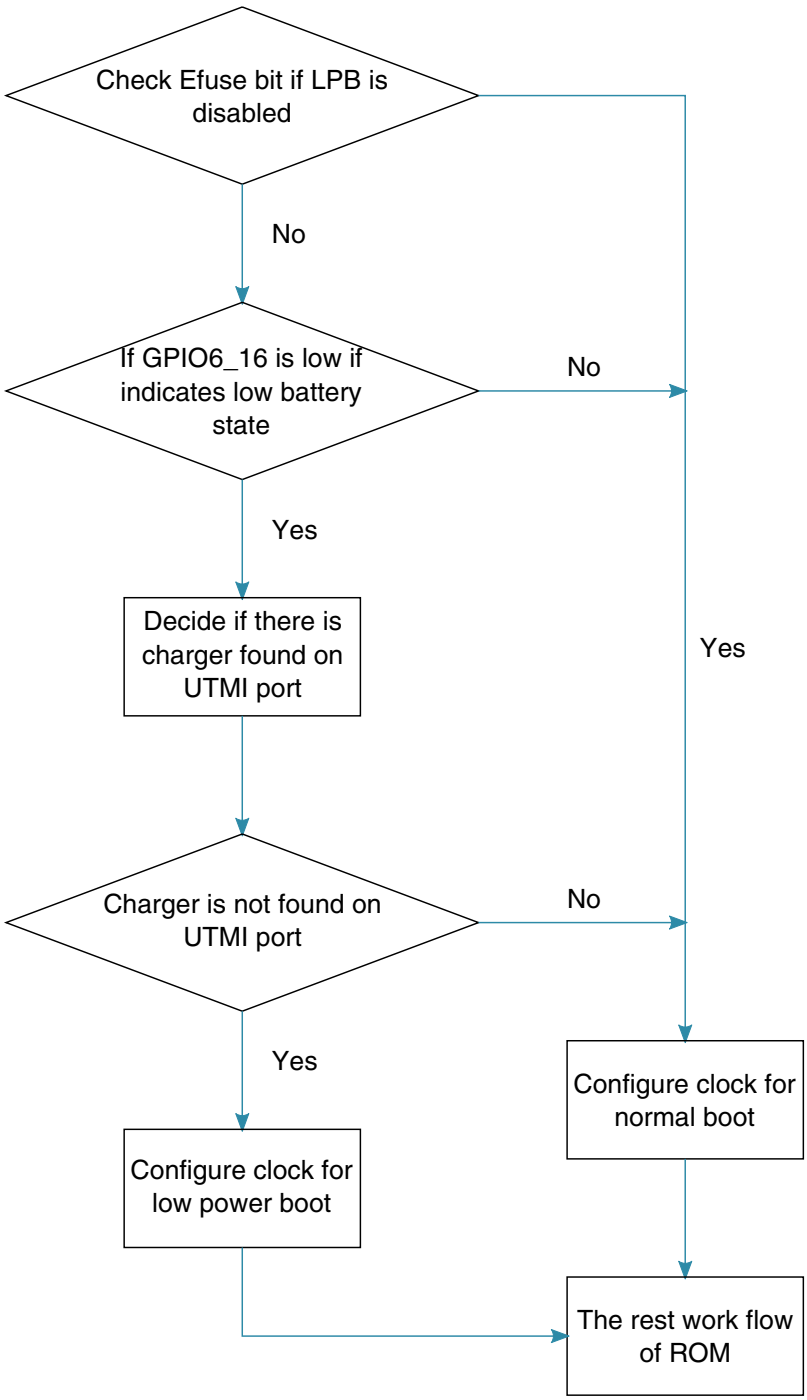


Figure 6-16. USB Low-Power Boot Flow

6.9 USB Downloader (BOOT_MODE[1:0] = 11)

The Serial Downloader provides a means to download a Program Image to the i.MX50 over a USB serial connection. In this mode if ENABLE_WDOG_TIMER_RESET fuse is blown then the ROM programs WDOG1 for a 32-second time-out and then continuously polls for USB activity. If no activity is found on USB and the watchdog timer expires the ARM core is reset. By default wdog timer reset is not enabled.

NOTE

If ENABLE_WDOG_TIMER_RESET fuse is blown then downloaded image must continue to service the watchdog timer to avoid an undesired reset from occurring.

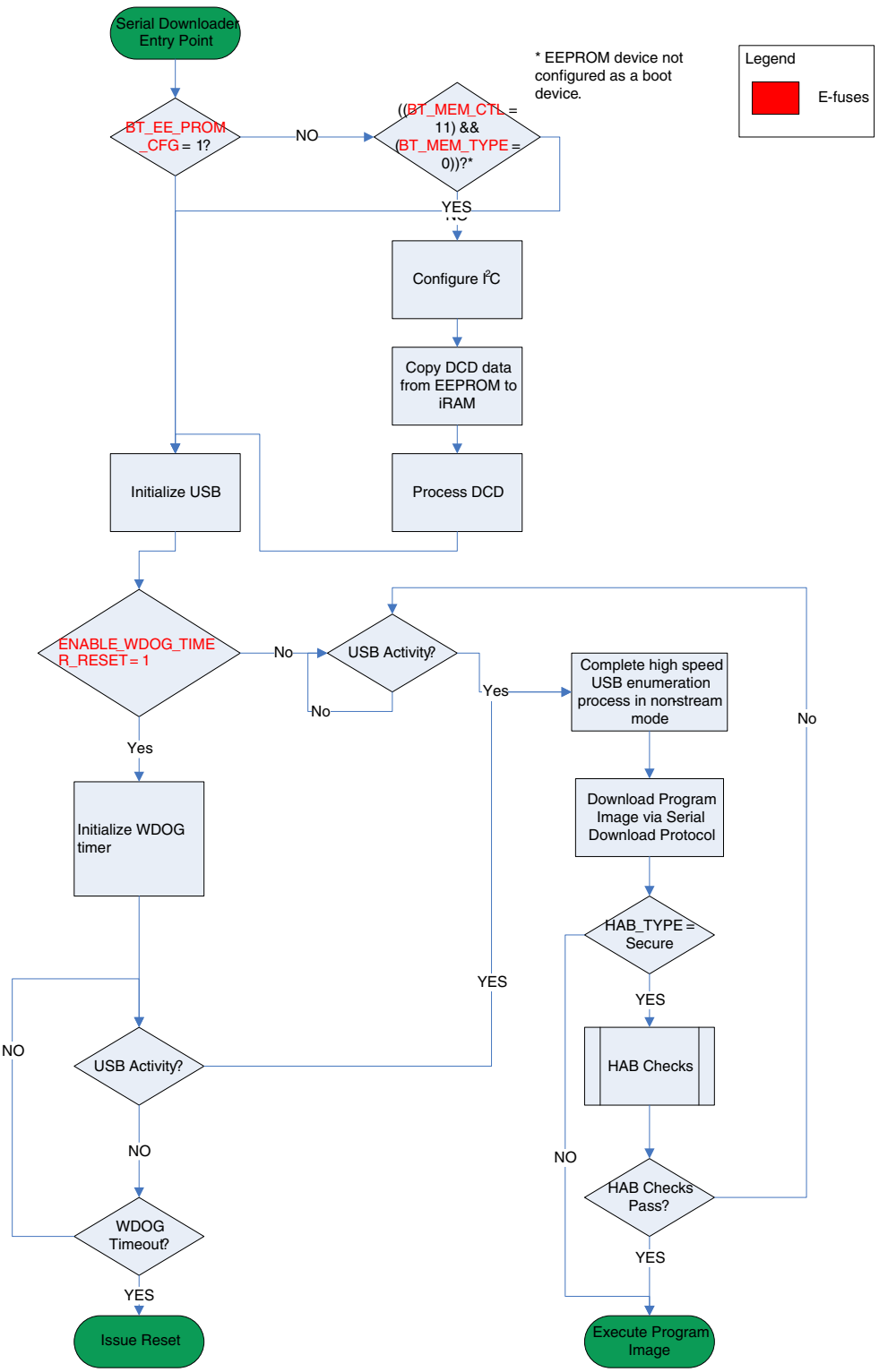


Figure 6-17. USB Boot Flow

6.9.1 USB

USB support in the i.MX50 is composed of the USB1 (USB OTG core controller, compliant with the USB 2.0 specification) and the USBPHY (HS USB transceiver). The i.MX50 ROM supports the USB OTG port for boot purposes. The other USB ports on the i.MX50 are not supported for boot purposes.

The i.MX50 USB Driver is implemented as a USB HID class. A collection of 4 HID reports are used to implement SDP protocol for data transfers.

- Report 1 transfer over control OUT endpoint, 16-byte SDP command from host to device. Total bytes are 17 with first byte reserved for report id set to 1.
- Report 2 transfer over control OUT endpoint from host to device data associated with report 1 SDP command. Max report size is 1025 bytes with first byte reserved for report id set to 2.
- Report 3 transfer over interrupt endpoint from device to host to send HAB security state. Max size of this report is 5 bytes with first byte reserved for report id set to 3. Device sends 0x12343412 in production mode and 0x56787856 in engineering mode in the remaining 4 bytes of this report.
- Report 4 transfer over interrupt endpoint from device to host to send data in response to SDP command in report 1. Max size of this report is 65 with first byte reserved for report id set to 4.

6.9.1.1 USB Configuration

The default VID/PID and strings for USB device driver are listed in Table 6-37.

The VID and PID can be customized using 16-bit efuses USB_VID and USB_PID.

Table 6-37. VID/PID and Strings for USB Device Driver

Descriptor	Value
VID	0x15A2 (Freescale vendor ID)
PID[1]	0x0052
Screen Descriptor1 (manufacturer)	Freescale Semiconductor, Inc.
Screen Descriptor2 (product)	S Blank i.MX50 SE Blank i.MX50 NS Blank i.MX50

Table continues on the next page...

Table 6-37. VID/PID and Strings for USB Device Driver (continued)

Descriptor	Value
Screen Descriptor4	Freescale Flash
Screen Descriptor5	Freescale Flash

1. Allocation based on BPN (Before Part Number)

6.9.1.2 Serial Download protocol

The 16 byte SDP command from host to device is sent using HID report 1. The following table describes 16 byte SDP command data structure:

Table 6-38. 16 Byte SDP Command Data Structure

BYTE Offset	Size	Name	Description
0	2	COMMAND TYPE	Following commands are supported for i.MX50 ROM: • 0x0101 READ_REGISTER • 0x0202 WRITE_REGISTER • 0x0404 WRITE_FILE • 0x0505 ERROR_STATUS • 0x0A0A DCD_WRITE (new to i.MX50) • 0x0B0B JUMP_ADDRESS (new to i.MX50)
2	4	ADDRESS	Only relevant for following commands: READ_REGISTER, WRITE_REGISTER, WRITE_FILE and JUMP_ADDRESS. For READ_REGISTER and WRITE_REGISTER commands, this field is address to a register. For WRITE_FILE and JUMP_ADDRESS commands, this field is an address to internal or external memory address.
6	1	FORMAT	Format of access, 0x8 for 8-bit access, 0x10 for 16-bit and 0x20 for 32-bit access. Only relevant for READ_REGISTER and WRITE_REGISTER commands.
7	4	DATA COUNT	Size of data to read or write. Only relevant for WRITE_FILE, READ_REGISTER, WRITE_REGISTER and DCD_WRITE commands. For WRITE_FILE command DATA COUNT is in bytes and for DCD_WRITE DATA COUNT means number of registers to write.
11	4	DATA	Value to write. Only relevant for WRITE_REGISTER command.
15	1	RESERVED	Not used in i.MX50 ROM

6.9.1.3 SDP Command

SDP commands are decribed in the following sections.

6.9.1.3.1 READ REGISTER

The transaction for command READ_REGISTER consists of following reports: Report1 for command, Report3 for HAB mode and Report4 for response or register value. The register to read is specified in ADDRESS field of SDP command. First device sends report-3 with hab mode followed by report 4 with bytes read at given address. If count is greater than 64 then multiple reports with report id 4 are sent until entire data requested by host is sent. The STATUS is either 0x12343412 for production parts and 0x56787856 for development parts.

Report1, Command, Host to Device:

2	Valid values for READ_REGISTER COMMAND, ADDRESS, FORMAT, DATA COUNT

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes HAB mode indicating Production/Development part

ID 4 bytes status

Report4, Response, Device to Host: first response report

4	Register Value

ID 4 bytes of data containing register value. If number of bytes requested is less than 4 then remaining bytes should be ignored by host.

... Multiple reports of report id 4 are sent until entire data requested is sent

Report4, Response, Device to Host: last respose report

4	Register Value

ID 64 bytes of data containing register value. If number of bytes requested is less than 64 then remaining bytes should be ignored by host.

6.9.1.3.2 WRITE REGISTER

The transaction for command WRITE_REGISTER consists of following reports: Report1 for command, Report3 for hab mode and Report4 for write status. Host sends Report1 with WRITE_REGISTER command. The register to write is specified in ADDRESS field of SDP command of Report1, with FORMAT field set to data type (number of bits to

write 8, 16 or 32) and value to write in DATA field of SDP command. Device writes the DATA to register address and returns WRITE_COMPLETE code using Report4 and Development/Engineering part status using Report3 to complete the transaction.

Report1, Command, Host to Device:

2	Valid values for WRITE_REGISTER COMMAND, ADDRESS, FORMAT, DATA COUNT and DATA

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes HAB mode indicating Production/Development part

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes of data with first 4 bytes to indicate write is completed with code 0x12828212. On failure device will report HAB error status.

6.9.1.3.3 WRITE_FILE

The transaction for command WRITE_REGISTER consists of following reports: Report1 for command, Report3 for hab mode and Report4 for write status. Host sends Report1 with WRITE_REGISTER command. The register to write is specified in ADDRESS field of SDP command of Report1, with FORMAT field set to data type (number of bits to write 8, 16, or 32) and value to write in DATA field of SDP command. Device writes the DATA to register address and returns WRITE_COMPLETE code using Report4 and Development/Engineering part status using Report3 to complete the transaction:

Report1, Command, Host to Device:

1	Valid values for WRITE_FILE COMMAND, ADDRESS, DATA COUNT

ID 16 byte SDP Command

==========Optional Begin==========

Host sends ERROR_STATUS command to query if HAB rejected the address

==========Optional End==========

Report2, Data, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report2, Data, Host to Device:

2	File data

ID Max 1024 bytes data per report

…

Report3, Response, Device to Host:

3	4 bytes HAB mode indicating Production/Development part

ID 4 bytes status

Report4, Response, Device to Host:

4	COMPLETE (0x88888888) status

ID 64 bytes data with first 4 bytes to indicate file download has completed with code 0x88888888. On failure device will report HAB error status.

6.9.1.3.4 ERROR_STATUS

The transaction for SDP command ERROR_STATUS consists of three reports. Report1 is used by host to send the command; device sends global error status in 4 bytes of Report4 after returning Development/Engineering status code in Report3. When device receives ERROR_STATUS command it will return global error status that is updated for each command. This command is useful to find out if last command resulted in device error or succeeded.

Report1, Command, Host to Device:

1	ERROR_STATUS COMMAND

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes HAB mode indicating Production/Development part

ID 4 bytes status

Report4, Response, Device to Host:

4	4 bytes Error status

ID first 4 bytes status in 64 bytes report 4

6.9.1.3.5 DCD WRITE

The SDP command DCD_WRITE is used by host to send a bunch of register writes in one shot. This command is provided to speed up the process of programming register writes such as to configure external RAM device. The command goes with Report1 from host with COMMAND TYPE set to DCD_WRITE and DATA_COUNT to number of registers sent in data out phase. In data phase host sends data for number of registers using Report2. The number of registers per DCD_WRITE command is limited to 85; this is mainly to fit the data in one data out packet from host. If there are more than 85 registers to write, host will have to send additional DCD_WRITE commands to device. Each register data needs 12 bytes: 4 bytes for FORMAT (data type, 8 for byte, 16 for 2 bytes and 32 for 4 bytes), 4 bytes for address of the register and 4 bytes for value to write to the register. If there are n registers to write then host will send n times 12 bytes of data in data phase. Device completes the transaction with Report3 indicating Development/ Production STATUS and report 4 with WRITE_COMPLETE code 0x12828212.

Report1, Command, Host to Device:

1	DCD_WRITE COMMAND, DATA_COUNT

ID 16 byte SDP Command

Report2, Data, Host to Device:

2	DCD binary data

ID Max 1024 bytes (85 register writes) per report

Report3, Response, Device to Host:

3	4 bytes HAB mode indicating Production/Development part

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes report with first 4 bytes to indicate write is completed with code 0x128A8A12. On failure device will report HAB error status.

Table 6-39 displays the format of DCD binary data:

Table 6-39. DCD Data for n Registers

4 bytes for data format	4 bytes for register1 address	4 bytes for register1 value to set
4 bytes for data format	4 bytes for register2 address	4 bytes for register2 value to set
4 bytes for data format	4 bytes for register3 address	4 bytes for register3 value to set
...	...	...
4 bytes for format	4 bytes for last (nth) register address	4 bytes for last register (nth) value

6.9.1.3.6 JUMP ADDRESS

The SDP command JUMP_ADDRESS will be the last command host can send to the device, after this command device will jump to the address specified in the ADDRESS field of SDP command and start executing. This command should typically follow after WRITE_FILE command. The command is sent by host in command-phase of transaction using Report1, there is no data phase for this command but device send status report3 to complete the transaction. And if HAB authentication fails then it will also send report 4 with HAB error status.

Report1, Command, Host to Device:

1	JUMP_ADDRESS COMMAND

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes HAB mode indicating Production/Development part

ID 4 bytes status

This report is sent by device only in case of an error jumping to the given address, device reports error in Report4, Response, Device to Host:

4	4 bytes Error status

ID 4 bytes status, 64 bytes report length

6.9.2 External Entry

In some use cases it may be desirable to call into the ROM USB downloader from program image code. The ROM provides an entry point for the USB downloader through ROM Vector Table (hab_rvt) structure. The interface to call into usb downloader is hab_rvt.failsafe. ROM Vector Table structure is located at address 0x00000094 in ROM memory region as discussed in ROM Vector Table Addresses. The failsafe member of RVT structure is at offset 0x28. To call into ROM the program image should jump to the address at hab_rvt.failsafe.

Note that pu_irom_boot_decision is compiled for Thumb state on the ARM processor core.

6.9.2.1 pu_irom_boot_decision

Syntax:

```
void pu_irom_boot_decision(void)
```

Description:

Entry point for the USB Downloader. Performs EEPROM DCD processing if BT_EEPROM_CFG OCOTP bit is not blown.

Inputs:

None.

Returned Value:

None.

PreConditions/Assumptions:

- BT_EEPROM_CFG is set appropriately

Post Conditions:

None.

6.10 High Assurance Boot (HAB)

The High Assurance Boot (HAB) component of the ROM protects against the potential threat of attackers modifying areas of code or data in programmable memory to make it behave in an incorrect manner. The HAB also prevents attempts to gain access to features which should not be available. The integration of the HAB feature with the ROM code ensures that i.MX50 does not enter an operational state if the existing hardware security modules have detected a condition that may be a security compromise or areas of memory deemed to be important have been modified. The HAB uses RSA digital signatures to enforce these policies.

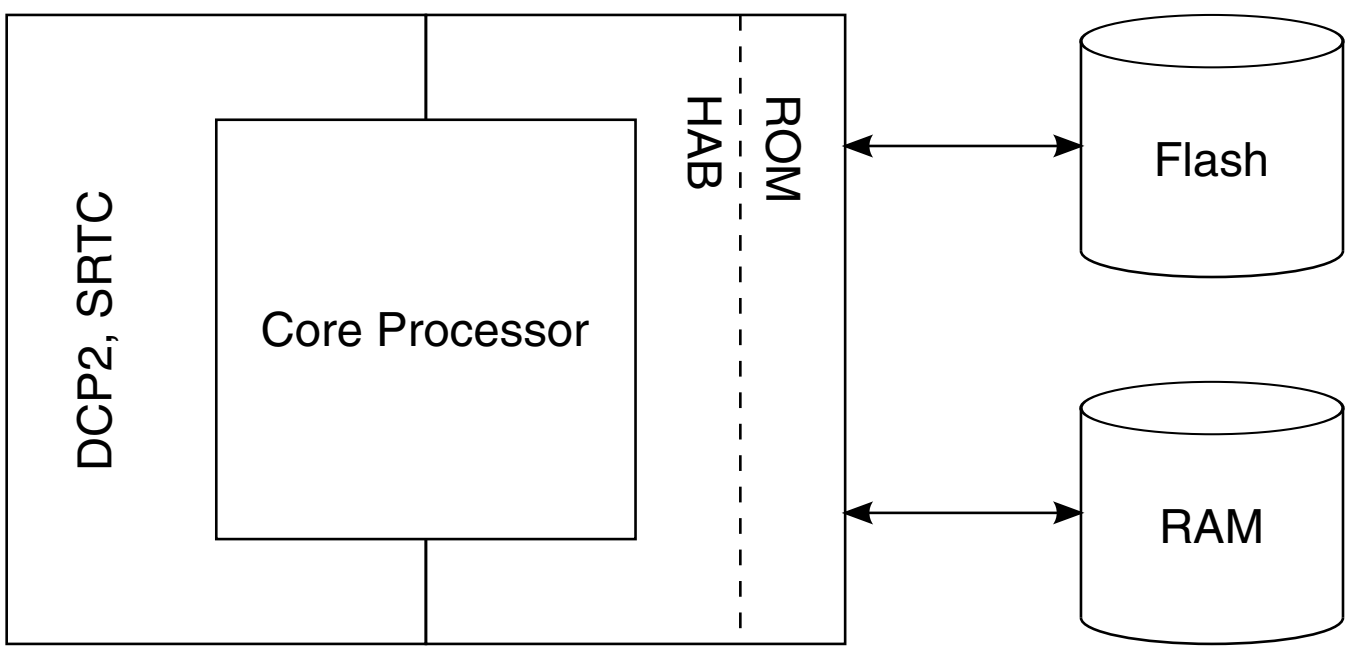


Figure 6-18. Secure Boot Components

Figure 6-18 illustrates the components used during a secure boot using HAB. The HAB makes use of the DCP2 hardware module to accelerate SHA-256 message digest operations performed during signature verifications. The HAB also includes a software implementation of SHA-256 for cases where a hardware accelerator cannot be used. The core RSA signature verification operations are performed by a software implementation contained in the HAB library. The main features supported by HAB are:

- X.509 Public key certificate support
- CMS signature format support

For further details on making use of the i.MX50 secure boot feature using HAB please contact your local Freescale representative.

6.10.1 ROM Vector Table Addresses

For devices that perform a secure boot, the HAB library may be called by boot stages that execute after ROM code. The RVT table contains the pointers to the HAB API functions and is located at 0x00000094 For additional information on secure boot on i.MX50 including the HAB API please contact your local Freescale representative

6.10.2 SRTC Initialization

For details on SRTC initialization when performing a secure boot with HAB please contact your local Freescale representative.

Chapter 7
System Debug

7.1 Overview

This chapter describes the hardware and software debug and application development features and resources of the i.MX50. There are core/platform-specific resources, resources associated with some of the more complex IP blocks, and chip-wide resources. Also discussed is the interface to external debug and development tools.

The debug architecture of the i.MX50 is based on that of previous members of the i.MX application processor family. A small number of changes and extensions were necessary to accommodate new IP.

An overview the i.MX50's primary debug capabilities and features include:

- JTAG-based access (control and monitoring), built around the System JTAG Controller (SJC)
- Trace Port
- Real-time and Halt-mode debug and profiling capabilities of the ARM Cortex™-A8 core platform
- Real-time and Halt-mode debug capabilities of the SDMA core
- An SOC-wide cross-trigger system built around ARM's Embedded Cross Trigger (ECT)
- Visibility of pre-selected critical internal signals via pin muxing
- External memory interface/controller arbitration profiling

Each of these features will be described in detail in the following sections of this chapter. The i.MX50 also support ARM CoreSight architecture for system debug and trace capabilities.

7.1.1 Debug Strategy

- Software debug-(MSFT kernel)

- Trace the ARM platform activity through ETM trace port or ETB (ARM only)
- Control and monitor through JTAG
- Monitor pre-selected critical internal signals through pin muxing-scope

7.2 System JTAG Controller-SJC

The SJC module is the bridge between external development and test instrumentation and the internal JTAG-accessible debug and test resources. It implements and manages the daisy-chained topology consisting of it's own TAP and those of the SDMA, a DSP core (not implemented on the i.MX50), the ARM core, and the ARM Debug Access Port (DAP). ARM core isn't part of a daisy-chain. Also, DSP TAP is excluded by SJC configuration tie-off, so there is no need to use BYPASS IR for it at all.

7.2.1 JTAG Topology

Figure 7-1 shows the JTAG topology.

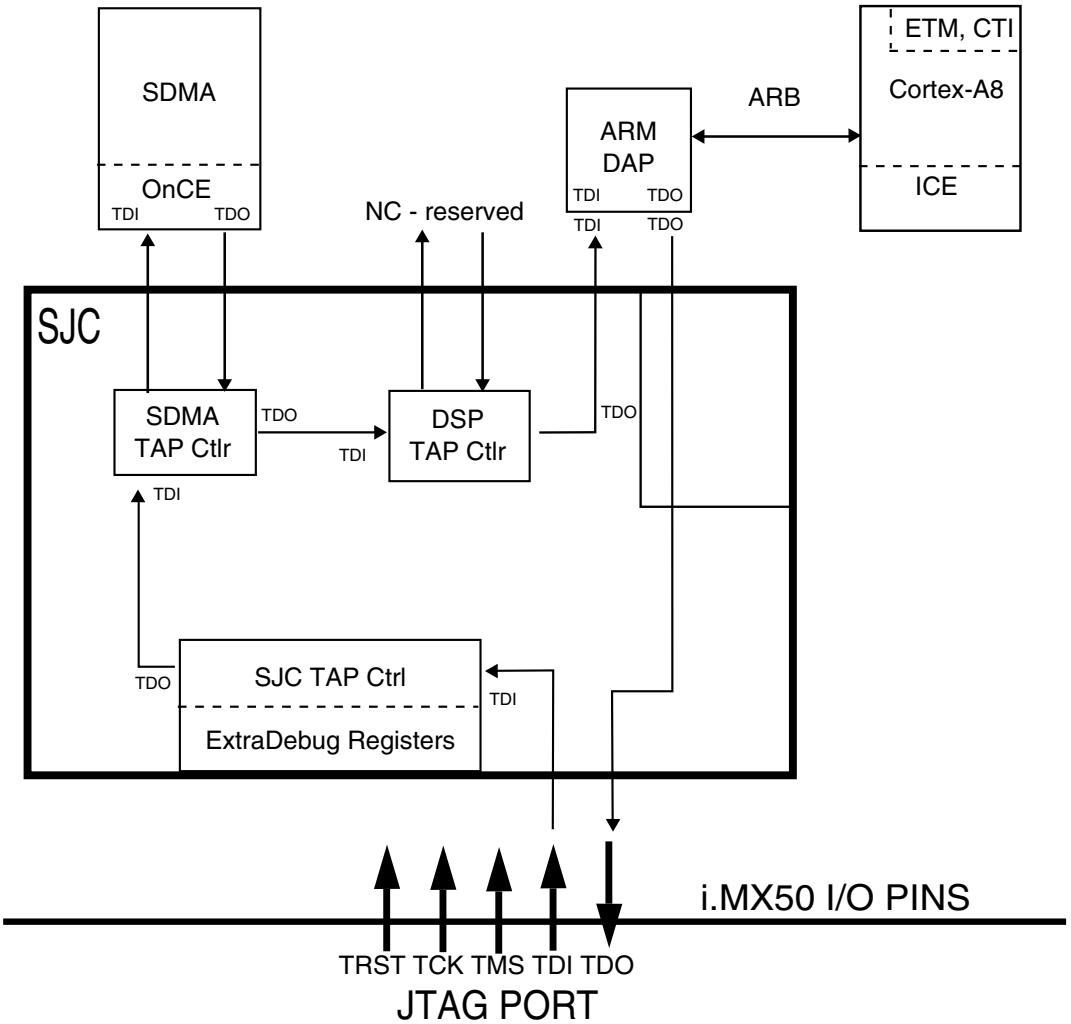


Figure 7-1. JTAG Connectivity

7.2.2 System JTAG Controller Main Features

The following list describes the System JTAG Controller's main features:

- IEEE P1149.1 (standard JTAG) interface to off-chip test and development equipment
  - includes an SJC-only mode for true IEEE 1149.1 compliance, used primarily for board-level implementation of boundary scan.
- Debug-related control and status, such as putting selected cores into reset and/or debug mode and the ability to monitor individual core status signals via JTAG.
- System status, such as the state of the PLLs (locked or not locked)
- Generic status and control, defined on a per-SOC basis by the architecture team

- Factory test related control/status such as PLL bypass and memory BIST
- 3 levels of security, ranging from no security to no JTAG accessibility to the chip.

7.2.3 SJC TAP Port

The SJC in i.MX50 supports the following standard JTAG pins: TRSTB, TDI, TDO, TCK, and TMS.

7.2.4 SJC Main Blocks

The following list describes the SJC main blocks:

- Interface to the outside world via the standard JTAG pins
- Interface to the external Debug_Event pin
- A master TAP controller, which implements the standard JTAG state machine.
- Implementation of the mandatory and optional IEEE P1149.1 (JTAG) instructions
  - Mandatory: "EXTEST", "SAMPLE/PRELOAD", and "BYPASS"
  - Optional: "ID_CODE" (SOC JTAG ID register), "HIGHZ"
- Supports the SDMA's DR-path-only JTAG architecture by implementing the controller portion of it's TAP (including "BYPASS" as the default state) within the SJC
- Provides the serial interfaces to various DSP-domain JTAG resources (not implemented on i.MX50)
- The ExtraDebug registers, which implement a variety of control and status features
  - Three 32-bit insecure general purpose status registers
  - Two 32-bit secure status registers-one predefined, one general purpose.
  - Control and status registers for debug, core, charge pump, PLL, and BIST related functions
  - Control bits for memory timing control (not used on the i.MX50)
- Three levels of fuse-defined security, ranging from no security to no access.

Both predefined and user-defined (SOC integration) Control and Status functions are supported by the SJC. The user-defined functions will be defined and documented by the SOC integration.

7.2.5 i.MX50 Specific SJC Features

7.2.5.1 JTAG Disable Mode

In addition to three different JTAG security modes that are implemented internally in the System JTAG Controller (SJC), there is an option to disable the SJC functionality by e-fuse configuration. This creates additional JTAG mode "JTAG Disabled" with highest level of JTAG protection. In this mode all JTAG features are disabled. Specifically, the following debug features are disabled in addition to the features that were already disabled in "No Debug" JTAG mode:

- Memory BIST
- Boundary scan register (BSR)
- Non-Secured JTAG control registers (PLL configuration, Deterministic Reset, PLL bypass)
- Non-Secured JTAG status registers (Core status)
- Chip Identification Code (IDCODE)

7.2.5.2 SJC i.MX50 Specific Registers Summary

There are many specific SJC registers for the i.MX50, for detailed description of System Jtag Controller(SJC) chapter.

7.2.5.3 JTAG ID

JTAG_ID:0x02D0_101D.

For detail description of the JTAG_ID, refer to the "System Jtag Controller (SJC)" chapter.

7.3 CoreSight Design Kit

The i.MX50 include ARM CoreSight component for multi core debug and trace solution. CoreSight component can be found in a few hierarchy levels: ARM core, ARM platform and top level.

- ARM core: include the following CoreSight components: ETM, CTI0
- ARM platform: include the following CoreSight components: ETB, ATB Replicator, CTI1, CTM
- Top level: include the following CoreSight components: DAP, CTI2, CTI3, TPIU

7.3.1 Memory Map and Register Definition

Each CSDK component has a 4 Kbyte location block in the CoreSight memory map. The base address of the CoreSight location block is not fixed but the address offsets are fixed. Each CSDK component has a 4KB memory map, that is, 1 K words.

Components connected to the DAP internal bus appear as part of a memory mapped structure with parallel address and two data busses, read data and write data. The bus master, JTAG-DP, controls the various APs, in this structure the slaves, using normal bus transactions. The JTAG-DP reads and writes registers within this bus.

7.3.2 CoreSight Clock Enable

By default the CoreSight component clocks are enabled, the system get out from reset with all clocks enabled. The CoreSight clocks gating are controlled by programmable CCM registers and DBGEN. CCM allow control of each CoreSight component separately (by default allowed) and DBGEN that allow and disable all debug clocks. DBGEN will be raised when detecting activity on JTAG ports (TMS) and descend on reset (by dap_sys) or controlled by ARM. Only if both CCM registers bits and DBGEN are high the clock will be propagate to CoreSight component.

7.3.3 CoreSight Debug Access Port (DAP) and DAP_SYS

ARM's Debug Access Port (DAP) has several functions. DAP_SYS is a wrapper around DAP and is the FSL block that is actually instantiated in a design (DAP is contained within DAP_SYS). The following list is a summary of the features provided by DAP_SYS:

- Enables debug-related communication between various parts of the system
  - It is a modular block
  - Slave-side support for JTAG and APB (ARM Peripheral Bus) protocols.
  - Debug master can access many debug resources in real time without having to halt the core
  - Enabling of system access to anything connected to the Debug-APB via the APB multiplexor.
- ROM Table-provides a list of memory locations of CoreSight components connected to the Debug APB. Visible from both tools and system access.
- AHB to APB gasket to translate System AHB accesses to APB (input as system APB to the DAP).
- APB Decode to issue select signals to CoreSight components on APB, and handle the PSLVERR and PREADY signals from the components to the DAP.

- AHB-Lite to AHB converter, to translate DAP's AHB-lite accesses to system's AHB protocol.
- Debug and Reset logic to generate DBGEN signal and reset signals.

The CoreSight components that reside on the Debug-APB include all Embedded Cross Trigger blocks, the ETM, and the ETB.

7.3.4 Embedded Cross Trigger (ECT)

System events, such as a debug request, a core entering or exiting debug mode, the occurrence of a breakpoint or watchpoint, the occurrence of a particular error, the state of a buffer, etc., can be useful or even essential for debugging or profiling system performance. Whatever the source of the event, the embedded cross trigger implements a flexible, programmable mechanism to transport these events from a source to one or multiple destinations. This includes handling handshake requirements with both the source and the destination as needed and synchronization of signals from different clock domains. The end result is that events of interest that occur in one domain, such as in the Cortex-A8 domain, can be recognized and responded to be a destination domain, such as the SDMA.

i.MX50 uses the CoreSight ECT (provided by ARM as part of their full CoreSight Debug Support package) and a custom wrapper to accommodate the various necessary interface scenarios. The ECT (as delivered by ARM) consists of two major blocks - a central Cross Trigger Matrix (CTM) and a Cross Trigger Interface (CTI) block. A Freescale-customized wrapper is added to the CTI to accommodate a variety of different input and output interface scenarios. The wrapped CTI is called the Extended CTI and is configured on a signal by signal basis with port tie-off's.

i.MX50's ECT architecture consists of a single CTM and three wrapped CTI's covering 3 basic groups of functionality as follows:

- CTI0-is in the Cortex-A8 core
- CTI1-is in the Tiger cortex-A8 core platform, 2 trigger in and 2 trigger out is used by the platform and the others by the device peripherals
- CTI2-is in the SOC level, is by SDMA
- CTI3-is in the SOC level, is used by MCU peripherals

Detail CTI pin assignment can be found in CoreSight CTI.

Figure 7-2 is a simple illustration of the relationship between the two processor domains, the SJC and peripherals, and the ARM CoreSight DAP. APB is an AMBA bus used for debugging. It defines what debug and trace components are required and how they are connected. The DAP is the Debug-APB master of the following debug units: 4 CTI

blocks, ETM, ETB, TPIU and Cortex-A8 debug unit. The Debug-APB bus differs from system-level APB's in that it can be directly accessed via JTAG as well as from the ARM core. Module selects for each slave module on the Debug-APB are generated by the DAP_SYS.

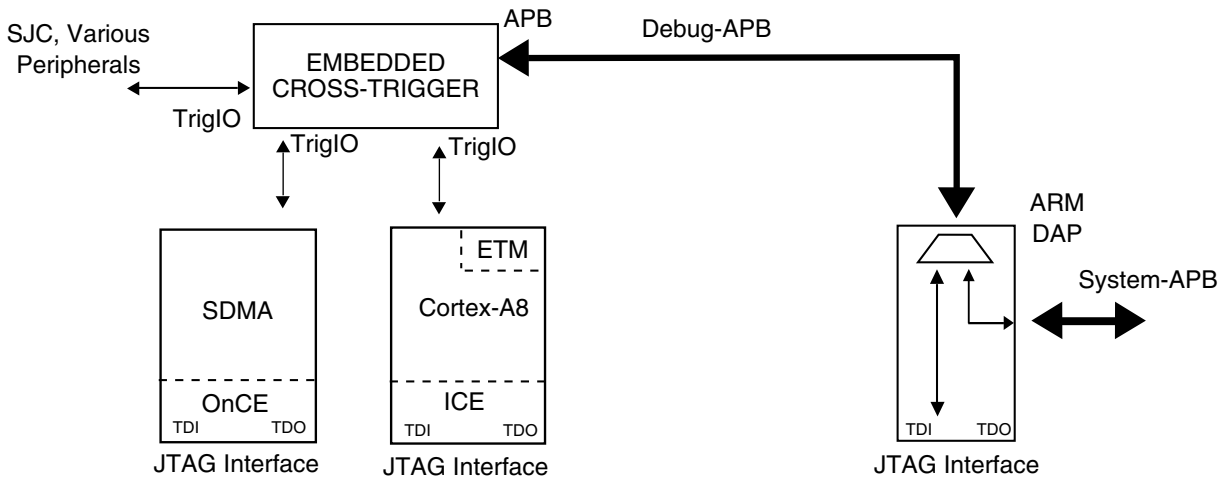


Figure 7-2. Embedded Cross Trigger Topology

Figure 7-3 illustrates the connectivity between the CTM and CTIO components of the CoreSight ECT.

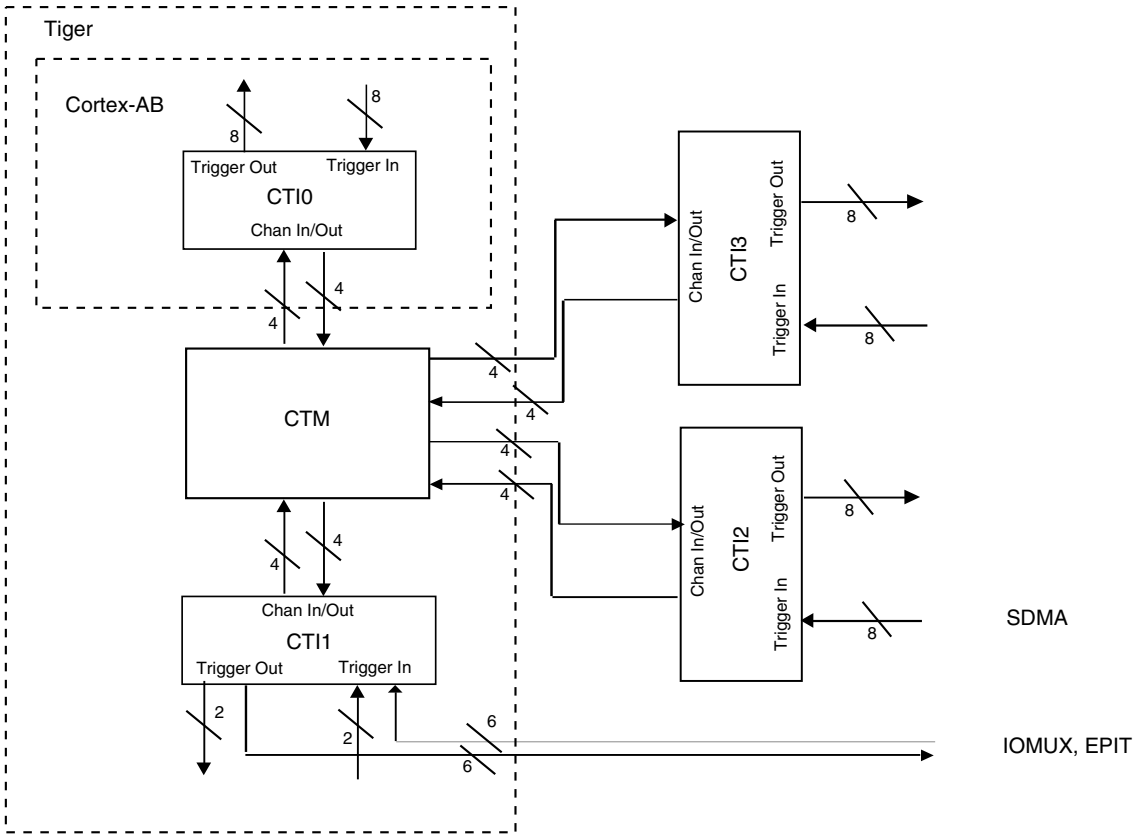


Figure 7-3. Embedded Cross Trigger-Basic Architecture

7.3.4.1 CoreSight CTM

The CTM (Cross Trigger Matrix) is provided by ARM. . The CTM is a relatively simple block with no programmability. On i.MX50, it's configured to have 4 input and 4 output channels. Whatever comes in on an input channel is simply routed to all corresponding output channels. For instance, if a trigger input comes in on input channel 0 from CTI0, it is routed to output channel 0 to CTI1, CTI2 & CTI3. All selection of drivers and receivers for an event is done in the individual CTI blocks.

The final feature provided by the CTM is handshake with each CTI. This insures that as trigger signals cross different clock domain boundaries, they won't be lost due to slower downstream sampling frequencies. It is a simple mechanism in which a CTI holds an output until it receives the acknowledgement from the CTM. Handshake is also implemented in the reverse direction, insuring that a trigger signal propagating from the CTM to a CTI won't be missed by the CTI.

7.3.4.2 CoreSight CTI

The CTI (Cross Trigger Interface) is also provided by ARM. A summary description of the CTI is provided here, but please refer to ARM documentation for detailed information. As mentioned earlier, there are 3 CTI's in i.MX50. Each of these has 8 trigger inputs and 8 trigger outputs that connect to logic in the domain to be debugged or profiled. Each CTI also includes a 4 channel interface to the CTM (4 inputs and 4 outputs as shown in Table 7-1 through Table 7-6). The CTIs are wrapped with additional logic to from the Extended CTI, which is described in the next section.

Table 7-1. CTI 1 Input Assignments

Trigger Input	Debug Resource Name	Channel Function	SYNC (dbg_atclk)	Active	ack	Comment
0	N/A	Configured interrupt	No	High	No	-
1	N/A	Configured interrupt	No	High	No	-
2-3	-	TRACING	-	-	-	ARM
4	N/A	Configured interrupt	No	High	No	-
5	N/A	Configured interrupt	No	High	No	-
6	N/A	Configured interrupt	No	High	No	-
7	cti_in[7]	Debug	-	-	Yes	IOMUX

Table 7-2. CTI 1 Output Assignments

Trigger Output	Debug Resource Name	Channel Function	SYNC (dbg_atclk)	Active	ack	Comment
0	intin_b	General Purpose	No	-	No	will allow creating interrupts
1	intin_b	General Purpose	No	-	No	will allow creating interrupts
2	intin_b	General Purpose	No	-	No	will allow creating interrupts
3	intin_b	General Purpose	No	-	No	will allow creating interrupts
4-5	-	-	-	-	-	ARM
6	cti_out[6]	Debug	No	-		IOMUX
7	cti_out[7]	Debug	No	-	-	IOMUX

Table 7-3. CTI 2 Input Assignments

Trigger Input	Debug Resource Name	Channel Function	SYNC (debug_apb_clk)	Active	ack	Comment
0	debug_mode	Debug Acknowledge	Yes	High	No	SDMA
1	debug_core_run	debug	Yes	High	No	SDMA
2	evt_chn_lines[0]	debug	Yes	High	No	SDMA
3	evt_chn_lines[1]	debug	Yes	High	No	SDMA
4	evt_chn_lines[2]	debug	Yes	High	No	SDMA
5	req_ma	debug	Yes	High	No	SPBA
6	req_mb	debug	Yes	High	No	SPBA
7	req_mc	debug	Yes	High	No	SPBA

Table 7-4. CTI 2 Output Assignments

Trigger Input	Debug Resource Name	Channel Function	SYNC (debug_apb_clk)	Active	ack	Comment
0	events[31]	-	Yes	High	No	SDMA
1	sdma_dreq_i	-	Yes	High	No	SDMA
2	FLUSHIN	-	No	High	Yes	TPIU, FLUSHINACK
3	TRIGIN	-	No	High	Yes	TPIU, TRIGINACK
4	system_debug	-	Yes	High	No	SJC
5	-	-	-	-	-	Not used
6	-	-	-	-	-	Not used
7	-	-	-	-	-	Not used

Table 7-5. CTI 3 Input Assignments

Trigger Input	Debug Resource Name	Channel Function	SYNC (debug_apb_clk)	Active	ack	Comment
0	ipi_int_epit_oc	TRACING	Yes	High	No	EPIT
1	-	-	-	-	-	Not used
2	-	-	-	-	-	Not used
3	-	-	-	-	-	Not used

Table continues on the next page...

Table 7-5. CTI 3 Input Assignments (continued)

Trigger Input	Debug Resource Name	Channel Function	SYNC (debug_apb_clk)	Active	ack	Comment
4	-	-	-	-	-	Not used
5	-	-	-	-	-	Not used
6	-	-	-	-	-	Not used
7	-	-	-	-	-	Not used

Table 7-6. CTI 3 Output Assignments

Trigger Input	Debug Resource Name	Channel Function	SYNC (debug_apb_clk)	Active	ack	Comment
0	-	-	-	-	-	Not used
1	-	-	-	-	-	Not used
2	-	-	-	-	-	Not used
3	-	-	-	-	-	Not used
4	-	-	-	-	-	Not used
6	-	-	-	-	-	Not used
7	-	-	-	-	-	Not used

7.3.4.3 Extended CTI (CTI Wrapper)

This wrapper consists of additional logic for each input and each output trigger to implement several features that can be selected or enabled with hard tie-off's on the module's boundary. It is combined with ARM's CTI block to form the CTI_Extended block. A summary of the extended input features implemented by logic in the wrapper is as follows:

- Invert the incoming signal
  - The ARM CTI assumes all inputs are active-high. An active-low input must first be inverted before going into the ARM CTI
- Sample the incoming trigger signal with a clock synchronous to the incoming signal
  - Samples the incoming trigger with it's own clock
- Convert the incoming signal to a pulse
  - Used when an input trigger has a long assertion time. Destinations may be sensitive to a long assertion time and may see it as multiple input triggers.

- Implement holding logic to insure that the ARM CTI captures and acknowledges the incoming signal. The holding logic clock must be synchronous to the clock that generated the input trigger.
- Provides a synchronized or asynchronous acknowledge signal back to the origin of the trigger.

If the asynchronous trigger source requires an ACK signal back from the CTI, but a clock from that source is not available to the CTI, the source must first synchronize the received ACK before using it in the trigger source logic.

7.3.5 CoreSight Trace Port Interface (TPIU)

The TPIU is one of the CoreSight trace sink components it acts as a bridge between the on-chip trace data to a data stream that is then driven out the trace port. ATB interface is used by the TPIU accepts trace data from a trace source, either direct from a trace source or using a Trace Funnel. The TPIU has 32-bit connected to the Chip pad. The APB interface is the programming interface for the TPIU configuration.

7.4 Cortex-A8 Core and Platform

Cortex-A8 Debug architecture includes support for CoreSight. The memory-mapped external debug interface replaces the coprocessor interface defined in the previous version of the Debug architecture. A full access to the processor debug capability available by Cortex-A8 debug register map through the Advanced Peripheral Bus (APB) slave port. The core include Processor Debug Unit allow: stop program execution, examine and alter processor and coprocessor state, examine and alter memory and input/output peripheral state and restart the processor core.

7.4.1 Cortex-A8 Core Debug Support Features

- CoreSight Embedded Trace Macro (ETM11)-race generator for the Cortex-A8 core
- Support for eFuse-related 3-level debug scheme:
  - Debug everywhere
  - Debug in Non-Secure privileged and user, and Secure user
  - Debug in Non-Secure only
- Embedded ICE-RT logic
  - Support for both monitor-mode and halt-mode debugging.

    - Core run/halt control, debug status/control
    - Breakpoint/watchpoint control
    - Core- and memory-mapped resource examination/modification
- Data communication channel between ARM core and host debugger via JTAG
- PMU-Performance Metrics Unit used for system profiling and debug.
- CP15 register for debugging the MMU, I & D L1 cache, and TLB
- EVTMON - L2 Event Monitor-Supports debug and profiling of the ARM's L2 cache activity

7.4.2 Embedded Cross Trigger Interface

Please refer to the Embedded Cross Trigger section of this chapter for detailed information about the Core and platfrom interface to that subsystem.

7.4.3 Additional Platform Debug Functionality

- CoreSight Embedded Trace Buffer (ETB11)-4 Kbyte RAM array to be used for on-chip capture of trace data output from the ETM11.
- ATB Replicator to connect the trace data to TPIU (Trace Port Interface) and ETB (Embedded Trace Buffer).
- Debug Visibility-select critical signals routed to the I/O pads as alternate outputs for external visibility

7.5 Smart DMA (SDMA) Core

The SDMA is a dedicated, programmable DMA engine. It is an integration of a 32-bit RISC core and DMA-specific hardware, and includes ports for the AP domain and a peripheral domain, along with a burst-capable port for direct external memory access. The SDMA and it's integration in i.MX50 is unchanged from previous SOCs.

The main SDMA debug features:

- OnCE-On Chip Emulator, provides the following capabilities:
    - SDMA core control - run/halt/single-step
    - SDMA core register/memory-map access
    - Event detection, watchpoints, and hardware breakpoints
    - Real time buffer and PC trace buffer capability
- Trace buffer

    - Contains information to identify the 32 last changes of flow detected during a program execution
- Context dump
    - Include information about all the channel dump activity
    - Current contents of SDMA RAM
- SDMA ROMPATCH

7.5.1 SDMA On Chip Emulation Module (OnCE) Feature Summary

The SDMA debug features are primarily defined by the OnCE portion of it's design, which are summarized as follows:

- Memory And Register Access-Dedicated logic enables user-access to SDMA memory and register locations. These accesses are supported only when the processor is in debug mode.
- Event Detection Unit-Watches signals from the data memory bus (DMBus) which is used by the RISC core to access its RAM, ROM, and memory-mapped registers
- Watchpoints-One output signal is available to watch event matching conditions at the chip level. Match conditions are defined by programming memory-mapped registers.
- Hardware Breakpoint-A counter is decremented after an event detection. A debug request is sent to the SDMA core only when the counter reaches the value of zero. It is possible to program the initial value of the counter or to disable the use of the counter if a debug request must be generated after each event detection.
- Real Time Buffer-The Real Time Buffer Register (RTB) is a single 32-bit memory-mapped register which can be accessed as a regular memory location during program execution. It is used to store and retrieve run time information without putting the SDMA in debug mode. Each write to this register causes an event. This register is, in fact, located in the OnCE. Executing through JTAG, a buffer command exports the content of this register through the JTAG port.
- Core Control (Core Status/Single Stepping)-Commands are provided to monitor and control processor activity. The commands can halt the core, rerun the core from another address location, and get processor status.
- Trace Buffer-A 32x32 buffer which records the last 32 changes of flow during program execution. The buffer stores data in a modulo fashion (that is, the 33rd instruction change replaces the 1st). Captured trace information is retrieved via reads to the Trace Buffer Register.

7.5.2 Other SDMA Debug Functionality

The following list is a description of other SDMA debug functionality:

- Core Trace-basic core trace capability is available through debug visibility functionality only. ETM/Nexus trace capability does not exist.
- ROM Patch-can be accomplished by manipulating the CHN0ADDR register through JTAG or via the MCU's ability to write to SDMA OnCE registers. This must be done right after reset and before the SDMA core is enabled to begin processing events.
- Additional debug control/status interaction with the SJC module
  - SJC-controlled Debug Request
  - SJC-readable Debug Acknowledge (in debug mode)
  - Debug clock control - allows SJC to force clocks on for debug purposes
  - Debug core state (SDMA RISC Core State) - 4 bits accessible from the SJC via JTAG
- Debug Visibility-observable outputs as alternate (programmable) output functions of I/O pins
  - Debug Request, Debug Mode
  - Debug Yield
  - Debug Event Channel[5:0] (indicates requesting event or channel being processed)
  - Debug PC [13:0] (for SDMA core trace)
  - Debug core state [3:0]
  - Debug Real-time Buffer write
  - Debug Addr/Data match
  - Debug bus error
  - Debug bus device
  - Debug bus r/w
  - Debug Event Channels[7:0]
  - Debug Core Run (active when core is running)

7.6 Debug Visibility-IOMUX

Certain predefined, internal signals can be viewed at the package pins. The pad logic for most pins includes an IOMUX, allowing that pad to be shared across multiple functions. Each pad has a primary function that is selected at reset. The alternate functions, such as displaying the state of an internal signal for debug purposes, is selected by reprogramming the IOMUX.

Table 7-7. Debug Visibility

Module	Signals	Pad Frequency
SDMA	debug_mode, debug_bus_error, debug_bus_device[4:0], debug_bus_rwb, debug_matched_dmbus, debug_rtbuffer_write, debug_evt_chn_lines[7:0]	50 MHz
TPIU	TRACEDATA[15:0]	80 MHz

The i.MX50 Block Guide will contain a complete listing of debug visibility signals and their assigned pins.

7.7 Supported Tools

i.MX50 support RealView™ ARM Debugger, the debugger should be connect to i.MX50 from host by RealView ICE protocol converter.

7.8 Interrupt Visibility

Similar to debug visibility, i.MX50 includes multiplexors that allow users to view selected internal interrupts. Up to five pads support this as alternate pad output functions, allowing the user to view up to five different interrupt sources simultaneously. These five signals also route to ECT inputs, allowing them to generate ECT trigger events as desired. Table 7-8 lists the 5 pads that could be used to observe internal interrupt signals.

Table 7-8. Interrupts Observe Pads

Signal	PAD
observe_mux_out_0	I2C3_SCL
observe_mux_out_1	I2C3_SDA
observe_mux_out_2	PWM1
observe_mux_out_3	PWM2
observe_mux_out_4	OWIRE

7.9 Miscellaneous

7.9.1 SOC-Level Bus Trace

There is no SOC-level bus trace capability on i.MX50.

7.9.2 Clock/Reset/Power

The interactions between the debug system and the system clock, reset, and power control blocks will include enhancements to insure that status of critical system components can be monitored at all times, and that a debugger can insure that all critical clocks and power are enabled when needed. Specifically, a master signal coming from an SJC control bit and going to the Clock/PLL logic will be implemented for each major clock domain. The same will be done for each power domain. Historically, JTAG based operations have not been possible under certain system conditions. For i.MX50, the debugger must have the ability to determine the status of all critical system elements and to override any condition that would prevent the desired debug resources from operating properly.

Chapter 8
ARM Cortex A8 Platform (ARM Platform)

8.1 Introduction

This chapter discusses the architecture and design of the ARM™Cortex® A8 platform. Detailed design information of each block within the platform is not covered in this document.

8.2 Overview

A block diagram of the ARM platform is shown in Figure 8-1. The ARM Platform consists of the ARM Ltd. ARM processor which includes a NEON™ co-processor, L1 cache, L2 cache, Embedded Trace Macrocell (ETM) and a Cross Trigger Interface (CTI). The platform includes the essential sub-blocks: platform control, test logic and debug (Cross trigger Matrix (CTM), Embedded Trace Buffer (ETB), and a second CTI).

The ARM processor instruction and data read/write AXI master port is connected from the non-ARM Platform side of the Level 2 Cache. The L2 can access external L3 memory through this port.

The core platform supports static debug through the debug logic to SoC. This includes the capability of real time trace through ARM's Coresight ETM, ETB and CTM sub-blocks. The second CTI sub-block allows cross triggering of internal and external trigger sources.

The ARM platform has two power domains (LP & GP) which are separated by level shifters. The LP power domain serves as a interface to the rest of the SoC. The GP power domain is completely contained within the platform and allows the ARM core and subsystem to run at much higher frequencies than the rest of the SoC.

The boundary of the two power domains is also an asynchronous boundary between the ARM platform and the rest of the SoC. Synchronizers in both the GP and LP power domains of the platform allow the ARM core and subsystem to run asynchronously from the rest of the SoC.

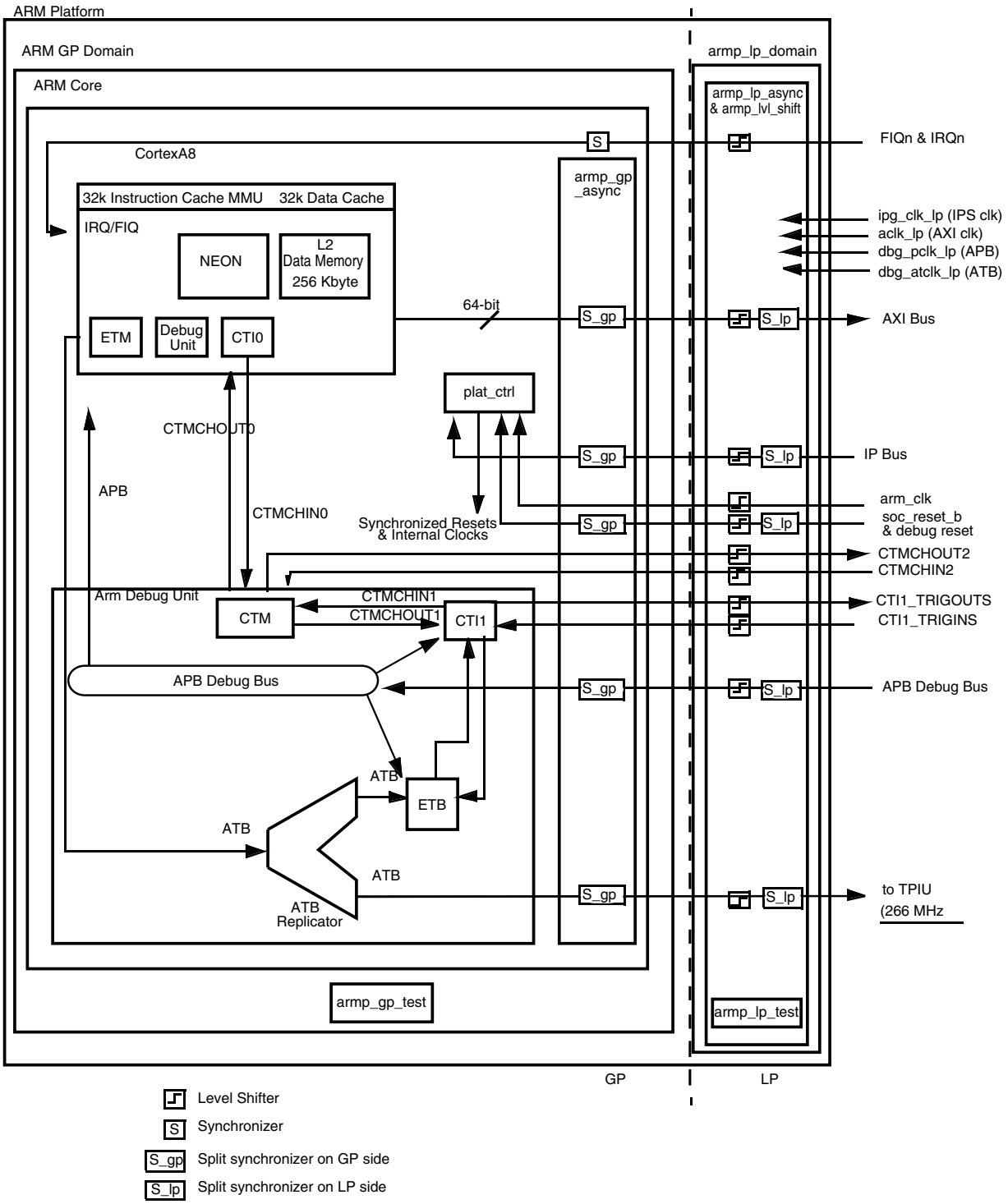


Figure 8-1. ARM platform Block Diagram

8.2.1 Core Platform Sub-Blocks

8.2.1.1 ARM platform

The information presented in this section focuses on design aspects of the ARM platform in the ARM platform subsystem. The ARM platform is ARM's first superscalar processor featuring technology for enhanced code density and performance, NEON™technology for multimedia and signal processing, and Jazelle®RCT (Runtime Compilation Target) technology for efficient support of ahead-of-time and just-in-time compilation of Java and other bytecode languages.

The ARM platform incorporates an integer core that implements the ARMv7-A architecture instruction set. It supports the ARMv7 and Thumb® -2 instruction sets. A NEON sub-block is included to accelerate the performance of multimedia applications. Included Vector Floating Point v3 architecture is IEEE 754 compliant. The processor includes an AMBA® 3 AXI high-performance 64-bit SoC interconnect.

The following are the features of ARM platform:

- The ARM platform's sophisticated pipeline architecture is based on dual, symmetric, in-order issue, 13-stage pipelines with advanced dynamic branch prediction achieving 2.0 DMIPS/MHz. The instruction execute unit consists of two symmetric Arithmetic Logical Unit (ALU) pipelines and the multiply pipeline.
  - In-order, dual-issue, superscalar microprocessor core
  - 13-stage main integer pipeline
  - 10-stage NEON media pipeline for executing NEON and VFP instruction sets
  - Dedicated L2 cache with programmable wait states
  - Global history based branch prediction
- Works in conjunction with a power optimized load store pipeline to deliver 2.0 DMIPS/MHz for power sensitive applications
- ARMv7 architecture compliant including:
  - Thumb-2 technology for greater performance, energy efficiency, and code density
  - NEON signal processing extensions to accelerate media codecs such as H.264 and MP3
  - Jazelle RCT Java-acceleration technology to optimize Just In Time (JIT) and Dynamic Adaptive Compilation (DAC), and reduce memory footprint by up to three times
- Integrated Level 2 Cache
  - Built using standard compiled LP RAMs

    - Sized at 256 Kb
    - Programmable delay
- Optimized Level 1 Caches
    - Customized design for performance and power optimization
    - Sized at 32 Kb Instruction and 32 Kb Data
    - Combine minimal access latency with hash way determination to maximize performance and minimize power consumption.
- Dynamic Branch Prediction
    - Enabled by branch target and global history buffers
    - Achieves 95% accuracy across industry benchmarks.
    - Replay mechanism minimizes miss-predict penalty
- Memory System
    - Single-cycle load-use penalty for access to the L1 cache
    - Hash array in the L1 cache limits activation of the memories to when they are likely to be needed.
    - Direct interface between the integrated, configurable L2 cache and the NEON media unit for data streaming
    - Banked L2 cache design that enables only one bank at a time
- Memory Management Unit (MMU) and separate instruction and data Translation Look-aside Buffers (TLBs) of 32 entries each
- Embedded Trace Macrocell (ETM) support for nonintrusive debug
- ARMv7 debug with watchpoint and breakpoint registers and a 32-bit Advanced Peripheral Bus (APB) interface to the CoreSight debug system.

8.2.1.2 Instruction Fetch

The instruction fetch unit predicts the instruction stream, fetches instructions from the L1 instruction cache, and places the fetched instructions into a buffer for consumption by the decode pipeline. The instruction fetch unit also includes the L1 instruction cache.

8.2.1.3 Instruction Decode

The instruction decode unit decodes and sequences all ARM and Thumb-2 instructions including the debug control coprocessor, CP14, and the system control coprocessor, CP15 instructions.

The instruction decode unit handles the sequencing of:

- Exceptions
- Debug Events
- Reset Initialization

- Memory Built-In Self Test (MBIST) for L1 cache
- Wait-for-interrupt
- Other Unusual Events
- Instruction Cycle Timing

8.2.1.4 Instruction Execute

The instruction execute unit consists of two symmetric Arithmetic Logical Unit (ALU) pipelines and the multiply pipeline. The execute pipelines also perform register write back.

The instruction execute unit:

- executes all integer ALU and multiply operations including flag generation
- generates the virtual addresses for loads and stores and the base write-back value, when required
- supplies formatted data for stores and also forwards data and flags
- processes branches and other changes of instruction stream and evaluates instruction condition codes.

8.2.1.5 Load/Store

The load/store unit encompasses the entire L1 data side memory system and the integer load/store pipeline. This includes the:

- L1 data cache.
- data side TLB.
- integer store buffer.
- NEON store buffer.
- integer load data alignment and formatting.
- integer store data alignment and formatting.

The pipeline accepts one load or store per cycle that can be present in either pipeline 0 or pipeline 1. This gives the processor flexibility when scheduling load and store instructions.

8.2.1.6 L2 Cache

The L2 cache unit includes the L2 cache and the Buffer Interface Unit (BIU). It services L1 cache misses from both the instruction fetch unit and the load/store unit.

8.2.1.7 NEON

The NEON unit includes the full 10-stage NEON pipeline that decodes and executes the NEON media instruction set. The NEON unit includes:

- the NEON instruction queue
- the NEON load data queue
- two pipelines of NEON decode logic
- three execution pipelines for NEON integer instructions
- two execution pipelines for NEON floating-point instructions
- one execution pipeline for NEON and VFP load/store instructions
- the VFP engine for full execution of the VFPv3 data-processing instruction set.

8.2.1.8 Processor Debug Unit

The processor debug unit assists in debugging software running on the processor. In combination with a software debugger program, the debug unit enables debugging the following:

- application software
- operating systems
- hardware systems based on an ARM processor

The debug unit enables:

- stopping program execution
- examining and altering processor and coprocessor states
- examining and altering memory and input/output peripheral states
- restarting the processor core

8.2.1.9 Embedded Trace Marcocell (ETM)

The ETMv3.3 unit is a nonintrusive trace macrocell that filters and compresses an instruction and data trace for use in system debugging and system profiling. The ETM unit has an external interface outside of the processor called the Advanced Trace Bus (ATB) interface.

The ARM platform ETM provides real time instruction trace for the ARM platform. It is designed to be used with the CoreSight Design Kit. Unlike previous versions of the ARM platforms, this ETM is embedded inside the Processor Core.

Real time tracing is controlled by specifying a set of filtering and triggering resources which include address and data comparators, counters and sequencers. Though data trace can not be enabled, the ETM can still trigger based on data values. Also, the ETM can trace data address values.

The ARM platform ETM contains the following main components:

- Core interface
- Trace generation
- Filtering and triggering resources
- Main FIFO
- AMBA 3 ATB interface
- AMBA 3 APB interface

8.2.1.10 Cross Trigger Interface 0 (CTI0)

This block controls the Trigger Interface (TI). The CTI combines and maps the trigger requests, and broadcasts them to all other interfaces on the embedded cross trigger (ECT) as channel events. When the CTI receives a channel event it maps this onto a trigger output. This enables subsystems to cross trigger with each other. The receiving and transmitting of triggers is performed through the TI.

Each CTI has 8 trigger inputs, 8 trigger outputs and 4 sets of 4 channel I/O(s).

8.2.2 Summary of Remaining Platform Components

8.2.2.1 Platform Controller

The Platform Control contains all the miscellaneous logic required for the platform. The main purpose of the Platform Control Module within the ARM platform is to implement a set of control and/or status registers and associated logic for the management of certain platform functions such as internally generated clocks, debug enable/status, low power control, platform version ID, etc.

The Platform Control Module provides these main features:

- Internal clock generation with the ability to programatically set the frequency (divide-by) for each clock independently.
- Ability to force (down-counter) preload for any one or more clocks at anytime independent of preload set.
- Provides a gated clock for the IP Bus asynchronous bridge for power savings.

- Provides 16 bits of General Purpose register bits for routing out to Platform boundary.
- Provides 8 bits of Platform Internal Control register bits for external Platform (SoC) use.
- Provides a set of Low Power control and status bits.
- Control bit to enable Debug along with Debug active status bit.
- NEON activity monitor which can be used in determining when to disable NEON thus saving power.

8.2.2.2 Debug Sub-Blocks

The ARM platform debug blocks are part of the overall Coresight debug system which include the ETB, CTM, CTI1, ATB replicator and APB address decode. It is expected that a DAP block and one or more additional CTI will be included at the SoC level. This section gives a brief overview of the sub-blocks that are implemented within the ARM platform. For details of the full Coresight debug subsystem, please refer to the SoC debug section.

8.2.2.2.1 Embedded Trace Buffer (ETB)

The ETB provides on-chip storage of trace data using 32-bit RAM. The ETB accepts trace data from the ARM ETM through an ATB port (passing through a replicator in between). Providing an on-chip buffer alleviates the pin count, bandwidth, and pad design requirements associated with sending trace data to a debugger directly through package pins in a real-time fashion.

Features:

- 4Kbyte compiled memory for the trace buffer and optionally can be used as a general purpose memory
- Only 32-bit accesses to the 4Kbyte buffer is supported
- AMBA Peripheral Bus programming interface for configuration and memory access

8.2.2.2.2 AMBA Trace Bus (ATB) Replicator

The ATB Replicator enables two trace sinks (ETB and an off platform port generally connected to a Trace Port Interface Unit - TPIU) to be wired together and receive ATB trace data from the same trace source (ETM). There are no programable registers. It takes incoming trace data from a single source (ETM) and replicates it as multiple masters.

The CSREPLICATOR is part of the armp_gp_platform block. Two output master ports are required for this design - one for the on-platform ETB and one for the off-platform TPIU. Placing the TPIU off platform allows future debug trace sources to connect to the TPIU via a FUNNEL, without the need to modify the ARM platform.

8.2.2.2.3 Cross Trigger Interface 1 (CTI1)

This block controls the Trigger Interface (TI). The CTI combines and maps the trigger requests, and broadcasts them to all other interfaces on the ECT as channel events. When the CTI receives a channel event it maps this onto a trigger output. This enables subsystems to cross trigger with each other. The receiving and transmitting of triggers is performed through the TI.

Each CTI has 8 trigger inputs, 8 trigger outputs and 4 sets of 4 channel I/O(s).

8.2.2.2.4 Cross Trigger Matrix - CTM

This block controls the distribution of channel events. It provides channel interfaces to the CTIs. The CTM can also connect to another CTM via a channel interface. This allows multiple CTMs to be connected.

The ARM platform has one CTM inside the platform to handle events between the platform's CTI, the processor's CTI and the rest of the ECT system outside the platform. Placing a CoreSight cross trigger matrix (CSCTM) on platform minimizes the event cycle time between the ETB and the ETM. The handshaking between the CTIs on-platform and the on-platform CTM are not enabled. This requires the ARM's CTI, the on-platform CTI1 and the on-platform CTM to all be in the same clock domain. The on-platform CTM connects to the ECT system via an off-platform CTM and the respective channels require synchronizing and handshaking enabled.

8.2.2.2.5 Advanced Peripheral Bus - APB Debug Bus

The APB originates off platform generally from a Debug Access Port (DAP) block. This bus allows access to the registers in all of the debug blocks which have addressable registers.

8.2.2.3 Asynchronous Wrapper

The ARM platform hard macro contains synchronizers for most signals into and out of the platform where synchronization is required. This means the ARM platform can function asynchronously from the rest of the SoC.

8.2.3 Configuration

There are several configuration options associated with the ARM platform. These are determined at the platform level and selected as follows:

- The L1 cache size is 32 Kb instruction and 32 Kb data
- There is parity on the L1 cache
- There is no error correction on the L1 cache
- The L2 cache size is 256 Kb
- There is no parity on the L2 cache
- There is no error correction on the L2 cache
- There are 2 L2 tag banks and 4 data banks per tag bank
- The AXI data bus width out of the platform is 64-bit

8.2.4 Endian Modes

The ARM platform supports Little Endian mode only.

8.2.5 Bus Interfaces

This section will discuss the major bus interfaces of the ARM platform. The ARM platform has the following bus interfaces:

- AMBA AXI interface
- APB CoreSight interface
- ATB CoreSight interface
- Peripheral Interface (IP Bus)

8.2.5.1 AMBA AXI Interface

The AXI bus interface is the main interface to the system bus. It performs L2 cache fills and noncacheable accesses for both instructions and data. The AXI interface utilizes a 64-bit wide input and output data bus. It also supports multiple outstanding requests on the AXI bus. The AXI bus is allowed to be asynchronous between the platform and SoC domains. This provides maximum flexibility at the SoC level.

8.2.5.2 APB CoreSight Interface

The APB is an AMBA bus used for debugging. The CoreSight interface is the ARM architecture for multi-processor trace and debug. It defines what debug and trace components are required and how they are connected. The platform Debug APB should be connected to the SoC DAP APB mux. The APB is allowed to be asynchronous between the platform and SoC domains. This provides maximum flexibility at the SoC level.

8.2.5.3 ATB CoreSight Interface

The ATB is a trace output bus used for debugging. The CoreSight components are programmed with the DAP using the APB programming bus. Trace is output over the ATB trace bus. The ATB is allowed to be asynchronous between the platform and SoC domains. This provides maximum flexibility at the SoC level.

8.2.5.4 Peripheral Interface (IP Bus)

The IP Bus interface to the platform connects the on-chip registers to the memory map and internal buses at the SoC level. Location (addresses of the registers) is determined at the SoC level. The platform acts as a slave to an SoC based IP Bus controller.

8.3 Memory Map and Register Definition

The memory map for ARM platform specific registers is shown below. This does not include coprocessor registers contained in the ARM platform or the ARM Coresight Debug. For documentation of registers contained in those blocks, please refer to the appropriate technical reference manual.

This register block address space uses five address lines which are fully decoded. The space includes nine registers and 23 unimplemented locations. The unimplemented locations will return an error if they are accessed. Depending on how the SoC decodes the register block, the entire 32-word register block could alias in the larger memory space.

ARM memory map

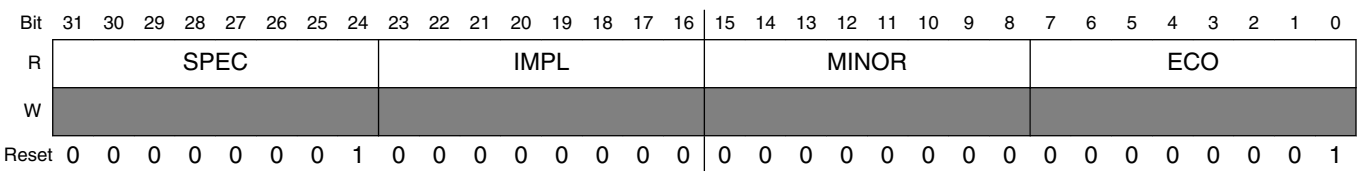
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FA_0000	Platform Version ID (ARM_PVID)	32	R	0100_0001h	8.3.1/513
63FA_0004	General Purpose Control (ARM_GPC)	32	R/W	0004_FF00h	8.3.2/514
63FA_000C	Low Power Control (ARM_LPC)	32	R/W	0000_0000h	8.3.3/515
63FA_0010	NEON Low Power Control (ARM_NLPC)	32	R/W	0000_0000h	8.3.4/516
63FA_0014	Internal Clock Generation Control (ARM_ICGC)	32	R/W	0000_7777h	8.3.5/517
63FA_0018	ARM Memory Configuration (ARM_AMC)	32	R/W	0000_0003h	8.3.6/519
63FA_0020	NEON Monitor Control (ARM_NMC)	32	R/W	000F_F000h	8.3.7/520
63FA_0024	NEON Monitor Status (ARM_NMS)	32	w1c	0000_0000h	8.3.8/521

8.3.1 Platform Version ID (ARM_PVID)

The platform version ID register (ARM_PVID) contains a 32-bit version ID which can be used to link the silicon to a specific version of the platform database. The version ID should match our release label applied to the platform database.

This register can be accessed by a 32-bit secure/non-secure, user/supervisor, read transaction. Writes return an error.

Address: ARM_PVID is 63FA_0000h base + 0h offset = 63FA_0000h



ARM_PVID field descriptions

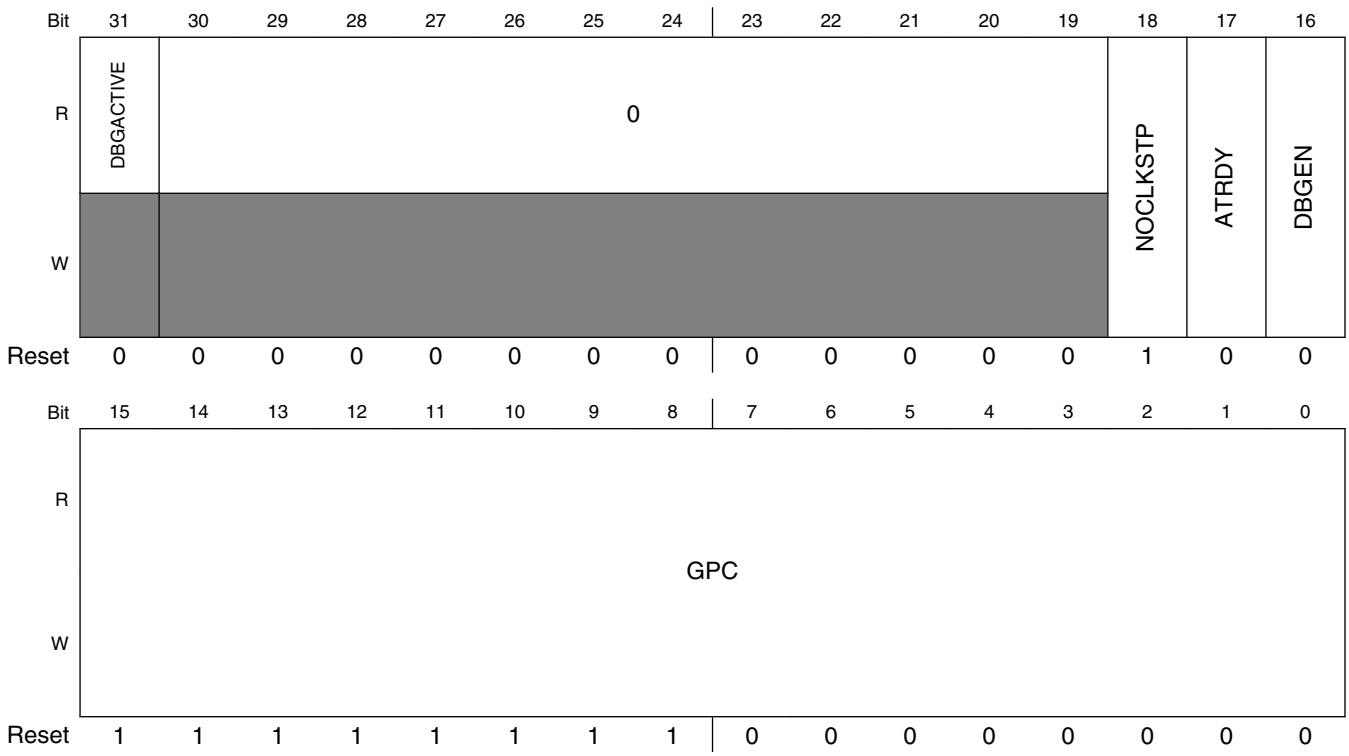
Field	Description
31–24 SPEC	Major architectural or significant spec changes
23–16 IMPL	Implementation changes
15–8 MINOR	Minor changes (bug fixes, I/O changes)
7–0 ECO	ECO changes

8.3.2 General Purpose Control (ARM_GPC)

This register contains 16-bits of general purpose control which drive core platform outputs that can be used within the SoC for general purpose control. This register also contains the DBG_ACTIVE status bit and DBG_EN control bit for entering Debug mode via a software access to this register.

This register can only be accessed by 32-bit secure supervisor transactions.

Address: ARM_GPC is 63FA_0000h base + 4h offset = 63FA_0004h



ARM_GPC field descriptions

Field	Description
31 DBGACTIVE	This bit indicates the status of debug. This allows the user to determine if debug has been enabled either from off platform, via the external DAP_SYS JTAG interface (dbgen_in platform input), or via software accesses to the DBGEN bit in this register. 0 debug is not enabled. Debug clocks are off and debug registers are inaccessible. 1 debug is enabled. Debug clocks are on and the debug system is ready to be used.
30–19 Reserved	This read-only field is reserved and always has the value zero. Reserved
18 NOCLKSTP	This bit is used to control clock-gating within the CORTEX-A8n. Note that this is distinctly different than the higher-level (platform-level) clock-gating/low-power control afforded by the other Plat_Ctrl low-power functionality (dsm_request along with the DSM and DBG_DSM control bits within the Low Power Control register).

Table continues on the next page...

ARM_GPC field descriptions (continued)

Field	Description
	1 All normal clock-gating activity (WFI, NEON, and so on) will be overridden and no clock gating will occur within the CORTEX-A8n. NOCLKSTP, however, has no affect on the higher-level platform clock-gating, and the low-power functions discussed in the LPC register will work as architected. 0 clock-gating within the CORTEX-A8n will be enabled. This is the recommended setting in order to reduce run mode power consumed by the clocking network within the Integer core as well as the Neon co-processor.
17 ATRDY	1 this bit disables the platform boundary ATB interface. In most SoCs this will be connected to a TPIU. This prevents traces to the ETB from stalling due to the ASYNCATB FIFO overflowing. 0 this bit allows the platform boundary ATB interface to be active. In this case, traces to the ETB could stall due to ASYNCATB FIFO becoming full. This may cause the ETM data loss.
16 DBGEN	Debug enable. This allows the user to manually activate clocks within the debug system. This register bit directly controls the platform's dbgen_out output signal which connects to the DAP_SYS to enable all debug clocks. Once enabled, the clocks cannot be disabled except by asserting the disable_trace input of the DAP_SYS.
15–0 GPC	General Purpose Control bits directly control the general_purpose_outs[15:0] output ports of the ARM Platform. These bits will be connected at the SoC level and therefore will control functions decided by the SoC architects, and must be documented by the SoC team.

8.3.3 Low Power Control (ARM_LPC)

This register is used to control entry to low power mode and can only be accessed by 32-bit secure supervisor transactions.

There exists a platform output, dsm_request, that when asserted, requests the SoC to turn off clocks to the platform, and optionally deactivate one or more power supplies. DSM can only be entered after the ARM Platform has executed a WFI instruction, completed all outstanding bus transactions, and all activity at the L2 level has completed. Assertion of either irq_b, fiq_b, or dbg_edbgrq platform inputs will cause the ARM platform to complete the WFI instruction and will result in the negation of dsm_request.

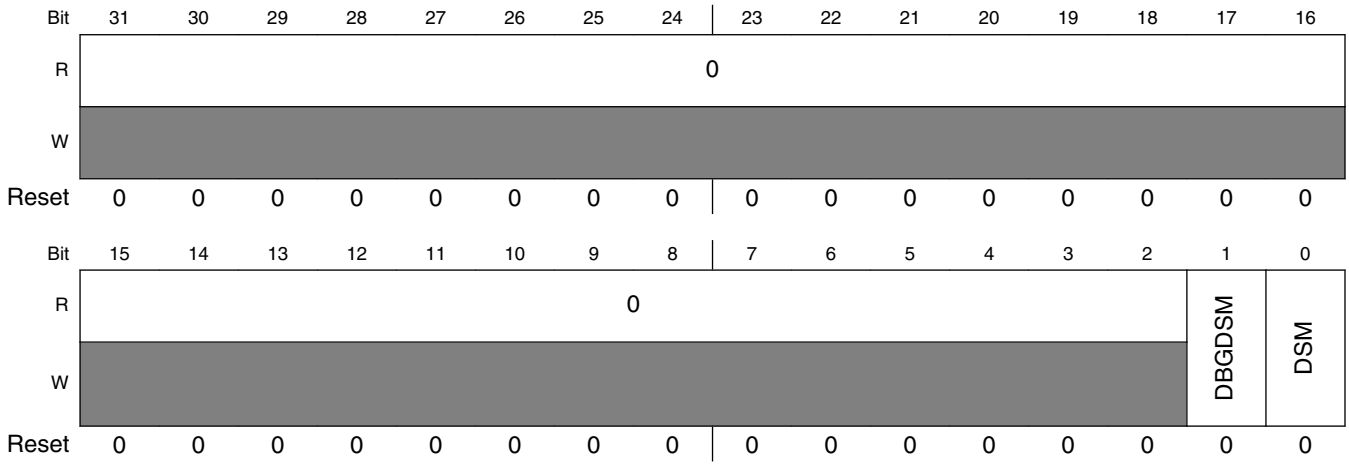
The DSM bit configures the platform to either: block deep sleep mode entry (default), or allow DSM entry when requested.

The DBG_DSM bit configures the platform to block deep sleep mode entry when debug mode is enabled (dbgen_in platform input is asserted high). This will allow the user to ensure deep sleep mode is not entered when debug is enabled.

The NEON_RST bit is used for placing the ARM NEON processor into, or releasing it from, reset.

This register can only be accessed by 32-bit secure supervisor transactions.

Address: ARM_LPC is 63FA_0000h base + Ch offset = 63FA_000Ch



ARM_LPC field descriptions

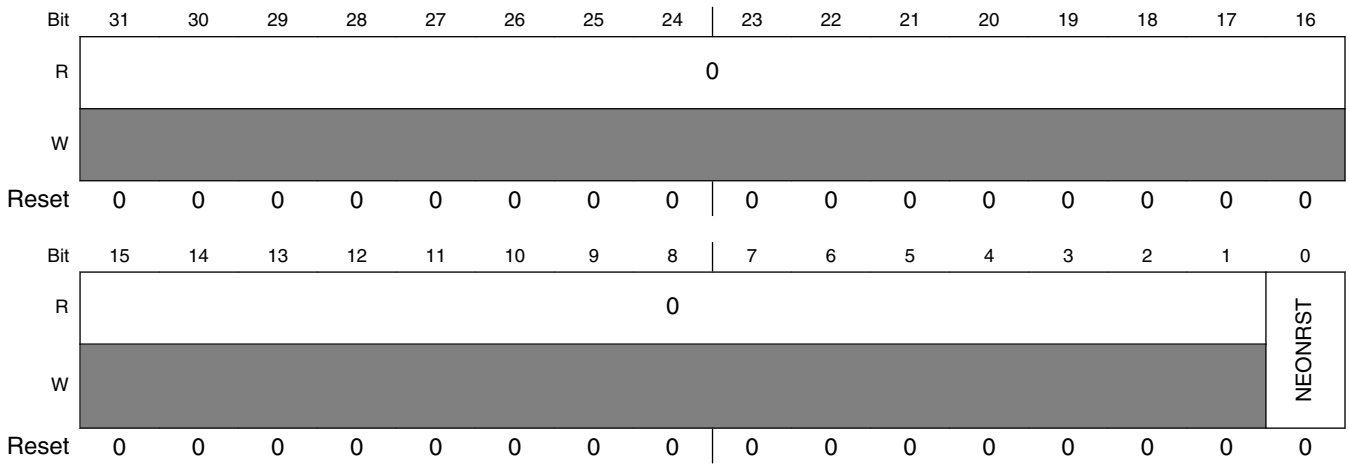
Field	Description
31–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 DBGDSM	Debug Deep Sleep Mode Enable 1 DSM can be entered (dsm_request will not be blocked) regardless if debug is enabled or not. 0 DSM can NOT be entered (dsm_request will be blocked) if debug is enabled
0 DSM	Deep Sleep Mode Enable is used to gate the platform's dsm_request signal, preventing the platform from issuing a deep sleep mode request. 1 DSM can be entered (dsm_request will not be blocked). 0 DSM can NOT be entered (dsm_request will be blocked).

8.3.4 NEON Low Power Control (ARM_NLPC)

This register is used to place the ARM NEON processor into, or releasing it from, reset. Refer to the table below for a description of this control bit.

This register can only be accessed by 32-bit secure supervisor transactions.

Address: ARM_NLPC is 63FA_0000h base + 10h offset = 63FA_0010h



ARM_NLPC field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 NEONRST	Hold the ARM NEON in its reset state in preparation for a low power mode. 1 ARESETNEONn input to the ARM is driven low - placing NEON into reset. 0 ARESETNEONn input to the ARM is driven high - releasing NEON from reset.

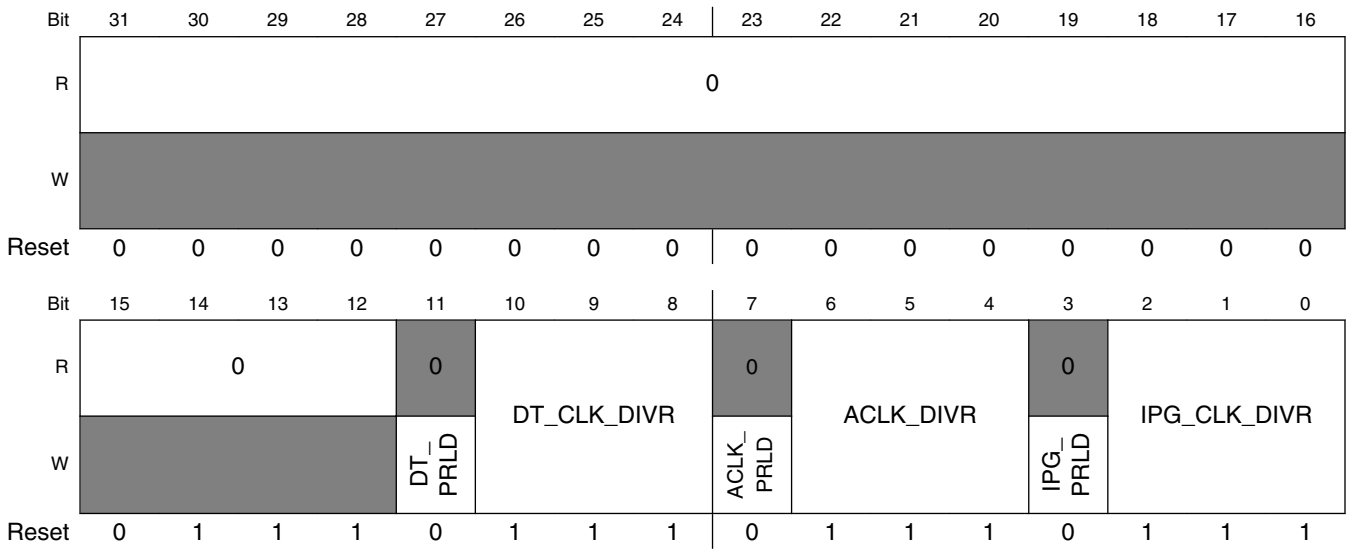
8.3.5 Internal Clock Generation Control (ARM_ICGC)

This register is used to control the clock generation circuitry for all derived clocks used within the ARM platform (ipg_clk, aclk, dbg_atclk, and dbg_pclk). The dbg_pclk clock is generated as a straight divide-by-2 of dbg_atclk.

Note: At reset, all generated clocks are edge aligned and are generated at 8:1. dbg_pclk will require 1 dbg_atclk clock cycle before it begins clocking as a divided dbg_atclk.

This register can only be accessed by 32-bit secure supervisor transactions.

Address: ARM_ICGC is 63FA_0000h base + 14h offset = 63FA_0014h



ARM_ICGC field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value zero. Reserved
11 DT_PRLD	Debug AMBA Trace Bus Clock Down Counter Preload Reads always return 0 0 has no effect. If preload is 0 during a write of a new divide ratio, the down counter will not be loaded with the new value until it reaches 0 (rolls over). 1 A write of 1 will force the DBG_ATCLK down counter to preload on the next arm_clk.
10–8 DT_CLK_DIVR	Debug AMBA Trace Bus Clock Divide Ratio These bits control the clock divide ratio for the debug AMBA trace bus (ATB) and GP async interface clocks. The ATB bus is used to transfer trace messages from the ETM to the ETB or TPIU. When ETM trace data is being transferred over the ATB to the embedded trace buffer (ETB), the ATB is capable of running up to half the ARM Platform frequency. However, when ETM trace data is being transferred to the TPIU to be sent out the trace port, the ATB should be slowed to some appropriate SoC frequency. DT_CLK_DIVR[2:0] = 0b000 results in a 1:1 clock ratio DT_CLK_DIVR[2:0] = 0b001 results in a 2:1 clock ratio DT_CLK_DIVR[2:0] = 0b010 results in a 3:1 clock ratio DT_CLK_DIVR[2:0] = 0b011 results in a 4:1 clock ratio DT_CLK_DIVR[2:0] = 0b100 results in a 5:1 clock ratio DT_CLK_DIVR[2:0] = 0b101 results in a 6:1 clock ratio DT_CLK_DIVR[2:0] = 0b110 results in a 7:1 clock ratio DT_CLK_DIVR[2:0] = 0b111 results in a 8:1 clock ratio
7 ACLK_PRLD	AXI Master Port Clock Down Counter Preload Reads always return 0

Table continues on the next page...

ARM_ICGC field descriptions (continued)

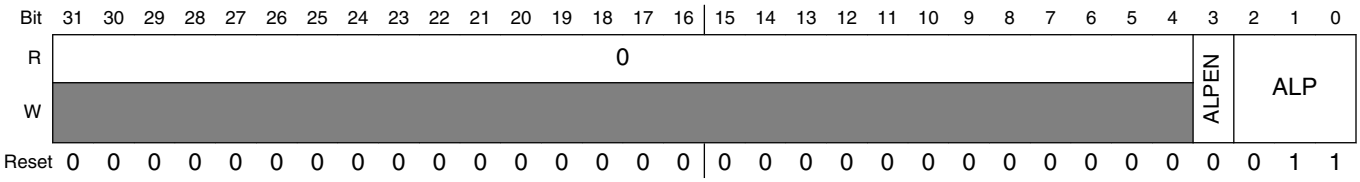
Field	Description
	0 has no effect. If preload is 0 during a write of a new divide ratio, the down counter will not be loaded with the new value until it reaches 0 (rolls over). 1 will force the ACLK down counter to preload on the next arm_clk
6–4 ACLK_DIVR	AXI Master Port Clock Divide Ratio Controls the clock divide ratio for the ARM Platform AXI bus and GP AXI async interface clocks and clock enable. The ratio of clock cycles generated for arm_clk to aclk is: ACLK[2:0] +1 : 1 ACLK_DIVR[2:0] = 0b000 results in a 1:1 clock ratio ACLK_DIVR[2:0] = 0b001 results in a 2:1 clock ratio ACLK_DIVR[2:0] = 0b010 results in a 3:1 clock ratio ACLK_DIVR[2:0] = 0b011 results in a 4:1 clock ratio ACLK_DIVR[2:0] = 0b100 results in a 5:1 clock ratio ACLK_DIVR[2:0] = 0b101 results in a 6:1 clock ratio ACLK_DIVR[2:0] = 0b110 results in a 7:1 clock ratio ACLK_DIVR[2:0] = 0b111 results in a 8:1 clock ratio
3 IPG_PRLD	Platform IP Bus Port Clock Down Counter Preload Reads always return 0 0 has no effect. If preload is 0 during a write of a new divide ratio, the down counter will not be loaded with the new value until it reaches 0 (rolls over). 1 will force the ACLK down counter to preload on the next arm_clk
2–0 IPG_CLK_DIVR	Platform IP Bus Port Clock Divide Ratio Controls the clock divide ratio for the Platform Controller and GP async IP Bus clock and clock enable. The ratio of clock cycles generated for arm_clk to ipg_clk is: IPG_CLK[2:0] +1 : 1 IPG_CLK_DIVR[2:0] = 0b000 results in a 1:1 clock ratio IPG_CLK_DIVR[2:0] = 0b001 results in a 2:1 clock ratio IPG_CLK_DIVR[2:0] = 0b010 results in a 3:1 clock ratio IPG_CLK_DIVR[2:0] = 0b011 results in a 4:1 clock ratio IPG_CLK_DIVR[2:0] = 0b100 results in a 5:1 clock ratio IPG_CLK_DIVR[2:0] = 0b101 results in a 6:1 clock ratio IPG_CLK_DIVR[2:0] = 0b110 results in a 7:1 clock ratio IPG_CLK_DIVR[2:0] = 0b111 results in a 8:1 clock ratio

8.3.6 ARM Memory Configuration (ARM_AMC)

This register is used to configure the ALP inputs to the ARM platform memories. These bits are used to set the leakage configuration bits on the memories.

This register can only be accessed by 32-bit secure supervisor transactions.

Address: ARM_AMC is 63FA_0000h base + 18h offset = 63FA_0018h



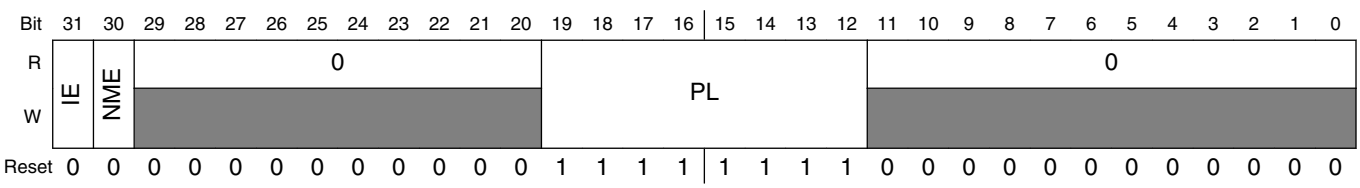
ARM_AMC field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value zero. Reserved
3 ALPEN	ALPEN - ALP Enable is used to activate the ALP bits in this register to override the default memory leakage configuration setting. 1 ALP bits of the memory are over-written with the ALP bits of this register. 0 ALP bits of the memory are driven to the default (reset) value of 3'b000.
2–0 ALP	ALP Memory leakage configuration bits

8.3.7 NEON Monitor Control (ARM_NMC)

This register is used to control the NEON monitor and can only be accessed by 32-bit secure/non-secure supervisor transactions.

Address: ARM_NMC is 63FA_0000h base + 20h offset = 63FA_0020h



ARM_NMC field descriptions

Field	Description
31 IE	Interrupt Enable 1 The monitor interrupt output is enabled and plat_ctrl_nm_irq_b is asserted when the monitor counter expires. 0 The monitor interrupt output is disabled.
30 NME	NEON Monitor Enable
29–20 Reserved	This read-only field is reserved and always has the value zero. Reserved
19–12 PL	Preload value for the upper 8 bits of the 16 bit NEON activity counter.

Table continues on the next page...

ARM_NMC field descriptions (continued)

Field	Description
11–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

8.3.8 NEON Monitor Status (ARM_NMS)

This register is used to read the status of the NEON monitor and can only be accessed by 32-bit secure/non-secure supervisor transactions.

Address: ARM_NMS is 63FA_0000h base + 24h offset = 63FA_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NI								0							
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ARM_NMS field descriptions

Field	Description
31 NI	NEON Idle Status 1 Indicates that the NEON activity counter has expired. plat_ctrl_nm_irq_b will assert if the IE bit is set. 0 Indicates that the NEON activity counter has not expired.
30–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

8.4 Platform Clocks

The clocking strategy of the ARM platform can be summarized by the following bullets:

- The ARM platform will receive one functional clock from the clock control module (CCM), **arm_clk**, and several SoC clocks for boundary synchronization purposes (**arm_clk** can be completely asynchronous from any other SoC clocks).
- The ARM platform, in conjunction with an external CCM, will support dynamic clock frequency scaling.

- Divided clocks based on **arm_clk** will be generated within the platform for clocking buses which cannot operate at the high **arm_clk** frequencies. These divided clock ratios can be configured by on-platform registers.
  - Derived Clocks:
    - ipg_clk: This clock is used for IP Bus logic.
    - aclk: This clock is used for AXI bus logic.
    - dbg_atclk: This clock is used for the Debug AMBA Trace logic.
    - dbg_pclk: This clock is used for the Debug synchronizer logic.
- Two-level clock gating will be employed.
  - Block level clock gating will be used when possible. A specific block's clock will be gated off when it is not in use.
  - Register level clock gating will be used throughout the platform via power design tools.
- **arm_clk** may be turned off for various low-power use cases by an external CCM which will monitor the **DSM_request** output.

8.5 Platform Power Management

The ARM platform contains low voltage logic elements, an asynchronous and level shifted interface, state retention latches, low leakage power switches, and floating node isolation circuits. These elements each serves a specific purpose in the power management scheme implemented in the ARM platform. The power management capabilities of the ARM platform are summarized in this section.

8.5.1 Voltage and Frequency Scaling

The ARM platform contains structures necessary for the independent voltage scaling of the GP supply. Voltage and frequency scaling control systems are implemented at the chip level with the ARM platform acting as an object of the control. Asynchronous Interface Logic and Level-shifting between SoC-logic and ARM-logic have been implemented in the ARM platform to support the voltage and frequency scaling capabilities of the integrated system.

8.5.1.1 Asynchronous Interface Logic

Voltage and frequency scaling is most effective when there is a continuum of available frequencies at which the ARM platform may be clocked. Asynchronous Interface Logic removes the clock ratio dependencies between the ARM platform and the system in which it operates.

8.5.1.2 Level Shifting between SoC-logic and ARM-logic

Independent voltage scaling of the ARM platform requires a distinct power supply for the GP logic. Level shifters exist at the interface of the SoC logic and the ARM logic to allow independent voltage control of the SoC power supply and the ARM power supply.

8.5.2 Power Gating in the ARM platform

To reduce standby leakage power consumption, the ARM platform contains internal power supplies that may be completely power gated during wait for interrupt modes. Entry and exit sequences of power gating modes are completely controlled by programmed operation of the General Power Controller, so this section will only serve to describe what is possible in terms of how the ARM platform may be powered down during wait for interrupt modes.

8.5.2.1 Isolation Circuitry at the ARM platform Interface

In order to power down digital logic correctly, a special interface must exist which will protect the system from floating signals. Such isolation circuitry has been carefully designed in order to allow glitch-free power down modes of the ARM platform.

8.5.2.2 Power Gating the Memory Peripheries

Once the STANDBYWFI mode has been invoked, all of the ARM platform memory peripheries may be power gated by the assertion of both the l1_pwrdwn and the l2_pwrdwn signals. This will significantly reduce the amount of leakage from the GP supply.

8.5.2.3 Retaining the State of the ARM platform Registers

The ARM platform has been implemented using State Retention Power Gating (SRPG) register elements. In order to safely enter a power gated mode that includes power gating of the ARM platform high performance logic gates, a sequenced assertion of signals must occur. Similarly, a sequenced deassertion is necessary when emerging from power gated states. The required sequences are automatically provided to the ARM platform by the General Power Controller.

8.5.2.4 Power Gating the ARM platform High Performance Logic Gates

Once the STANDBYWFI mode has been invoked, the ARM platform high performance logic gates are power gated by assertion and negation of signals by internal Power management sub-blocks. This will turn off a portion of the leakage from the GP supply.

8.5.2.5 Power Gating the L2 Bit Arrays

If the L2 cache has been flushed, the L2 bit array may be power gated during the subsequent STANDBYWFI mode. Once the STANDBYWFI mode has been invoked, the L2 memory bit arrays may be power gated. This will turn off a portion of the leakage from the LP supply.

8.5.2.6 Controlling Power Gating using the General Power Controller (GPC)

Specific details concerning the power gating entry and exit sequence may be obtained from the block guide of the General Power Controller (GPC). This high level summary is intended to provide some overview.

Code must be formed that will take advantage of low workloads by putting the ARM platform into a STANDBYWFI state whenever possible. Prior to entering the STANDBYWFI state, the code should assess several things about the workload and the battery conservation requirements:

- What is the maximum acceptable interrupt service latency?
- What is the expected time to be spent in STANDBYWFI?

Once these time requirements are understood, then Table 8-10 can be used to determine the appropriate STANDBYWFI mode to invoke. The GPC registers can be programmed with the appropriate values to invoke the appropriate STANDBYWFI mode. After GPC programming is complete, the code should execute the STANDBYWFI instruction.

The GPC will enforce the appropriate sequence for entering and exiting the programmed power gated modes.

8.5.2.7 Power Gating the NEON Block

The NEON co-processor in the ARM platform can be put into a state retention mode and powered down to reduce current consumption when it is not needed. The sequence is shown in the flow diagram of Figure 8-10. Each box in the diagram is numbered to correspond to the following more thorough description:

1. User Selects NEON to be Powered Down

   The NEON Coprocessor is enabled out of reset and will remain powered up unless the user decides to power it down. This method will power down NEON after a user programmable time of NEON inactivity. The current state of the NEON is held in SRPG flops. To select this operation:

   The user code will program the amount of time that the NEON needs to be inactive before powerdown in the NEON Monitor Control Register. Effectively, the user programs the upper 8 bits of a 20-bit counter. This counter runs at the arm_clk rate and so the user can program a timeout from 4K to 1M arm_clk rising edges.

   In the same register, the user code enables the NEON Monitor and enables the NEON Monitor Interrupt.

2. Pre-Set Timer Count

   The timer is loaded (automatically by hardware) with the count value chosen by the user.

3. Decision: NEON Instruction Encountered?

   If a NEON instruction is encountered the sequence is aborted until the NEON is no longer in use. As long as NEON Instructions are executing, the NEON status bit will remain in the active state and the counter gets forced with the pre-set count.

4. Decrement Timer

   When no NEON instructions are encountered, the timer will count towards the timeout.

5. Decision: Timed Out?

   If the user selected time has lapsed without any NEON activity, then the hardware will generate an interrupt.

6. Exception Sequence: Power Down NEON

   The interrupt service routine disables the NEON sub-block with the following sequence:

   A. Software must disable access to the NEON unit using the Coprocessor Access Control Register, see c1, Coprocessor Access Control Register on page 3-64 of the ARM TRM. All outstanding NEON instructions retire and all subsequent NEON instruction cause an Undefined instruction exception.

```
MRC p15, 0, <Rd>, c1, c0, 2; Read Coprocessor Access Control Register
BIC <Rd>, <Rd>, #0xF00000; Disable access to CP10 and CP11
MCR p15, 0, <Rd>, c1, c0, 2; Write Coprocessor Access Control Register
```

   B. Activate the NEON output clamps.

   C. Place NEON into state retention.

   D. Prepare the NEON memory for power down keeping the state of its bits.

   E. Separate the NEON power supplies so that the ones that retain state can continue to have power.

   F. Remove power from the NEON power domain.

7. Decision: NEON Instruction Attempted

   The part will continue running regular ARM Platform instructions with the NEON in low power state as long as no NEON instructions are attempted. When a NEON instruction is attempted, an unimplemented instruction exception occurs.

8. Exception Sequence: Power Up NEON

   An exception is taken when a NEON instruction is attempted with the NEON is disabled. The exception routine will determine that this is indeed what occurred, enable the NEON block and wait for the NEON to power up. Returning from the exception will cause the NEON instruction to be restarted and run successfully. This is a more detailed description of the sequence:

   A. Power is turned on to the NEON power domain.

   B. Reconnect the NEON power supplies together.

   C. Restore the NEON memory power maintaining the state of it's saved bits.

   D. Reload the flops with the SRPG saved contents.

E. Release the NEON output clamps.

F. Software must enable access to the NEON unit using the Coprocessor Access Control Register, see c1, Coprocessor Access Control Register on page 3-64 of the ARM TRM.

```
MRC p15, 0, <Rd>, c1, c0, 2; Read Coprocessor Access Control Register
ORR <Rd>, <Rd>, #0xF00000; Enable access to CP10 and CP11
MCR p15, 0, <Rd>, c1, c0, 2; Write Coprocessor Access Control Register
```

G. Return from the exception causing the NEON instruction to be reloaded and executed.

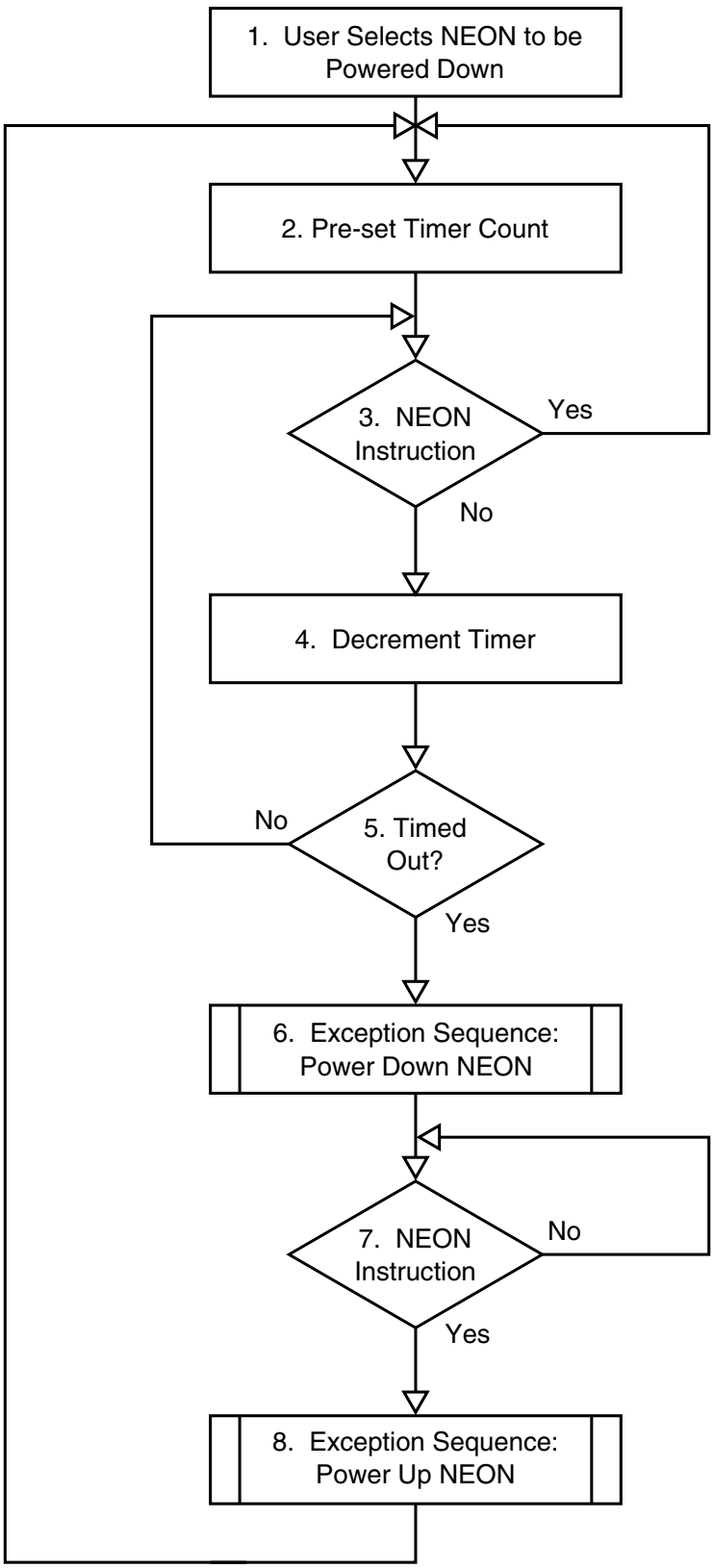


Figure 8-10. Flow Diagram of NEON Power Down Sequence

8.5.3 Modes of Operation

There are several basic low power modes of the ARM platform.

Table 8-10. Modes of Operation of the ARM platform

MODE	Description
RUN	ARM_CLK running, code executing
CLOCKED_WAIT	ARM_CLK running, STANDBYWFI=TRUE
UNCLOCKED_WAIT	ARM_CLK off, STANDBYWFI=TRUE
STOP1	UNCLOCKED_WAIT plus L1 and L2 peripheries power gated
STOP2	UNCLOCKED_WAIT plus L1 and L2 peripheries power gated, ARM_HiP_logic power gated
STOP3	UNCLOCKED_WAIT plus L1 and L2 peripheries power gated, ARM_HiP_logic power gated, L2 cache flushed, and L2 bit arrays power gated

There are several combinations of reduced power in which one of more portions of the platform are in a power saving mode.

Table 8-11. Platform Power Modes

Core and Platform	NEON	L1	L2
Powered Up	Powered Up	Powered Up	Powered Up
Powered Up	Reset (ARM Mode)	Powered Up	Powered Up
State Retention	State Retention	State Retention	State Retention
State Retention	State Retention	Powered Down	State Retention
State Retention	State Retention	State Retention	Powered Down
State Retention	State Retention	Powered Down	Powered Down

Chapter 9
ARM Platform Debug

9.1 Introduction

The purpose of this document is to provide an overview of the ARM platform debug. It will cover the sub-blocks inside the armp_gp_debug block and also the blocks that support the platform debug within the SOC, but external to the ARM platform.

Debug for the ARM platform uses ARM's DK11 CoreSight Debug Kit. The following CoreSight blocks are used: Debug Access Port (DAP) , Embedded Trace Macrocell v 3.3 (ETM), CoreSight replicator (CSREPLICATOR), CoreSight trace port interface unit (CSTPIU), CoreSight embedded trace buffer (CSETB), CoreSight cross trigger interface (CSCTI) and the CoreSight cross trigger matrix (CSCTM). In this introduction, a brief description of each block will be given. For more in depth information, please refer to the CoreSight_DK_TRM and other relative documentation from ARM.

9.1.1 Overview

ARM's CoreSight Design Kit (CSDK) provides a single solution for multi core and bus trace. The CSDK provides the following capabilities for system-wide trace:

- debug and trace visibility of the entire system
- cross triggering support between SOC subsystems
- multi-source trace in a single stream
- higher data compression than previous solutions
- standard programmer's models for standard tools solutions

The CoreSight Design Kit comprises the following main components: Control and access, sources, links and sinks.

- Control and access components configure, access and control the generation of trace. They do not generate, nor process trace data. Examples include the debug access port (DAP) and the embedded cross trigger (ECT) interface.

- Source components generate trace data. Example source components are the ETM and the future AXI trace macrocell (XTM).
- Link components provide connection, triggering and flow of trace data. Link examples include the ATB replicator and the CoreSight Trace funnel.
- Sink components are the end point for trace data on an SOC. Example sinks are the trace port interface unit (TPIU) and the embedded trace buffer (ETB).

The figure below shows the ARM Platform debug block diagram. Further detail of the sub-blocks will be provided in the sections that follow.

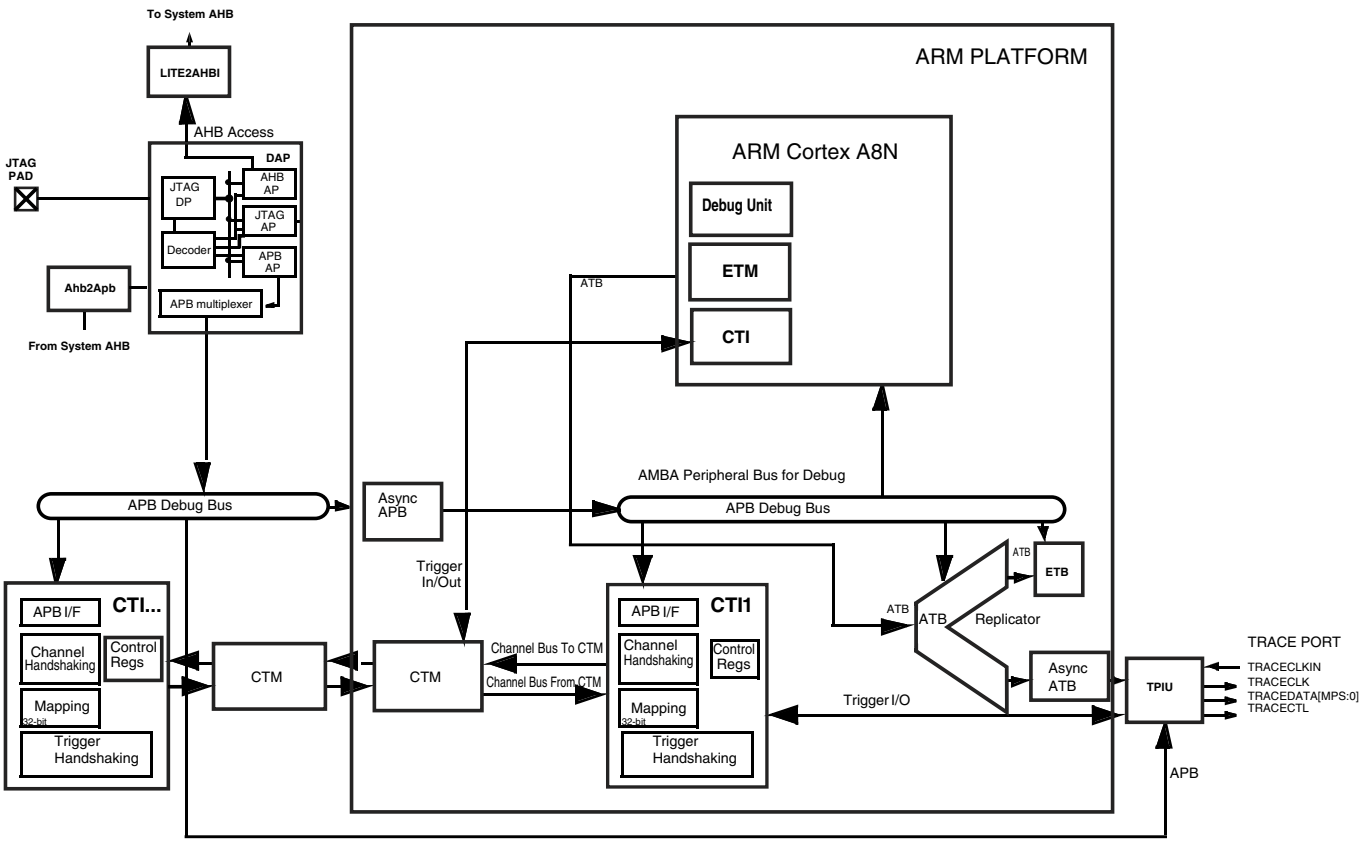


Figure 9-1. Block Diagram

9.1.2 ARM Debug Blocks

This section gives an overview of the Debug blocks used in the ARM platform and also the blocks outside the platform that are included in the CoreSight debug system.

The following blocks are included in the ARM platform debug block: CSETB, CSCTI, CSCTM, CSREPLICATOR and APB address decode logic.

The following blocks are part of the ARM platform, outside the debug block: ETM (embedded in the Cortex), CTI (also embedded in the Cortex), the Processor Debug Unit and two asynchronous bridges - the asyncapb and the asyncatb.

Also included in this block guide is an overview of the DAP and TPIU which reside outside the ARM platform, yet are key components to the debug system.

9.1.2.1 Processor Debug Unit

The processor debug unit assists in debugging software running on the processor. In combination with a software debugger program, the debug unit enables debugging the following:

- application software
- operating systems
- hardware systems based on an ARM processor

The debug unit enables:

- stopping program execution
- examining and altering processor and coprocessor states
- examining and altering memory and input/output peripheral states
- restarting the processor core

There are three ways to debug software running on the processor:

- Halt debug-mode debugging
- Monitor debug-mode debugging
- Trace debugging (ETM) - covered in a different section

9.1.2.1.1 Halting Debug-Mode Debugging

Halting debug-mode debugging is invasive. The processor halts when a debug event, such as a break point, occurs. An external debugger can examine and modify the processor state via the APB while the processor is halted.

9.1.2.1.2 Monitor Debug-Mode Debugging

When a debug event occurs during monitor debug-mode debugging, instead of halting the processor, the processor takes an exception. Special software can then take control to examine or alter the processor state. When execution of a monitor target starts, the state of the processor is preserved in the same manner as all ARM exceptions.

9.1.2.1.3 Programming the Debug Unit

The processor unit is programmed using the APB slave port. Features that can be accessed using the memory-mapped APB registers are:

- instruction address comparators for triggering break points
- data address comparators for triggering watchpoints
- a bidirectional Debug Communication Channel (DCC)
- all other state information associated with the debug unit

9.1.2.2 ARM embedded trace macrocell (ARM ETM)

The ARM ETM provides real time instruction trace for the Cortex A8N processor. It is designed to be used with the CoreSight Design Kit. Unlike previous versions of the ARM platforms, this ETM is embedded inside the ARM Core.

Real time tracing is controlled by specifying a set of filtering and triggering resources which include address and data comparators, counters and sequencers.

NOTE

Though data trace can not be enabled, the ETM can still trigger based on data values. The ETM can also trace data address values.

The ARM ETM contains the following main components:

- Core interface
- Trace generation
- Filtering and triggering resources
- Main FIFO
- AMBA 3 ATB interface
- AMBA 3 APB interface

The ETMv3.3 provides a number of configurations. The table below shows the options implemented in the ETM.

Table 9-1. ETMv3.3 Configurations

Resource Description	Configuration
Instruction trace	Yes
Data address trace	Yes
Data value trace	No
Jazelle trace	-

Table continues on the next page...

Table 9-1. ETMv3.3 Configurations (continued)

Resource Description	Configuration
Address comparator pairs	2
Data comparator	4
Context ID comparators	1
Sequencer	Yes
Start/stop sub-block	Yes
Embedded ICE comparators	0
External inputs	4
External outputs	2
Extended external inputs	49
Extended external input selectors	2
Instrumentation resources	4
FIFO full	No
FIFO full level setting	N/A
Branch broadcasting	Yes
ASIC control register (bits)	8
Data suppression	Yes
Software access to registers	Memory
Readable registers	Yes
FIFO size	128 bytes
Minimum port size	32
Maximum port size	32
Port modes	Dynamic
Asynchronous ATB	Yes
Load pc first	No
Fetch comparisons	No

9.1.2.3 CoreSight Embedded Trace Buffer (CSETB)

The ETB provides on chip storage of trace data. The ARM platform implements a 32-bit 4K RAM.

The ETB accepts trace data from a CoreSight replicator via an ATB write port. The ETB is configurable through an APB port.

Features:

- 4kB compiled memory for the trace buffer. Note: It can no longer be used as a general purpose memory.
- Only 32-bit accesses to the 4kB buffer is supported (Note: RAM size is 4K "bytes" - i.e. 1Kx32)
- Amba Peripheral Bus interface for configuration and memory access

9.1.2.4 CoreSight Replicator (CSREPLICATOR)

The ATB Replicator enables two trace sinks (TPIU and ETB) to be wired together and receive ATB trace data from the same trace source (ETM). There are no programable registers. It takes incoming trace data from a single source (ETM) and replicates it to two master ports.

The CSREPLICATOR is part of the armp_gp_debug block. Two output master ports are required for this design - one for the on-platform ETB and one for the off-platform TPIU. Placing the TPIU off platform allows future debug trace sources to connect to the TPIU through a FUNNEL, without the need to modify the ARM platform. The figure below shows a block diagram for the CSREPLICATOR.

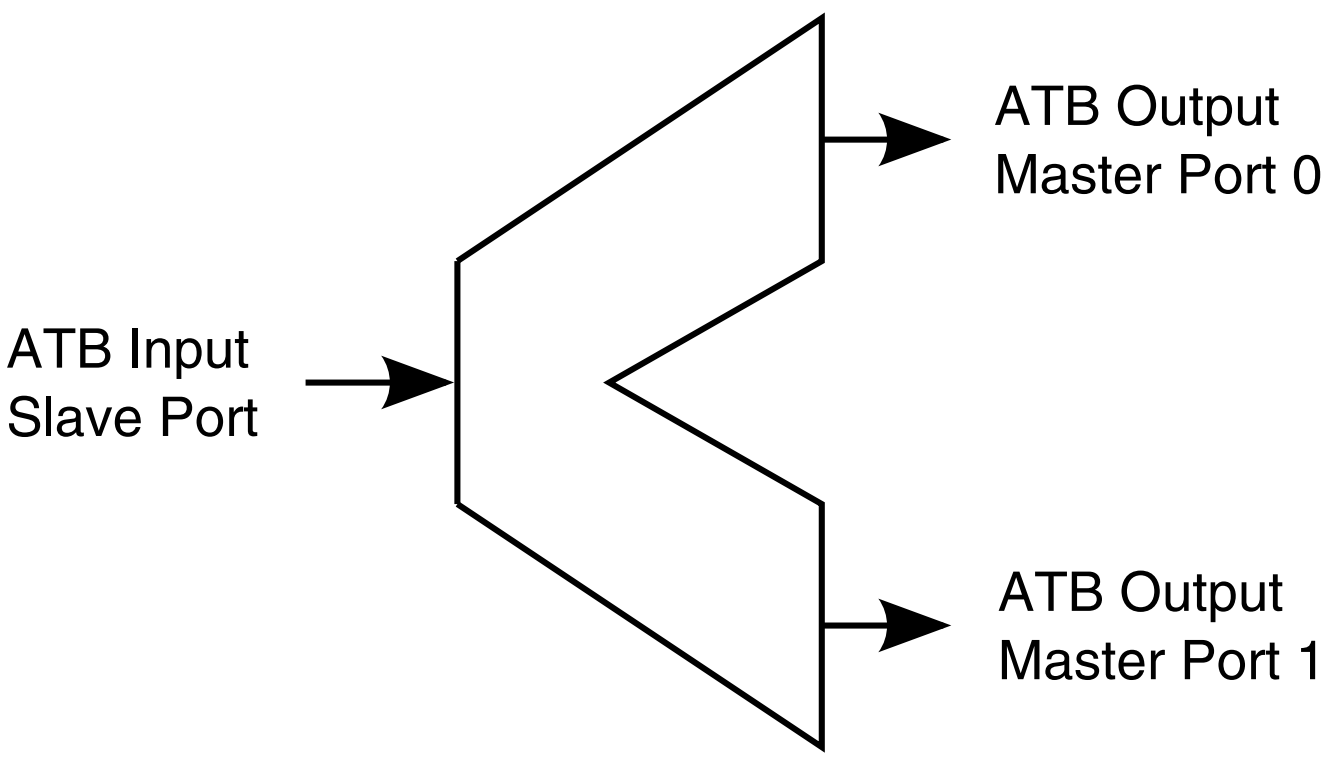


Figure 9-2. CSREPLICATOR Block Diagram

9.1.2.5 CoreSight trace port interface unit (CSTPIU)

The TPIU acts as a bridge between on-chip trace data, ID distinguishable, and a TPA. It receives ATB trace data and sends it off chip via ARM's standard trace signals TRACECLK, TRACEDATA and TRACECTL. The following sub-blocks are included in the TPIU (see the figure below): ATB interface, APB interface, Formatter, Asynchronous FIFO, Register bank, Trace out serializer and a Pattern generator. Further information on the TPIU can be found in ARM's technical reference manuals.

The TPIU is not on the ARM platform and is configurable via the APB.

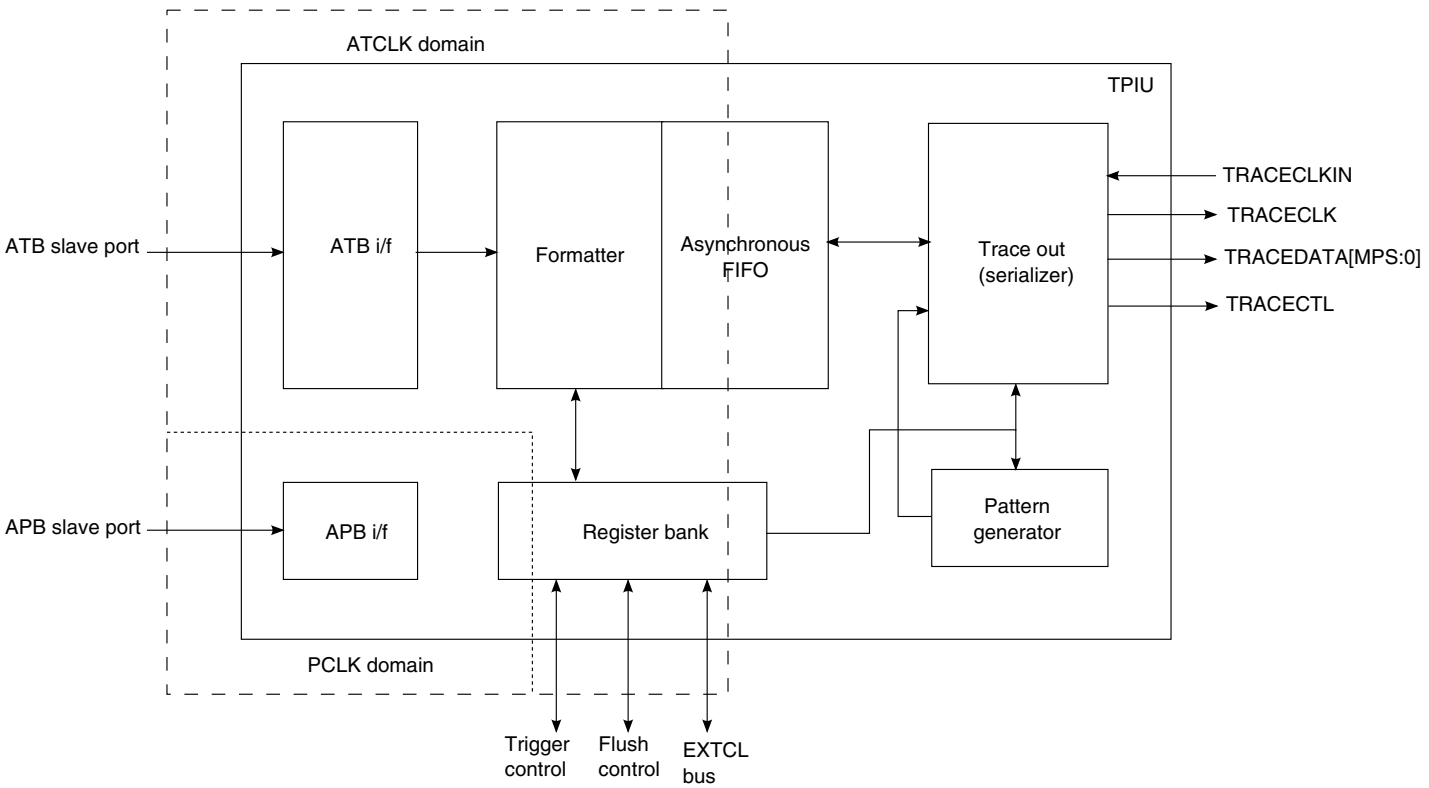


Figure 9-3. TPIU Block Diagram

All signal paths to the pads are subject to wire delays. Special consideration should be taken to re-balance the paths, removing the relative skews between the signals. An extra delay must be added on the TRACECLK path to ensure its rising and falling edge are during the stable part of TRACEDATA and TRACECTL.

9.1.2.6 CoreSight cross trigger interface (CSCTI)

The CSCTI is the CoreSight cross trigger interface component of an Embedded Cross Trigger (ECT) system. The CSCTI combines and maps trigger requests, broadcasting them to all other interfaces on the ECT as channel events. This enables subsystems to cross trigger with each other.

Each CTI has 8 trigger inputs, 8 trigger outputs and 4 sets of 4 channel I/Os. There are also acknowledge signals, configurable on/off, for the trigger and channel outputs. The channels connect to a CoreSight Cross Trigger Matrix (CSCTM). The CSCTM is off-platform, so special consideration should be taken when enabling the synchronizers and handshaking features on both the CSCTI and the CSCTM.

The platform requires one CTI block in addition to the embedded CTI in the ARM core. CTI0 resides in the Core and handles triggers/events from the CSETM and the CORE; CTI1 handles the triggers/events for the TPIU (off platform) and the CSETB. Since the TPIU is outside the platform (asynchronous), the TRIGIN synchronizers are enabled and handshaking is turned on.

The on-platform CTI1 is synchronous and at the same clock speed as the on-platform CTM. This allows the channel handshaking and channel synchronizers to be bypassed.

The figure below shows the recommended connections for CSCTI1.

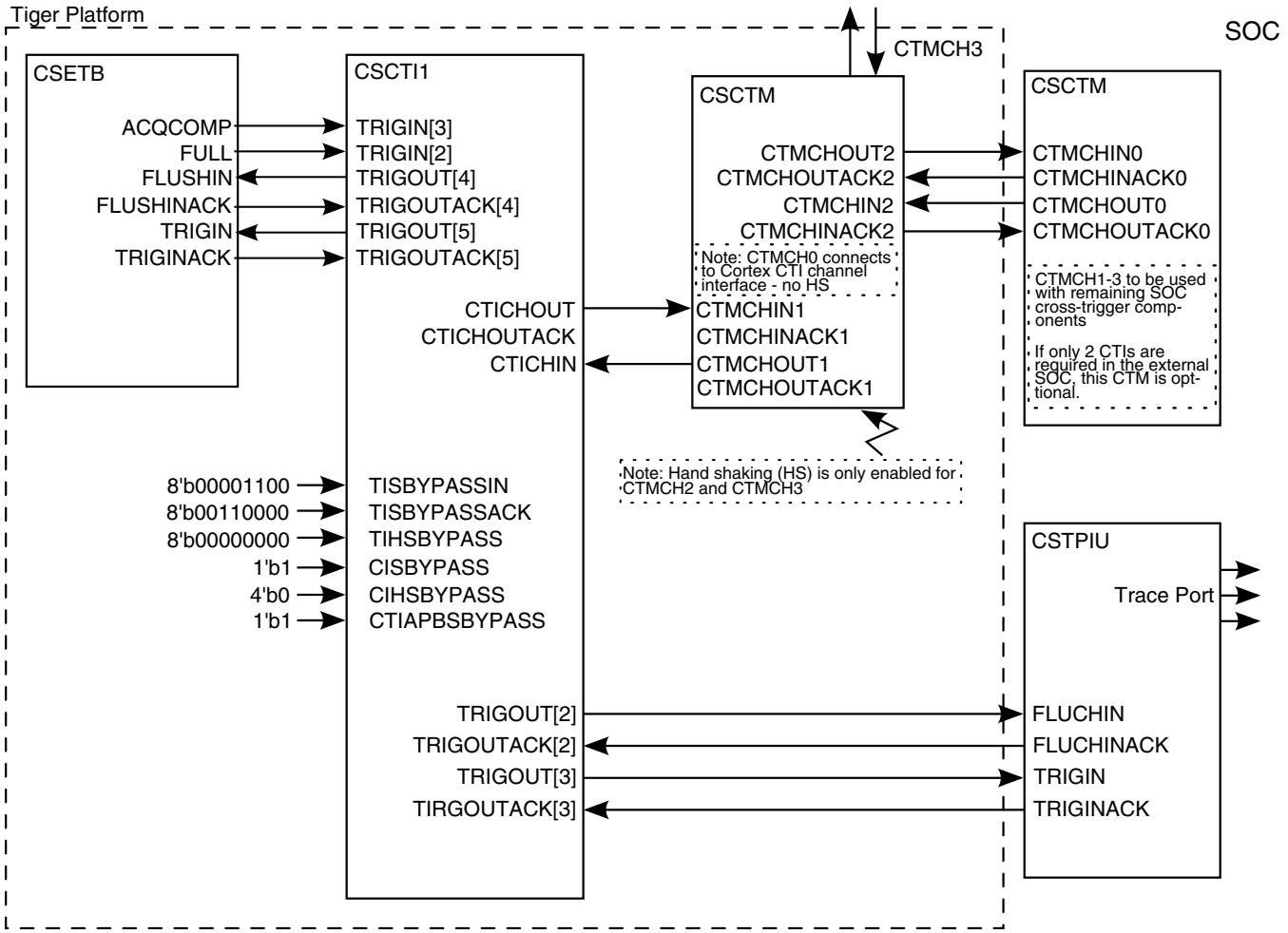


Figure 9-4. CSCTI1 Connections

The SOC CTIs require additional logic to support asynchronous trigger sources/ destinations. The cscti_extended sub-block includes logic, in addition to the CSCTI block, to support triggering across asynchronous boundaries where the source/destination might not have the necessary logic to support asynchronous boundaries.

9.1.2.7 CoreSight Cross Trigger Matrix (CSCTM)

This block controls the distribution of channel events. It provides channel interfaces to the CSCTIs. The CSCTM can also connect to another CSCTM via a channel interface. This allows multiple CSCTMs to be connected.

The ARM has one CSCTM inside the platform to handle events between the platform's CSCTIs and the rest of the ECT system outside the platform. Placing a CSCTM on platform minimizes the event cycle time between the ETB and the ETM. The handshaking between the CTIs on-platform and the on-platform CTM are not enabled.

This requires the Cortex's CTI, the on-platform CTI1 and the on-platform CTM to all be in the same clock domain. The on-platform CTM connects to the ECT system via an off-platform CTM and the respective channels require synchronizing and handshaking enabled.

If only two CTIs are required in the SOC, they can be connected directly to the on-platform CTM eliminating the need for a CTM outside the platform.

More information on the CoreSight ECT system can be found in ARM's Technical Reference manuals.

9.1.2.8 Debug Access Port (DAP)

The DAP provides multiple master driving ports, all accessible via a single external interface port. Components that access the DAP are called Debug Ports (DP) and components that access internal interfaces are called Access Ports.

The DAP provides real time access for the debugger without halting the core to:

- All debug configuration registers
- AMBA system memory and peripheral registers.

The DAP enables debug access to the complete SOC through a number of access ports. Access to the CoreSight Debug APB is enabled through the APB access port (APB-AP). System access can be accomplished via an AHB access port (AHB-AP). An APB multiplexer allows system accesses to debug CoreSight components connected to the APB.

There is also a JTAG access port (JTAG-AP) providing accesses to on-chip JTAG components. The DAP acts as a JTAG master. This feature will not be used for the ARM platform since the core's debug logic is now accessible through the APB.

The figure below shows a block diagram of the DAP.

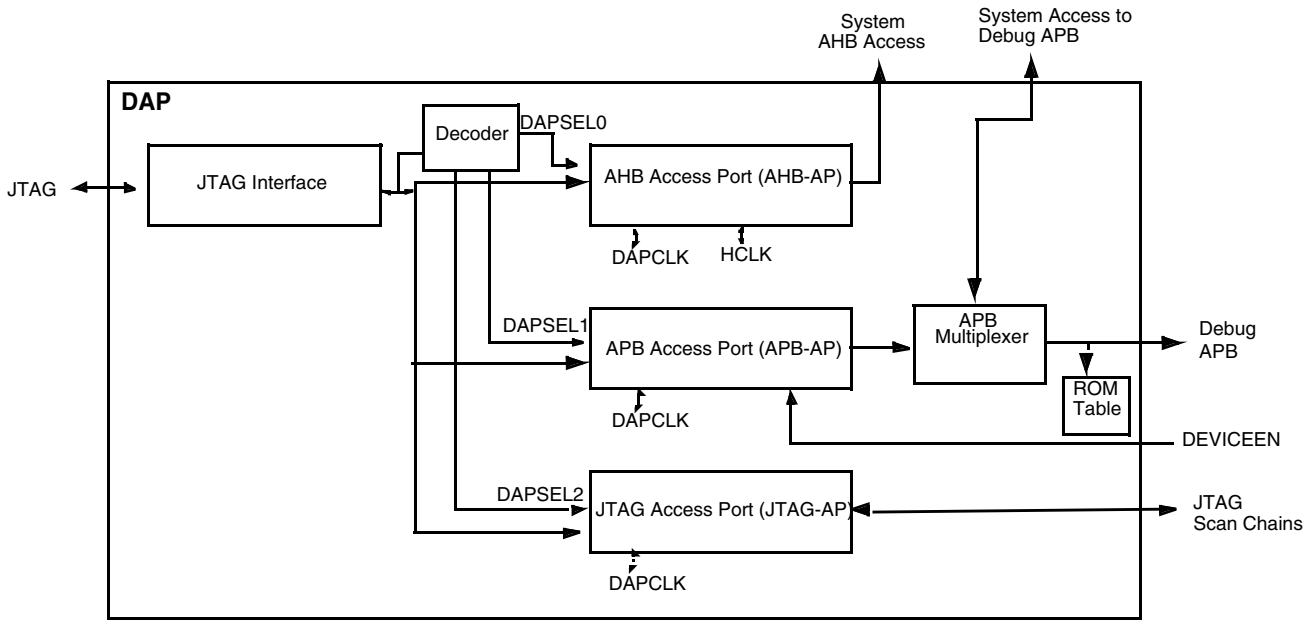


Figure 9-5. DAP Block Diagram

9.1.2.8.1 DAP_SYS

The DAP also requires decode logic for the CoreSight component PSEL signals, an AHB to APB bridge, an AHB lite to AHB bridge, detect logic to enable debug and reset synchronization logic. The dap_sys sub-block wraps these components with the DAP. The figure below shows a block diagram of the DAP_SYS.

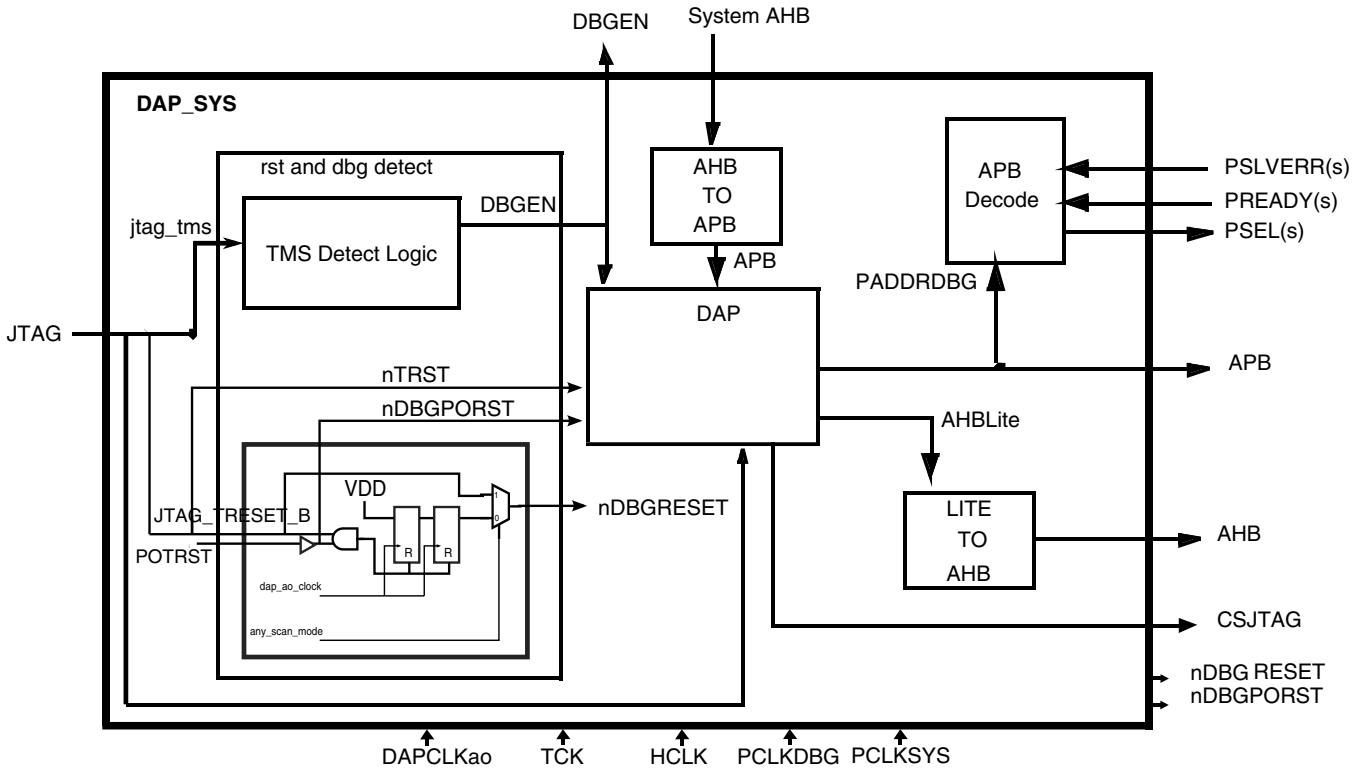


Figure 9-6. DAP_SYS Block Diagram

9.1.3 Modes of Operation

9.1.3.1 ARM Invasive Debug Mode

ARM Invasive Debug mode is usually entered through the JTAG port. Inside the dap_sys there is logic to assert DBGEN high when jtag_tms is active. The user can also invoke DBGEN by writing to a register inside the ARM plat_ctrl sub-block.

It is required that NIDEN is asserted while DBGEN is asserted. Otherwise, CTI functionality is limited when TINIDENSEL is tied to 0. Due to this requirement, NIDEN into the Core and the Debug block is a combination of dbg_niden ored with dbgen_in.

9.1.3.2 ARM Non-Invasive Debug Mode (Real-time Trace)

There are two methods to enter non-invasive debug mode - through JTAG or via memory mapped accesses. For memory mapped accesses, non-invasive debug can be enabled by writing to two secure supervisor registers, one within the Central Security Unit (CSU) and one within the ARM platform.

9.1.3.3 Normal Operating Modes

During normal operating mode, debug is not enabled.

9.1.3.4 Low Power Modes

During deep sleep mode (DSM) all debug is disabled. The platform_ctl sub-block has a bit allocated to overriding DSM when debug is enabled. This bit currently resets to a state disabling DSM while debug is enabled. For more information, refer to the platform control block guide.

The core has an input, DBGNOCLKSTOP (from ARM_GPC[NOCLKSTOP]), allowing the core's debug clocks to stay on while in WFI. The same bit used to override DSM should be connected to DBGNOCLKSTOP (inverted). Another solution is to allocated separate bit for this function in the platform_ctl sub-block. Make note that if the dsm override bit is not set and the platform goes into DSM, it doesn't matter what DBGNOCLKSTOP is set to.

9.2 Memory Map and Register Definition

Each CoreSight component has an allocated 4K. CoreSight components are part of either CoreSight Class or ROM Class. The figure below shows the CoreSight class layout for a 4KB block.

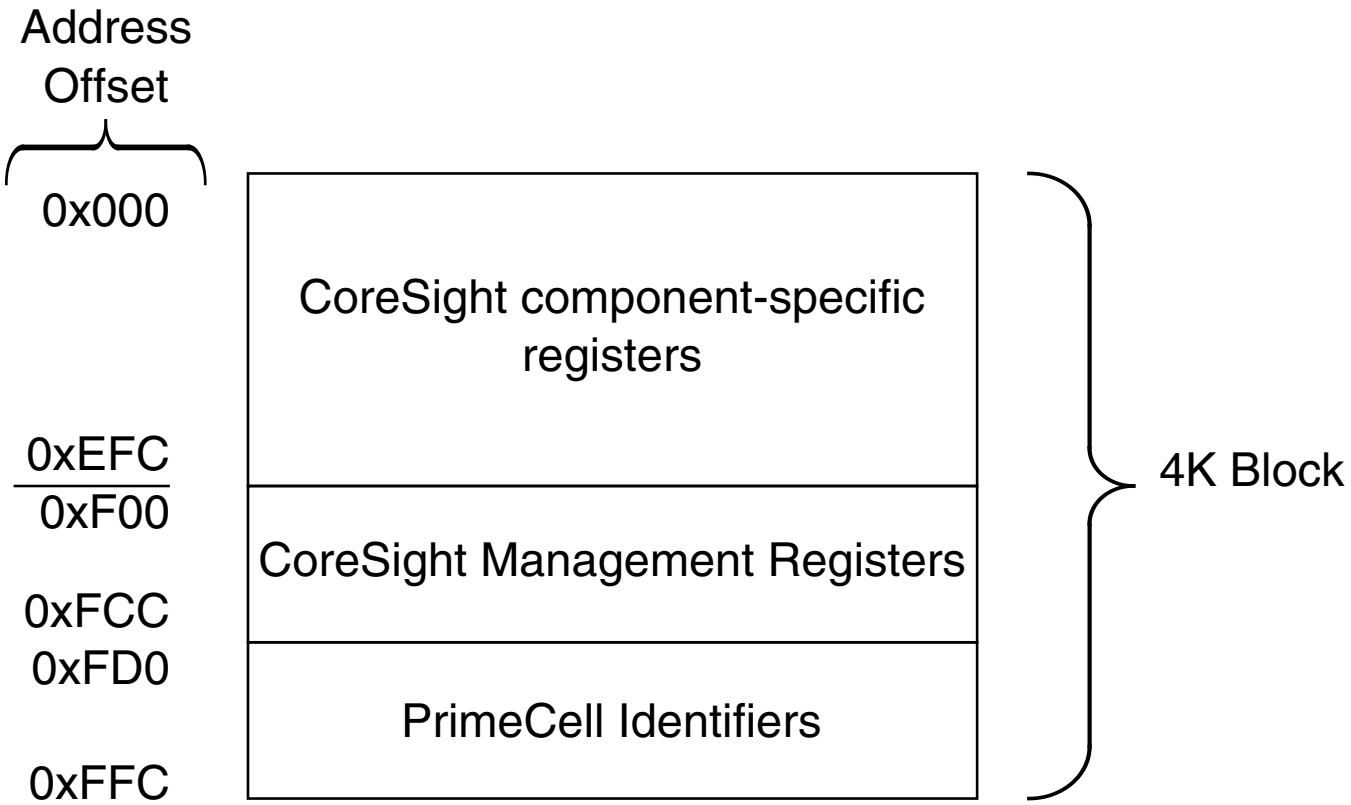


Figure 9-7. CoreSight Component Memory Map

The following table shows an example Debug memory map. The most significant half-word of the address will change based on the chip level memory map.

Table 9-2. Block Memory Map

Start Address	End Address	Size	Target IP	Comments
B0C00000	B0C00FFF	4K	Debug ROM	Located inside DAP
B0C01000	B0C01FFF	4K	ETB	
B0C02000	B0C02FFF	4K	ETM	
B0C03000	B0C03FFF	4K	TPIU	
B0C04000	B0C04FFF	4K	CTI0	On platform
B0C05000	B0C05FFF	4K	CTI1	On platform
B0C06000	B0C06FFF	4K	CTI2	Off platform
B0C07000	B0C07FFF	4K	CTI3	Off platform

9.2.1 Register Summary

This section summarizes the CoreSight Component registers in a table format. The tables are divided by Components. For a more detailed description of each register, refer to the CoreSight_DK_TRM.

The conventions in the table below serve as a key for the register summary.

Table 9-3. Register Conventions

Register Field Types	
RO	Read only. Writing this bit has no effect.
WO	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
RAZ	Read-as-zero
WI	Writes ignored
Self-clearing bit	Writing a one has some effect on the Block, but it always reads as zero. (Previously designated slfclr)

9.2.1.1 DAP JTAG DP Registers

Accessing a JTAG DP register depends on both the instruction register (IR) value of the DAP access and the address field of the DAP access.

The following table shows a register summary table for the DAP DP. Detailed register descriptions can be found inside the CoreSight_DK_TRM provided by ARM.

Table 9-4. DAP JTAG DP Registers

Address Field	IR Contents	Type	Register Name
1	IDCODE	RO	Identification Control Register
0x0	DAPACC	RAZ/WI	Reserved
0x4	DAPACC	R/W	DP Control/Status Register
0x8	DAPACC	R/W	Select Register
0x0	ABORT	WO	DAP Abort Register
0x4-0xC	ABORT	2	Reserved

1. There is no address associated with the ID code register. For more information, see the CoreSight_DK_TRM.
2. The value read on the abort scan chain is unpredictable. The result of accessing the abort scan chain without the address set to 0x0 is unpredictable.

9.2.1.2 DAP ROM Register Summary

The ROM table stores the locations of components on the debug APB. The DAP ROM is read only and configurable through the DAPROM.v and DapRomDefs.v RTL files. Writes are ignored.

The following table shows a register summary table for the DAP Rom. Detailed descriptions can be found inside the CoreSight_DK_TRM provided by ARM.

Table 9-5. DAP ROM Registers

Offset	Name	Type	Description
0xFD0	Peripheral ID4	RO	See CoreSight_DK_TRM
0xFD4	Peripheral ID5	RAZ	Reserved
0xFD8	Peripheral ID6	RAZ	Reserved
0xFDC	Peripheral ID7	RAZ	Reserved
0xFE0	Peripheral ID0	RO	See CoreSight_DK_TRM
0xFE4	Peripheral ID1	RO	See CoreSight_DK_TRM
0xFE8	Peripheral ID2	RO	See CoreSight_DK_TRM
0xFEC	Peripheral ID3	RO	See CoreSight_DK_TRM
0xFF0	Component ID0	RO	Set to 0x0D
0xFF4	Component ID1	RO	Set to 0x10
0xFF8	Component ID2	RO	Set to 0x05
0xFFC	Component ID3	RO	Set to 0xB1

9.2.1.3 Processor Debug Unit Register Summary

Most of the debug unit registers are accessible through the APB. There are some registers that can also be accessed through a coprocessor interface - CP14.

By default, CP14 registers can be accessed from a non-privileged mode. However, the processor can be programmed to disable user access modes using bit [12] of the Debug Status and Control Register (DSCR). For more information on the DSCR and access to CP14 registers, refer to the CortexA8 TRM document.

9.2.1.3.1 Coprocessor Registers Summary

The table below shows the valid debug instructions for accessing the debug registers.

Table 9-6. CP14 Debug Registers Summary

Instruction	Mnemonic	Description
MRC p14, 0, <Rd>, c0, c0, 0	DIDR	Debug Identification Register
MRC p14, 0, <Rd>, c1, c0, 0	DRAR	Debug ROM Address Register
MRC p14, 0, <Rd>, c2, c0, 0	DSAR	Debug Self Address Register
MRC p14, 0, <Rd>, c0, c5, 0 STC p14, c5, <addressing mode>	DTRRX	Data Transfer Register - Receive
MCR p14, 0, <Rd>, c0, c5, 0 LDC p14, c5, <addressing mode>	DTRTX	Data Transfer Register - Transmit
MRC p14, 0, <Rd>, c0, c1, 0 MRC p14, 0, PC, c0, c1, 0	DSCR	Debug Status and Control Register

9.2.1.3.2 Memory Mapped Registers Summary

The table below shows a complete list of memory mapped registers accessible using the APB slave port.

Table 9-7. Debug Unit APB Accessible Registers Summary

Offset	Register Number	Mnemonic	Type	Description
0x000	c0	DIDR	R	CP14 c0, Debug ID Register
0x004-0x014	c1-c5	-	R	RAZ
0x018	c6	WFAR	R/W	Watchpoint Fault Address Register
0x01C	c7	VCR	R/W	Vector Catch Register
0x020	c8	-	R	RAZ
0x024	c9	ECR	R/W	Event Catch Register
0x028	c10	DSCCR	R/W	Debug State Cache Control Register
0x02C	c11	-	R	RAZ
0x030-0x07C	c12-c31	-	R	RAZ
0x080	c32	DTRRX	R/W	Data Transfer Register - Receive
0x084	c33	ITR	W	Instruction Transfer Register
0x088	c34	DSCR	R/W	CP14 c1, Debug Status and Control Register
0x08C	c35	DTRTX	R/W	Data Transfer Register - Transmit
0x090	c36	DRCR	W	Debug Run Control Register
0x094-0x09C	c37-c63	-	R	RAZ
0x100-0x114	c64-c69	BVR	R/W	Breakpoint Value Registers
0x118-0x13c	c70-c79	-	R	RAZ
0x140-0x154	c80-c85	BCR	R/W	Breakpoint Control Register

Table continues on the next page...

Table 9-7. Debug Unit APB Accessible Registers Summary (continued)

Offset	Register Number	Mnemonic	Type	Description
0x158-0x17C	c86-c95	-	R	RAZ
0x180-0x184	c96-c97	WVR	R/W	Watchpoint Value Register
0x188-0x1BC	c97-c111	-	R	RAZ
0x1C0-0x1C4	c112-c113	WCR	R/W	Watchpoint Control Registers
0x1C8-0x1FC	c114-c127	-	R	RAZ
0x200-0x2FC	c128-c191	-	R	RAZ
0x300	c192	OSLAR	W	Operating System Lock Access Register
0x304	c193	OSLSR	R	Operating System Lock Status Register
0x308	c194	OSSRR	R/W	Operating System Save and Restore Register
0x30C	c195	-	R	RAZ
0x310	c196	PRCR	R/W	Device Power Down and Reset Control Register
0x314	c197	PRSR	R	Device Power Down and Reset Status Register
0x318-0x7FC	c198-c511	-	R	RAZ
0x800-0x8FC	c512-575	-	R	RAZ
0x900-0xCFC	c576-c831	-	R	RAZ
0xD00-0xFFC	c832-c1023	-	-	Management Register

9.2.1.4 ETB Register Summary

The ETB registers are summarized in the table below. Detailed information can be found in ARM's CoreSight_DK_TRM document.

Table 9-8. ETB Registers

Offset	Name	Type	Description
0x004	RAM Depth Register, RDP	RO	See CoreSight_DK_TRM
0x010	RAM Read Data Register, RRD	RO	See CoreSight_DK_TRM
0x014	RAM Read Pointer Register, RRP	R/W	See CoreSight_DK_TRM
0x00C	Status Register, STS	RO	See CoreSight_DK_TRM
0x018	RAM Write Pointer Register, RWP	R/W	See CoreSight_DK_TRM
0x01C	Trigger Counter, TRG	R/W	See CoreSight_DK_TRM
0x020	Control Register, CTL	R/W	See CoreSight_DK_TRM
0x024	RAM Write Data, RWD	WO	See CoreSight_DK_TRM

Table continues on the next page...

Table 9-8. ETB Registers (continued)

Offset	Name	Type	Description
0x300	Formatter and Flush Status FFSR	RO	See CoreSight_DK_TRM
0x304	Formatter and Flush Control FFCR	R/W	See CoreSight_DK_TRM
0xEE0	Integration Register ITMISCOP0	WO	See CoreSight_DK_TRM
0xEE4	Integration Register ITTRFLINACK	WO	See CoreSight_DK_TRM
0xEE8	Integration Register ITTRFLIN	RO	See CoreSight_DK_TRM
0xEEC	Integration Register ITATBDATA0	RO	See CoreSight_DK_TRM
0xEF0	Integration Register ITATBCTR2	WO	See CoreSight_DK_TRM
0xEF4	Integration Register ITATBCTR1	RO	See CoreSight_DK_TRM
0xEF8	Integration Register ITATBCTR0	RO	See CoreSight_DK_TRM
0xF00	Integration Mode Control Register	R/W	See CoreSight_DK_TRM
0xFA0	Claim Tag Set Register	R/W	See CoreSight_DK_TRM
0xFA4	Claim Tag Clear Register	R/W	See CoreSight_DK_TRM
0xFB0	Lock Access Register	WO	See CoreSight_DK_TRM
0xFB4	Lock Status Register	RO	See CoreSight_DK_TRM
0xFB8	Authentication Status Register	RO	See CoreSight_DK_TRM
0xFC8	Device ID	RO	See CoreSight_DK_TRM
0xFCC	Device Type Identifier Register	RO	See CoreSight_DK_TRM
0xFD0	Peripheral ID4	RO	See CoreSight_DK_TRM
0xFD4	Peripheral ID5	RAZ	Reserved
0xFD8	Peripheral ID6	RAZ	Reserved
0xFDC	Peripheral ID7	RAZ	Reserved
0xFE0	Peripheral ID0	RO	See CoreSight_DK_TRM
0xFE4	Peripheral ID1	RO	See CoreSight_DK_TRM
0xFE8	Peripheral ID2	RO	See CoreSight_DK_TRM

Table continues on the next page...

Table 9-8. ETB Registers (continued)

Offset	Name	Type	Description
0xFEC	Peripheral ID3	RO	See CoreSight_DK_TRM
0xFF0	Component ID0	RO	Set to 0x0D
0xFF4	Component ID1	RO	Set to 0x10
0xFF8	Component ID2	RO	Set to 0x05
0xFFC	Component ID3	RO	Set to 0xB1

9.2.1.5 ETM Register Summary

The ETM registers are described in the table below. A more detailed description of the ETM registers can be found in ARM's ETM_ARCHITECTURE_SPEC document.

Table 9-9. ETM Register Summary

Offset	Name	Type	Description
0x00	ETM Control	R/W	See ETMv3.3 Architecture Specification
0x04	ETM Configuration Control	RO	See ETMv3.3 Architecture Specification
0x08	Trigger Event	WO [1]	See ETMv3.3 Architecture Specification
0x0C	ASIC Control	WO[1]	See ETMv3.3 Architecture Specification
0x10	ETM Status	R/W	See ETMv3.3 Architecture Specification
0x14	System Configuration	RO	See ETMv3.3 Architecture Specification
0x18	Trace Start/Stop Resource Control	WO[1]	See ETMv3.3 Architecture Specification
0x1C	Trace Enable Control 2	WO[1]	See ETMv3.3 Architecture Specification
0x20	TraceEnable	WO[1]	See ETMv3.3 Architecture Specification
0x24	TraceEnable Control 1	WO[1]	See ETMv3.3 Architecture Specification
0x30	ViewData Event	WO[1]	See ETMv3.3 Architecture Specification
0x34	ViewData Control 1	WO[1]	See ETMv3.3 Architecture Specification
0x38	ViewData Control 2	WO[1]	See ETMv3.3 Architecture Specification
0x3C	ViewData Control 3	WO[1]	See ETMv3.3 Architecture Specification
0x40-0x7C	Address Comparator Value 1-16	WO[1]	See ETMv3.3 Architecture Specification
0x80-0xBC	Address Access Type 1-16	WO[1]	See ETMv3.3 Architecture Specification
0xC0-0xFC	Data Comparator Value 1-16	WO[1]	See ETMv3.3 Architecture Specification
0x40-0x4F	Data Comparator Mask 1-16	WO[1]	See ETMv3.3 Architecture Specification

Table continues on the next page...

Table 9-9. ETM Register Summary (continued)

Offset	Name	Type	Description
0x50-0x53	Counter Reload Value 1-4	WO[1]	See ETMv3.3 Architecture Specification
0x54-0x57	Counter Enable 1-4	WO[1]	See ETMv3.3 Architecture Specification
0x58-0x5B	Counter Reload Event 1-4	WO[1]	See ETMv3.3 Architecture Specification
0x5C-0x5F	Counter Value 1-4	R/W	See ETMv3.3 Architecture Specification
0x60-0x65	Sequencer State Transition Event	WO[1]	See ETMv3.3 Architecture Specification
0x66	-	-	Reserved
0x67	Current Sequencer State	R/W	See ETMv3.3 Architecture Specification
0x68-0x6B	External Output Event 1-4	WO[1]	See ETMv3.3 Architecture Specification
0x6C-0x6E	Context ID Comparator Value	WO[1]	See ETMv3.3 Architecture Specification
0x6F	Context ID Comparator Mask	WO[1]	See ETMv3.3 Architecture Specification
0x70-0x77	Implementation specific	WO[1]	See ETMv3.3 Architecture Specification
0x78	Synchronization Frequency	WO[1]	See ETMv3.3 Architecture Specification
0x79	ETM ID	RO	See ETMv3.3 Architecture Specification
0x7A	Configuration Code Extension	RO	See ETMv3.3 Architecture Specification
0x7B	Extended External Input Selection	WO[1]	See ETMv3.3 Architecture Specification
0x7C	Trace Start/Stop Embedded ICE Control	WO[1]	See ETMv3.3 Architecture Specification
0x7D	Embedded ICE Behavior Control	WO[1]	See ETMv3.3 Architecture Specification
0x7E-0x7F	-	-	Reserved
0x080	CoreSight Trace ID	R/W	See ETMv3.3 Architecture Specification
0x81-0xBF	-	-	Reserved
0xC0	OS Lock Access	WO	See ETMv3.3 Architecture Specification
0xC1	OS Lock Status	RO	See ETMv3.3 Architecture Specification
0xC2	OS Save/Restore	R/W	See ETMv3.3 Architecture Specification

Table continues on the next page...

Table 9-9. ETM Register Summary (continued)

Offset	Name	Type	Description
0xC3-0xCF	-	-	Reserved
0x380- 0x3BF	Integration registers	-	Reserved for Implementation-defined topology detection and integration registers.
0xF00	Integration Mode Control	R/W	See ETMv3.3 Architecture Specification
0xFA0	Claim Tag Set	R/W	See ETMv3.3 Architecture Specification
0xFA4	Claim Tag Clear	R/W	See ETMv3.3 Architecture Specification
0xFB0	Lock Access	WO	See ETMv3.3 Architecture Specification
0xFB4	Lock Status	RO	See ETMv3.3 Architecture Specification
0xFB8	Authentication Status	RO	See ETMv3.3 Architecture Specification
0xFC8	Device Configuration	RO	See ETMv3.3 Architecture Specification
0xFCC	Device Type	RO	See ETMv3.3 Architecture Specification
0xFD0	Peripheral ID4	RO	See ETMv3.3 Architecture Specification
0xFD4	Peripheral ID5	RO	See ETMv3.3 Architecture Specification
0xFD8	Peripheral ID6	RO	See ETMv3.3 Architecture Specification
0xFDC	Peripheral ID7	RO	See ETMv3.3 Architecture Specification
0xFE0	Peripheral ID0	RO	See ETMv3.3 Architecture Specification
0xFE4	Peripheral ID1	RO	See ETMv3.3 Architecture Specification
0xFE8	Peripheral ID2	RO	See ETMv3.3 Architecture Specification
0xFEC	Peripheral ID3	RO	See ETMv3.3 Architecture Specification
0xFF0	Component ID0	RO	See ETMv3.3 Architecture Specification
0xFF4	Component ID1	RO	See ETMv3.3 Architecture Specification
0xFF8	Component ID2	RO	See ETMv3.3 Architecture Specification
0xFFC	Component ID3	RO	See ETMv3.3 Architecture Specification

1. In ETMv3.1 and later, register is read/write if bit [11] of the ETM Configuration Code Extension Register (0x7A) is set to b1.

9.2.1.6 CSCTI Register Summary

The CSCTI registers are described in the table below. A more detailed description of the CSCTI registers can be found in ARM's CoreSight_DK_TRM document.

The table describes the location by an offset.

Table 9-10. CSCTI Register Summary

Offset	Name	Type	Width	Reset value	Description
0x000	CTICONTROL	R/W	1	0x0	See ARM CoreSight_DK_TRM
0x010	CTIINTACK	WO	8	-	See ARM CoreSight_DK_TRM
0x014	CTIAPPSET	R/W	4	0x0	See ARM CoreSight_DK_TRM
0x018	CTIAPPCLEAR	WO	?	0x0	See ARM CoreSight_DK_TRM
0x01C	CTIAPPPULSE	WO	4	0x0	See ARM CoreSight_DK_TRM
0x020-0x03C	CTIINEN	R/W	4	0x00	See ARM CoreSight_DK_TRM
0x0A0-0x0BC	CTIOUTEN	R/W	4	0x00	See ARM CoreSight_DK_TRM
0x130	CTITRIGINSTATUS	RO	8	-	See ARM CoreSight_DK_TRM
0x134	CTITRIGOUTSTATUS	RO	8	0x00	See ARM CoreSight_DK_TRM
0x138	CTICHINSTATUS	RO	4	-	See ARM CoreSight_DK_TRM
0x13C	CTICHOUTSTATUS	RO	4	0x0	See ARM CoreSight_DK_TRM
0x140	Channel gate	R/W	4	0xF	See ARM CoreSight_DK_TRM
0x144	External multiplexer control	R/W	8	0x00	See ARM CoreSight_DK_TRM
0xEDC	ITCHINACK	WO	4	0x0	See ARM CoreSight_DK_TRM
0xEE0	ITTRIGINACK	WO	8	0x00	See ARM CoreSight_DK_TRM
0xEE4	ITCHOUT	WO	4	0x0	See ARM CoreSight_DK_TRM
0xEE8	ITTRIGOUT	WO	8	0x00	See ARM CoreSight_DK_TRM
0xEEC	ITCHOUTACK	RO	4	0x0	See ARM CoreSight_DK_TRM
0xEF0	ITTRIGOUTACK	RO	8	0x00	See ARM CoreSight_DK_TRM
0xEF4	ITCHIN	RO	4	0x0	See ARM CoreSight_DK_TRM
0xEF8	ITTRIGIN	RO	8	0x00	See ARM CoreSight_DK_TRM
0xEFC-0xF7C	-	-	-	-	See ARM CoreSight_DK_TRM
0xF00	ITCTRL	R/W	1	0x0	See ARM CoreSight_DK_TRM
0xFA0	Claim Tag Set	R/W	4	0xF	See ARM CoreSight_DK_TRM
0xFA4	Claim Tag Clear	R/W	4	0x0	See ARM CoreSight_DK_TRM
0xFB0	Lock Access Register	WO	32	-	See ARM CoreSight_DK_TRM
0xFB4	Lock Status Register	RO	2	0x3	See ARM CoreSight_DK_TRM
0xFB8	Authentication Status	RO	4	0xA	See ARM CoreSight_DK_TRM
0xFC0-0xFC4	-	-	-	-	See ARM CoreSight_DK_TRM
0xFC8	Device ID	RO	20	0x40800	See ARM CoreSight_DK_TRM

Table continues on the next page...

Table 9-10. CSCTI Register Summary (continued)

Offset	Name	Type	Width	Reset value	Description
0xFCC	Device Type Identifier	RO	8	0x14	See ARM CoreSight_DK_TRM
0xFD0	PeripheralID4	RO	8	0x04	See ARM CoreSight_DK_TRM
0xFD4	PeripheralID5	-	-	-	See ARM CoreSight_DK_TRM
0xFD8	PeripheralID6	-	-	-	See ARM CoreSight_DK_TRM
0xFDC	PeripheralID7	-	-	-	See ARM CoreSight_DK_TRM
0xFE0	PeripheralID0	RO	8	0x06	See ARM CoreSight_DK_TRM
0xFE4	PeripheralID1	RO	8	0xB9	See ARM CoreSight_DK_TRM
0xFE8	PeripheralID2	RO	8	0x0B	See ARM CoreSight_DK_TRM
0xFEC	PeripheralID3	RO	8	0x00	See ARM CoreSight_DK_TRM
0xFF0	Component ID0	RO	8	0x0D	See ARM CoreSight_DK_TRM
0xFF4	Component ID1	RO	8	0x90	See ARM CoreSight_DK_TRM
0xFF8	Component ID2	RO	8	0x05	See ARM CoreSight_DK_TRM
0xFFC	Component ID3	RO	8	0xB1	See ARM CoreSight_DK_TRM

9.2.1.7 TPIU Register Summary

The TPIU (Trace Port Interface Unit) registers are described in the following table. A more detailed description of the TPIU registers can be found in ARM's CoreSight_DK_TRM document. These registers can be accessed via the APB port and are mem map accessible.

Table 9-11. TPIU Register Summary

Offset	Name	Type	Width	Reset value	Description
0x000	Supported port sizes	RO	32	0xFFFFFFFF	See ARM CoreSight_DK_TRM
0x004	Current port size	R/W	32	0x00000001	See ARM CoreSight_DK_TRM
0x100	Supported trigger modes	RO	18	0x11F	See ARM CoreSight_DK_TRM
0x104	Trigger counter value	R/W	8	0x00	See ARM CoreSight_DK_TRM
0x108	Trigger multiplier	R/W	5	0x00	See ARM CoreSight_DK_TRM

Table continues on the next page...

Table 9-11. TPIU Register Summary (continued)

Offset	Name	Type	Width	Reset value	Description
0x200	Supported test pattern/ modes	RO	18	0x3000F	See ARM CoreSight_DK_TRM
0x204	Current test pattern/ mode	RO	18	0x00000	See ARM CoreSight_DK_TRM
0x208	Test pattern repeat counter	R/W	8	0x00	See ARM CoreSight_DK_TRM
0x300	Formatter flush and status	RO	3	0x6	See ARM CoreSight_DK_TRM
0x304	Fomatter flush and control	R/W	14	0x1000	See ARM CoreSight_DK_TRM
0x308	Formatter synchroniz ation counter	R/W	12	0x040	See ARM CoreSight_DK_TRM
0x400	EXTCTL In Port	RO	8	Undefined	See ARM CoreSight_DK_TRM
0x408	EXTCTL Out Port	R/W	8	0x00	See ARM CoreSight_DK_TRM
0xEE4	Integration Register, ITTRFLIN ACK	WO	2	-	See ARM CoreSight_DK_TRM
0xEE8	Integration Register, ITTRFLIN	WO	2	Undefined	See ARM CoreSight_DK_TRM
0xEEC	Integration Register, ITATBDAT A0	WO	5	Undefined	See ARM CoreSight_DK_TRM
0xEF0	Integration Register, ITATBCT R2	WO	2	-	See ARM CoreSight_DK_TRM
0xEF4	Integration Register, ITATBCT R1	RO	7	Undefined	See ARM CoreSight_DK_TRM

Table continues on the next page...

Table 9-11. TPIU Register Summary (continued)

Offset	Name	Type	Width	Reset value	Description
0xEF8	Integration Register, ITATBCTR0	RO	10	Undefined	See ARM CoreSight_DK_TRM
0xF00	Integration Register, ITATBCTR0	R/W	1	0x0	See ARM CoreSight_DK_TRM
0xFA0	Claim Tag Set Register	R/W	4	0xF	See ARM CoreSight_DK_TRM
0xFA4	Claim Tag Clear Register	R/W	4	0x0	See ARM CoreSight_DK_TRM
0xFB0	Lock Access Register	WO	32	-	See ARM CoreSight_DK_TRM
0xFB4	Lock Status Register	RO	3	0x3	See ARM CoreSight_DK_TRM
0xFB8	Authentication Status Register	RO	8	0x00	See ARM CoreSight_DK_TRM
0xFC8	Device ID	RO	32	0x00	See ARM CoreSight_DK_TRM
0xFCC	Device Type Identifier Register	RO	8	0x21	See ARM CoreSight_DK_TRM
0xFD0	Peripheral ID4	RO	8	0x04	See ARM CoreSight_DK_TRM
0xFD4	Peripheral ID5	RO	8	0x00 (reserved)	See ARM CoreSight_DK_TRM
0xFD*	Peripheral ID6	RO	8	0x00 (reserved)	See ARM CoreSight_DK_TRM
0xFDC	Peripheral ID7	RO	8	0x00 (reserved)	See ARM CoreSight_DK_TRM
0xFE0	Peripheral ID0	RO	8	0x07	See ARM CoreSight_DK_TRM
0xFE4	Peripheral ID1	RO	8	0xB9	See ARM CoreSight_DK_TRM
0xFE8	Peripheral ID2	RO	8	0x0B	See ARM CoreSight_DK_TRM

Table continues on the next page...

Table 9-11. TPIU Register Summary (continued)

Offset	Name	Type	Width	Reset value	Description
0xFEC	Peripheral ID3	RO	8	0x00	See ARM CoreSight_DK_TRM
0xFF0	Component ID0	RO	8	0x0D	See ARM CoreSight_DK_TRM
0xFF4	Component ID1	RO	8	0x90	See ARM CoreSight_DK_TRM
0xFF8	Component ID2	RO	8	0x05	See ARM CoreSight_DK_TRM
0xFFC	Component ID3	RO	8	0xB1	See ARM CoreSight_DK_TRM

9.2.2 Clocks

The list below describes the Debug clocks within the ARM platform.

- ATCLK - This is the AMBA Trace Bus (ATB) clock. This clock is also used to clock the on-platform CTICLK and on-platform CTMCLK. ATCLK needs to be synchronous (equivalent or faster) to PCLKDBG.
- CTICLK - This is the main clock for the CTI block. The current ARM configuration requires this clock to be synchronous to the on-platform CTM clock.
- PCLKDBG - This is the Debug APB clock. It must be synchronous (equivalent or slower) to the ATCLK.
- TRACECLKIN - This is the Trace Port Interface Unit trace clock input. Since the trace port pads are limited to 133 mhz, TRACECLKIN is pad limited to less than or equal to 266 mhz.
- TRACECLK - This is the clock for the external trace port. It's a DDR clock and runs at 1/2 the speed of TRACECLKIN.

The platform control block generates the debug clocks from the arm clock. The requirement above states that the PCLKDBG needs to be synchronous to the ATCLK. When configuring the divide by settings, make sure this requirement is followed. PCLKDBG is a divide by 2 of ATCLK.

9.2.3 Reset

All debug resets are derived from JTAG_TRST_B and POR. These two resets go into the DAP_SYS to be combined and de-asserted synchronously to the DAPCLKao (always on clock). They also go into the ARM platform, being synchronously de-asserted inside the plat_cntrl block, generating resets for the debug logic inside the platform. Debug resets

that do not require synchronous de-assertion are routed directly to their destination, such as por for the Core and ETM. The figure below shows a diagram of the ARM platform debug reset strategy.

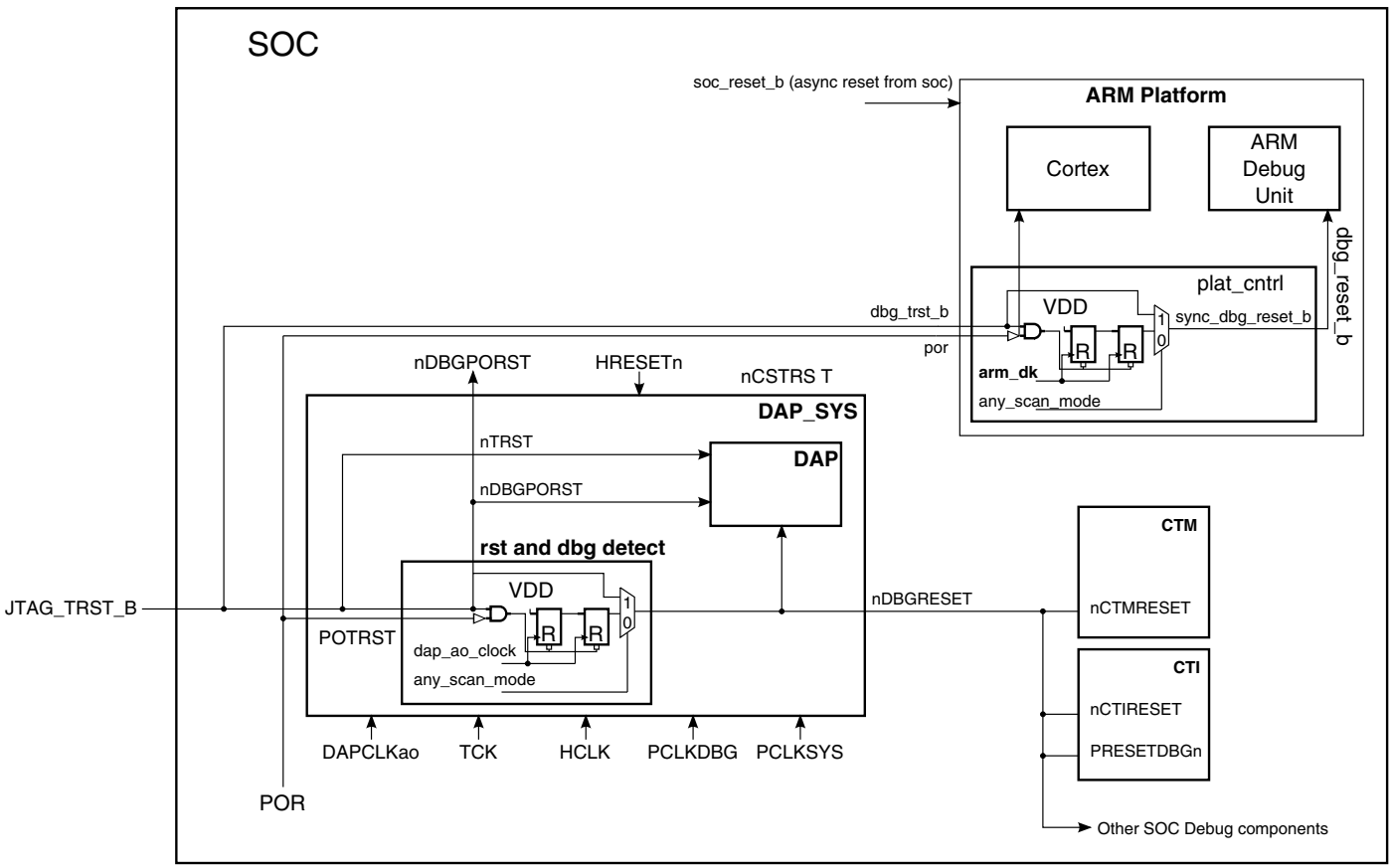


Figure 9-8. ARM Debug Reset Strategy

Chapter 10
Multi-Layer AHB Crossbar Switch (AHBMAX)

10.1 Overview

This section provides an overview of the Multi-Layer AHB Crossbar Switch (AHBMAX). The purpose of the AHBMAX is to concurrently support up to four simultaneous connections between master ports and slave ports. The AHBMAX supports a 32-bit address bus width and a 32-bit data bus width at all master and slave ports. A simplified block diagram is shown in Figure 10-1.

NOTE

The AHBMAX implements a 7 master by 4 slave configuration.

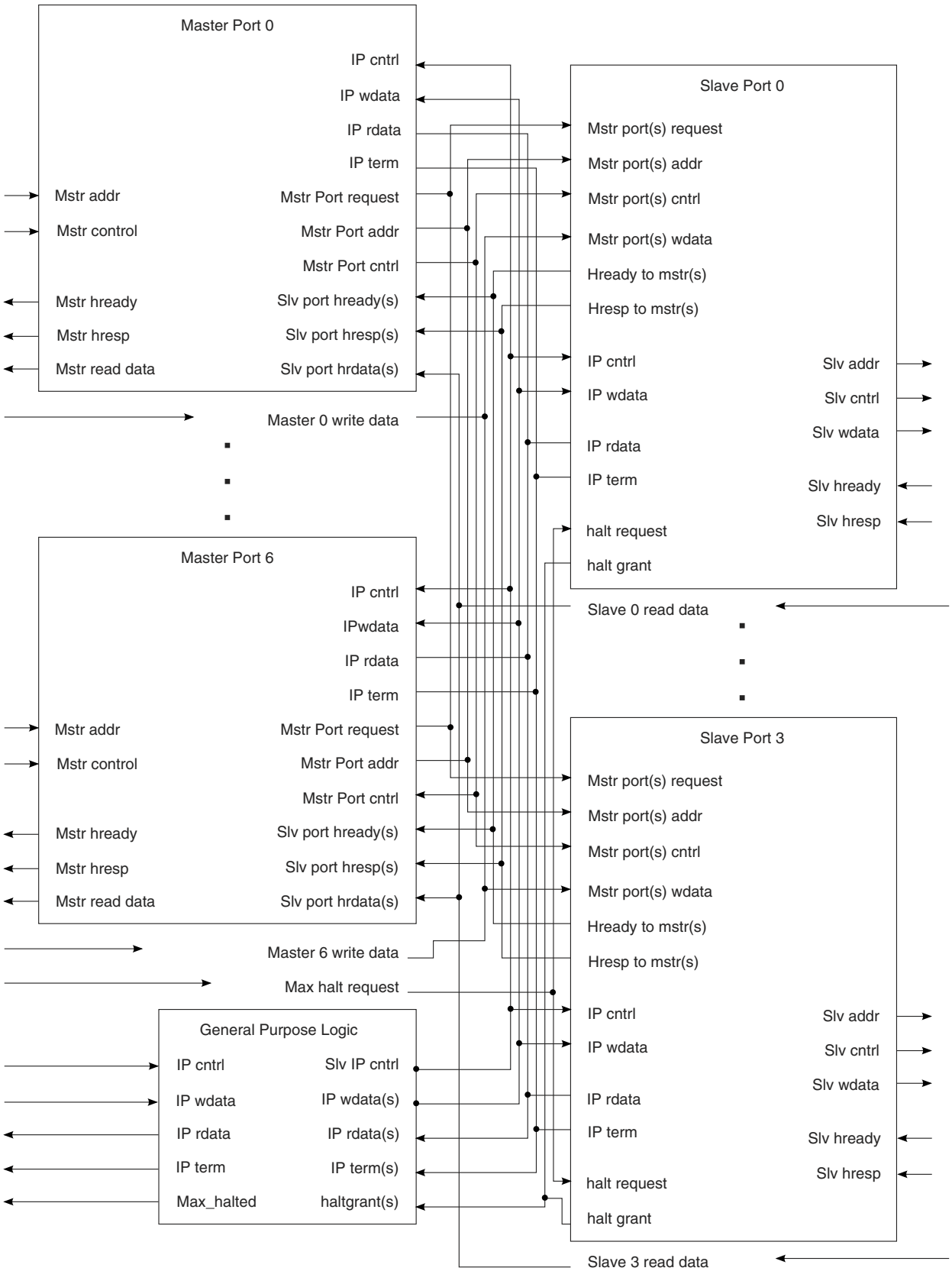


Figure 10-1. AHBMAX Block Diagram

10.2 Features

The AHBMAX has the ability to gain control of all the slave ports and prevent any masters from making accesses to the slave ports. This feature is useful when the user wishes to turn off the clocks to the system and needs to ensure that no bus activity will be interrupted.

The AHBMAX can put each slave port into a low power park mode so that slave port will not dissipate any power transitioning address, control or data signals when not being actively accessed by a master port.

Each slave port can also support multiple master priority schemes. Each slave port has a hardware input which selects the master priority scheme so the user can dynamically change master priority levels on a slave port by slave port basis.

The AHBMAX will allow for concurrent transactions to occur from any master port to any slave port. It is possible for four master ports and all slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic will select the higher priority master and grant it ownership of the slave port. All other masters requesting that slave port will stalled until the higher priority master completes its transactions.

10.2.1 Limitations

The AHBMAX routes bus transactions initiated on the master ports to the appropriate slave ports. There is no provision included to route transactions initiated on the slave ports to other slave ports or to master ports. Simply put, the slave ports do not support the bus request/bus grant protocol, the AHBMAX assumes it is the sole master of each slave port.

Because the AHBMAX does not support the bus request/bus grant protocol, if multiple masters are to be connected to a single master port an external arbiter will need to be used. In the case of a single master connecting to a master port the single master's bus grant signal must be tied off in the asserted state.

Each master and slave port is fully AHB-Lite + AMBA V6 extensions compliant. The ports are not fully AHB compliant because the AHBMAX does not support SPLITs or RETRYs.

10.2.2 General Operation

When a master makes an access to the AHBMAX the access will be immediately taken by the AHBMAX. If the targeted slave port of the access is available then the access will be immediately presented on the slave port. It is possible to make single clock (zero wait state) accesses through the AHBMAX. If the targeted slave port of the access is busy or parked on a different master port the requesting master will simply see wait states inserted (hready held negated) until the targeted slave port can service the master's request. The latency in servicing the request will depend on each master's priority level and the responding peripheral's access time.

Because the AHBMAX appears to be just another slave to the master device, the master device will have no knowledge of whether or not it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting it will simply be wait stated.

A master will be given control of the targeted slave port only after a previous access to a different slave port has completed, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when a master has an outstanding request to one slave port that has a long response time, has a pending access to a different slave port, and a lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

Once the master has control of the slave port it is targeting the master will remain in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a locked or fixed length burst transfer it will retain control of the slave port until that transfer is completed. Based on the AULB bit in the MGPCR (Master General Purpose Control Register) the master will either retain control of the slave port when doing undefined length incrementing burst transfers or will lose the bus to a higher priority master.

The AHBMAX will terminate all master IDLE transfers (as opposed to allowing the termination to come from one of the slave busses). Additionally, when no master is requesting access to a slave port the AHBMAX will drive IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port. When the AHBMAX is controlling the slave bus (i.e. during low power park or halt mode) the hmaster field will indicate 4'b0000.

When a slave bus is being IDLEd by the AHBMAX it can park the slave port on the master port indicated by the PARK bits in the AHBMAX_SGPCR (Slave General Purpose Control Register). This can be done in an attempt to save the initial clock of

arbitration delay that would otherwise be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low power park mode in attempt to save power.

10.3 AHBMAX Interface Signals

This section provides information on AHBMAX interface signals, including AHB master and slave interface signals as well as IP bus interface signals.

10.3.1 AHBMAX Signal Descriptions

Please reference the AMBA Specification Rev 2.0 for a description of the AHB signals in the AHBMAX and the IP Bus Specification Rev 2.0 for a description of the IP Bus signals in the AHBMAX.

10.3.1.1 max_halt_request

This input signal is a request to halt all slave port bus activity (run AHBMAX originated IDLE cycles on each slave port bus, blocking all master port accesses). This signal can be used to gracefully shut down the AHBMAX so the system clock can be stopped for low power mode. This signal is captured by a flop inside the AHBMAX before use.

Once the AHBMAX is halted it will remain halted until max_halt_request is negated.

10.3.1.2 max_halted

This output is asserted once the AHBMAX is in control and running IDLE cycles on each slave port.

10.4 Coherency

Since the content of the registers has a real time effect on the operation of the AHBMAX it is important for the user to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave port related AHB accesses but instead track only with IP bus accesses.

The exception to this rule are the AULB bits in the AHBMAX_MGPCR*n*. The update of these bits is only recognized when the master on that master port runs an IDLE cycle, even though the IP bus cycle to write them will have long since terminated successfully. If the AULB bits in the AHBMAX_MGPCR*n* are written in between two burst accesses the new AULB encodings will not take effect until an IDLE cycle has been initiated by the master on that master port.

10.5 Detailed Functional Description

This section describes the functionality of the AHBMAX in greater detail.

10.5.1 Arbitration

The AHBMAX supports two arbitration schemes; a simple fixed-priority comparison algorithm, and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

10.5.1.1 Arbitration During Undefined Length Bursts

Arbitration points during an undefined length burst are defined by the current master's AHBMAX_MGPCR*n*[AULB] field setting. When a defined length is imposed on the burst via the AULB bits the undefined length burst will be treated as a single or series of single back to back fixed length burst accesses.

Example: A master runs an undefined length burst and the AULB bits in the AHBMAX_MGPCR*n* indicate arbitration will occur after the fourth beat of the burst. The master runs two sequential beats and then starts what will be an 12 beat undefined length burst access to a new address within the same slave port region as the previous access. The AHBMAX will not allow an arbitration point until the fourth overall access (second beat of the second burst). At that point all remaining accesses will be open for arbitration until the master loses control of the slave port.

Assume the master loses control of the slave port after the fifth beat of the second burst. Once the master regains control of the slave port no arbitration point will be available until after the master has run four more beats of its burst. After the fourth beat of the (now continued) burst (ninth beat of the second burst from the master's perspective) is taken all beats of the burst will once again be open for arbitration until the master loses control of the slave port.

Assume the master again loses control of the slave port on the fifth beat of the third (now continued) burst (10th beat of the second burst from the master's perspective). Once the master regains control of the slave port it will be allowed to complete its final two beats of its burst without facing arbitration.

Note that fixed length burst accesses will not be affected by the AULB bits. All fixed length burst accesses will lock out arbitration until the last beat of the fixed length burst.

10.5.1.2 Fixed Priority Operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the AHBMAX_MPR*n* (Master Priority Register). If two masters both request access to a slave port the master with the highest priority in the selected priority register will gain control over the slave port.

Any time a master makes a request to a slave port the slave port checks to see if the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port the new requesting master will be granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed length burst transfer or a locked transfer. In this case the new requesting master will have to wait until the end of the burst transfer or locked transfer before it will be granted control of the slave port. If the master is running an undefined length burst transfer the new requesting master must wait until an arbitration point for the undefined length burst transfer before it will be granted control of the slave port. Arbitration points for an undefined length burst are defined in the MGPCR for each master.

If the new requesting master's priority level is lower than that of the master that currently has control of the slave port the new requesting master will be forced to wait until the master that currently has control of the slave port either runs an IDLE cycle or runs a non IDLE cycle to a location other than the current slave port.

10.5.1.3 Round-Robin Priority Operation

When operating in round-robin mode, each master is assigned a relative priority based on the master number. This relative priority is compared to the ID of the last master to perform a transfer on the slave bus. The highest priority requesting master will become

owner of the slave bus as the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far ahead the ID of the requesting master is to the ID of the last master (ID is defined by master port number, not the hmaster field).

Once granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line will be granted access to the slave port at the next assertion of slave "n" hready, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the AHBMAX is implemented with master ports 0, 1, 2, 3, 4 and 5. If the last master of the slave port was master 1, and master 0, 4 and 5 make simultaneous requests, (master ports 2 and 3 make no requests), they will be serviced in the order 4, 5 and then 0.

Parking may still be used in a round-robin mode, but will not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff will occur to the next master in line after one cycle of arbitration. If the slave port is put into low power park mode the round-robin pointer will be reset to point at master port 0, giving it the highest priority.

10.5.2 Priority Assignment

Each master port needs to be assigned a unique 3 bit priority level. If an attempt is made to program multiple master ports with the same priority level within a register (MPR) the AHBMAX will respond with an error and the registers will not be updated.

10.5.3 Master Port Functionality

10.5.3.1 Master Port General Information

Each master port consists of two decoders, a capture unit, a register slice, a mux and a small state machine.

The first decoder is used to decode the haddr and control signals coming directly from the master, telling the state machine where the master's next access will be and if it is in fact a legal access. The second decoder gets its input from the capture unit, so it may be looking directly at the signals coming from the master or it may be looking at captured signals coming from the master, depending entirely on the state of the targeted slave port. The second decoder is then used to generate the access requests that go to the slave ports.

The capture unit is used to capture the address and control information coming from the master in the event that the targeted slave port cannot immediately service the master. The capture unit is controlled by outputs from the state machine which tell it to either pass through the original master signals or the captured signals.

The register slice contains the registers associated with the specific master port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine.

The mux is used simply to select which slave's read data is sent back to the master. The mux is controlled by the state machine.

The state machine controls all aspects of the master port. It knows which slave port the master wants to make a request to and controls when that request is made. It also has knowledge of each slave port, knowing whether or not the slave port is ready to accept an access from the master port. This will determine whether or not the master may immediately have its request taken by the slave port or whether the master port will have to capture the master's request and queue it at the slave port boundary.

For a block diagram of a master port see Figure 10-2.

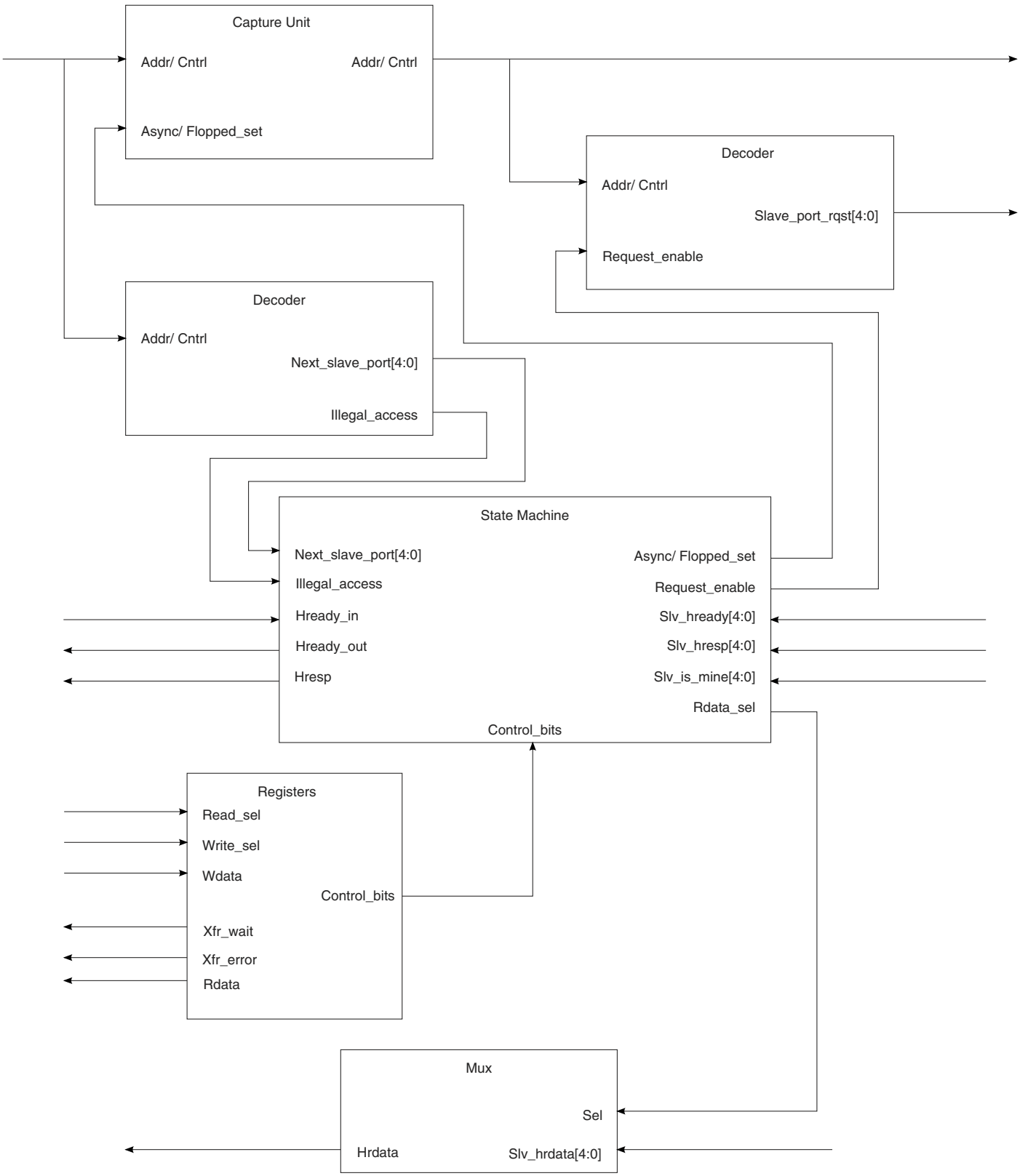


Figure 10-2. AHBMAX Master Port Block Diagram

10.5.3.2 Master Port Decoders

The decoders are very simple as they ensure an access request is allowed to be made and that the slave port targeted is actually present in the design. The decoders feeding the state machine are always enabled. The decoders that select the slave are enabled only when the master port controlling state machine wants to make a request to a slave port. This is necessary so that if a master port is making an access to a slave port and is being wait stated, and its next access is to a different slave port, the request to the second slave port can be held off until the access to the first slave port is terminated.

The decoders also output a "hole decode" or illegal access signal which tells the state machine that the master is trying to access a slave port that does not exist.

10.5.3.3 Master Port Capture Unit

The capture unit simply captures the state of the master's address and control signals if the AHBMAX cannot immediately pass the master's request through to the proper slave port. The capture unit consists of a set of flops and a mux which selects either the asynchronous path from address and control or the flopped (captured) address and control information.

10.5.3.4 Master Port Registers

The registers in the master port are only those registers associated with this particular master port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design.

There is a register control block at the same level of the master port and slave port instantiations in the AHBMAX. This control block ensures that all accesses are 32-bit supervisor accesses before passing them on to the master ports.

The register outputs are connected directly to the state machine.

10.5.3.5 Master Port State Machine

10.5.3.5.1 Master Port State Machine States

The master side state machine's main function is to monitor the activities of the master port. The state machine has six states: busy, idle, stalled, steady state, first cycle error response and second cycle error response.

The busy state is used when the master runs a BUSY cycle to the master port. The master port maintains its request to the slave port if it currently owns the slave port; however, if it loses control of the slave port it will no longer maintain its request. If the master port loses control of the slave port it will not be allowed to make another request to the slave port until it runs a NSEQ or SEQ cycle.

The idle state is used when the master runs a valid IDLE cycle to the master port. The master port makes no requests to the slave ports (disables the slave port decoder) and terminates the IDLE cycle.

The stalled state is used when the master makes a request to a slave port that is not immediately ready to receive the request. In this case the state machine will direct the capture unit to send out the captured address and control signals and will enable the slave port decoder to indicate a pending request to the appropriate slave port.

The steady state is used when the master port and slave port are in fully asynchronous mode, making the AHBMAX completely transparent in the access. The state machine selects the appropriate slave's hresp0, hready and hrdata to pass back to the master.

The first cycle error response and second cycle error response states are self explanatory. The AHBMAX will respond with an error response to the master if the master tries to access an unimplemented memory location through the AHBMAX (i.e. a slave port that does not exist).

10.5.3.5.2 Master Port State Machine Slave Swapping

The design of the master side state machine is fairly straight forward. The one real decision to be made is how to handle the master moving from one slave port access to another slave port access. The approach that was taken was to minimize or eliminate, when possible, any "bubbles" that would get inserted into the access due to switching slave ports.

The state machine will not allow the master to request access to another slave port until the current access being made is terminated. This prevents a single master from owning two slave ports at the same time (the slave port it is currently accessing and the slave port it wishes to access next).

The state machine also maintains watch on the slave port the master is accessing as well as the slave port the master wishes to switch to. If the new slave port is parked on the master then the master will be able to make the switch without incurring any delays. The termination of the current access will also act as the launch of the new access on the new slave port. If the new slave port is not parked on the master then the master will incur a minimum one clock delay before it can launch its access on the new slave port.

This is the same for switching from the busy or idle state to actively accessing a slave port. If the slave port is parked on the master the state machine will go to the steady state and the access will begin immediately. If the slave port is not parked on the master (serving another master, parked on another master or in low power park mode) then the state machine will transition to the stalled state and at least a one clock penalty will be paid.

10.5.4 Slave Port Functionality

10.5.4.1 Slave Port General Information

Each slave port consists of a register slice, a bank of muxes and a state machine.

The register slice contains the registers associated with the specific slave port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine.

The muxes are a series of 6 to 1 multiplexers that take in all the address, control and write data information from each of the master ports and then pass the correct master's signals to the slave port. The state machine controls all the muxes.

The state machine is where the main slave port arbitration occurs, it decides which master is in control of the slave port and which master will be in control of the slave port in the next bus cycle.

For a block diagram of a slave port see Figure 10-3.

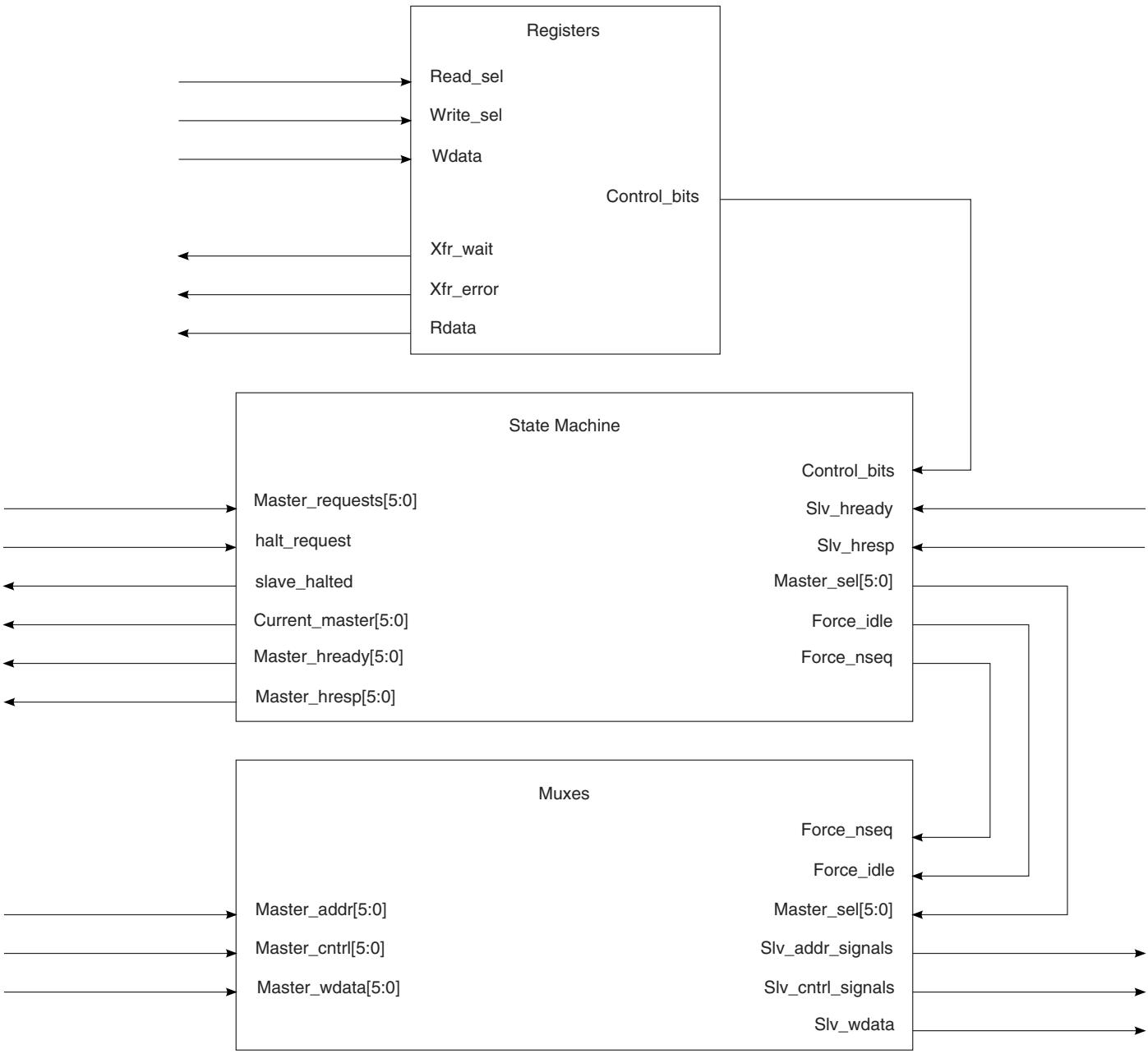


Figure 10-3. AHBMAX Slave Port Block Diagram

10.5.4.2 Slave Port Muxes

The block diagram (Figure 10-3) shows only one block for all the muxes. In reality that block instantiates many 6 to 1 muxes, one for each master-to-slave signal in fact. All the muxes are designed in an AND - OR fashion, so that if no master is selected the output of the muxes will be zero. (This is an important feature for low power park mode.)

The muxes also have an override signal which is used by the slave port to asynchronously force IDLE cycles onto the slave bus. When the state machine forces an IDLE cycle it zeros out htrans and hmastlock, making sure the slave bus sees a valid IDLE cycle being run by the AHBMAX.

The enable to the mux controlling htrans also contains an additional control signal from the state machine so that a NSEQ transaction can be forced. This is done any time the slave port switches masters to ensure that no IDLE-SEQ, BUSY-SEQ or NSEQ-SEQ transactions are seen on the slave port when they shouldn't be. If the state machine indicates to run both an IDLE and an NSEQ cycle, the IDLE directive will have priority.

NOTE

IDLE-SEQ is in fact an illegal access, but a possible scenario given the multi-master environment in the AHBMAX unless corrected by the AHBMAX.

10.5.4.3 Slave Port Registers

There is a register control block at the same level of the master port and slave port instantiations in the AHBMAX. This control block ensures that all accesses are 32-bit supervisor accesses before passing them on to the master and slave ports.

The registers in the slave port are only those registers associated with this particular slave port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design.

The register outputs are connected directly to the slave state machine. The registers can be read from an unlimited number of times. The registers can only be written to as long as the RO bit is written to 0 in the AHBMAX_SGPCRn, once it is written to a 1 only a hardware reset will allow the registers to be written again.

10.5.4.4 Slave Port State Machine

10.5.4.4.1 Slave Port State Machine States

At the heart of the slave port is the state machine. The state machine is simplicity itself, requiring only four states - steady state, transition state, transition hold state and hold state. Either the slave port is owned by the same master it was in the last clock cycle (either by active use or by parking), it is transitioning to a new master (either for active use or parking), it is transitioning to a new master during wait states or it is being held on the same master pending a transition to a new master.

10.5.4.4.2 Slave Port State Machine Arbitration

The real work in the state machine is determining which master port will be in control of the slave port in the next clock cycle, the arbitration. Each master is programmed with a fixed 3 bit priority level. The AHBMAX uses these bits in determining priority levels when programmed for fixed priority mode of operation.

Arbitration always occurs on a clock edge, but only occurs on edges when a change in mastership will not violate AHB-Lite protocols. Valid arbitrations points include any clock cycle in which slave n hready is asserted (provide the master is not performing a burst or locked cycle) and any wait state in which the master owning the bus indicates a transfer type of IDLE (provided the master is not performing a locked cycle).

Since arbitration can occur on every clock cycle the slave port masks off all master requests if the current master is performing a locked transfer or a protected burst transfer, guaranteeing that no matter how low its priority level it will be allowed to finish its locked or protected portion of a burst sequence.

10.5.4.4.3 Slave Port State Machine Master Handoff

The only times the slave port will switch masters when programmed for fixed priority mode of operation is when a higher priority master makes a request or when the current master is the highest priority and it gives up the slave port by either running and IDLE cycle to the slave port or running a valid access to a location other than the slave port.

If the current master loses control of the slave port because a higher priority master takes it away the slave port will not incur any wasted cycles. The current master will get its current cycle terminated by the slave port at the same time the new master's address and control information will be recognized by the slave port. This will look like a seamless transition on the slave port.

If the current master is being wait stated when the higher priority master makes its request, then the current master will be allowed to make one more transaction on the slave bus before giving it up to the new master. Figure 10-4 illustrates the effect of a higher priority master taking control of the bus when the slave port is programmed for a fixed priority mode of operation.

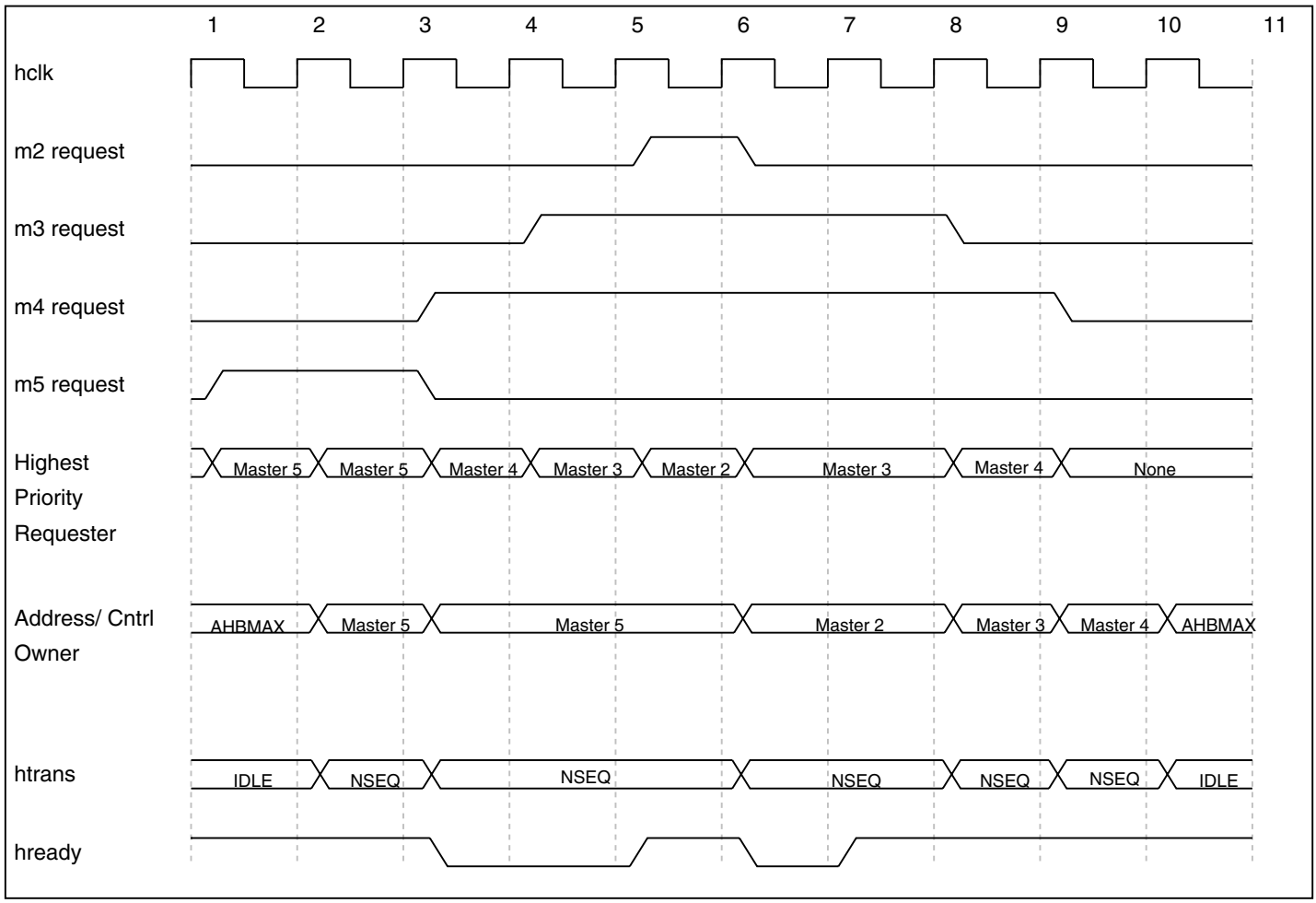


Figure 10-4. Low to High Priority Mastership Change

If the current master is the highest priority master and it gives up the slave port by running an IDLE cycle or by running a valid cycle to another location other than the slave port the next highest priority master will gain control of the slave port. If the current access incurs any wait states then the transition will be seamless and no bandwidth will be lost; however, if the current transaction is terminated without wait states then one IDLE cycle will be forced onto the slave bus by the AHBMAX before the new master will be able to take control of the slave port. If no other master is requesting the bus then IDLE cycles will be run by the AHBMAX but no bandwidth will truly be lost since no master is making a request.

Figure 10-5 illustrates the effect of a higher priority master giving up control of the bus.

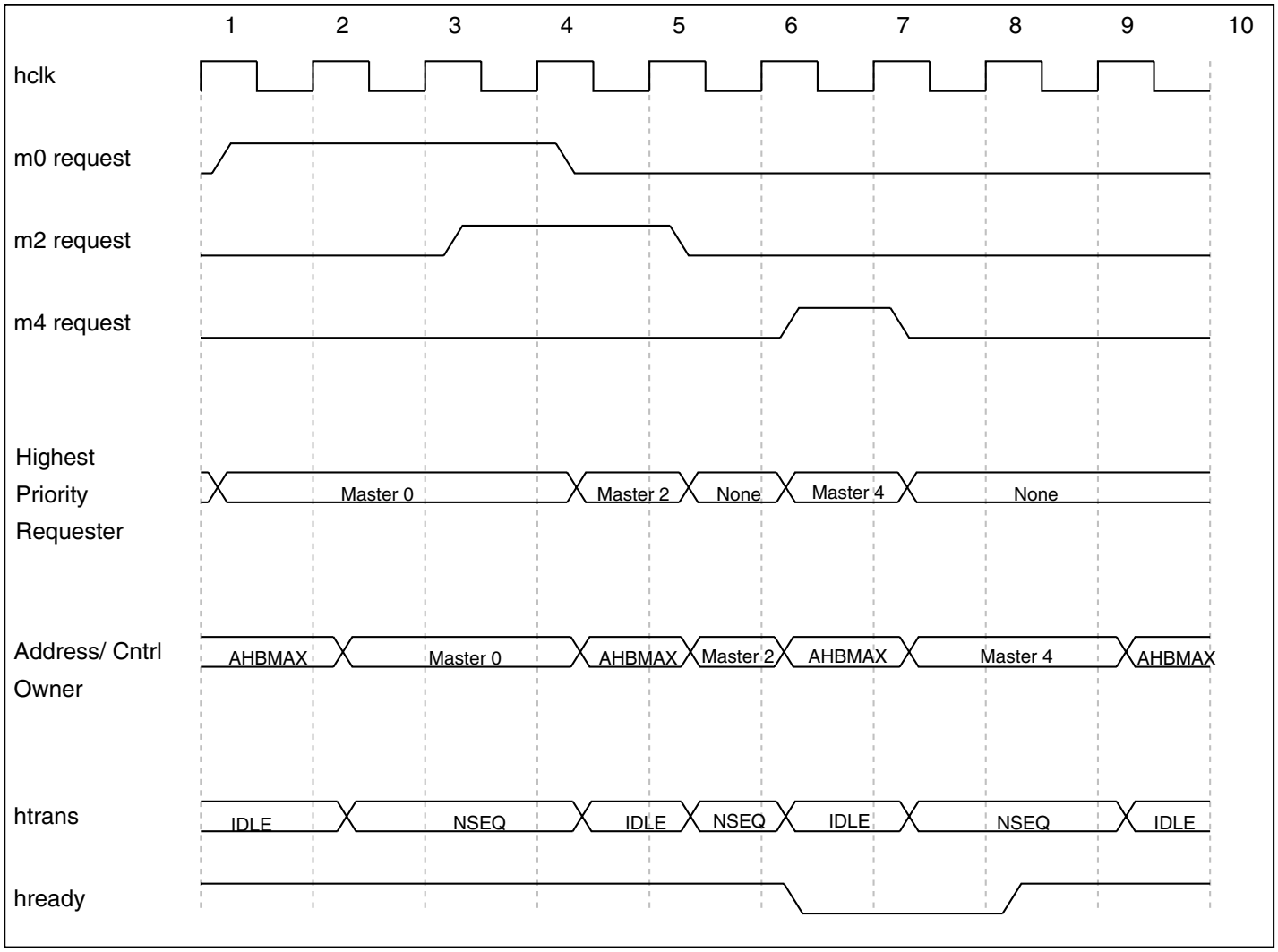


Figure 10-5. High to Low Priority Mastership Change

When the slave port is programmed for round-robin mode of arbitration then the slave port will switch masters any time there is more than one master actively making a request to the slave port. This will happen because any master other than the one which presently owns the bus will be considered to have higher priority.

Figure 10-6 shows an example of round-robin mode of operation.

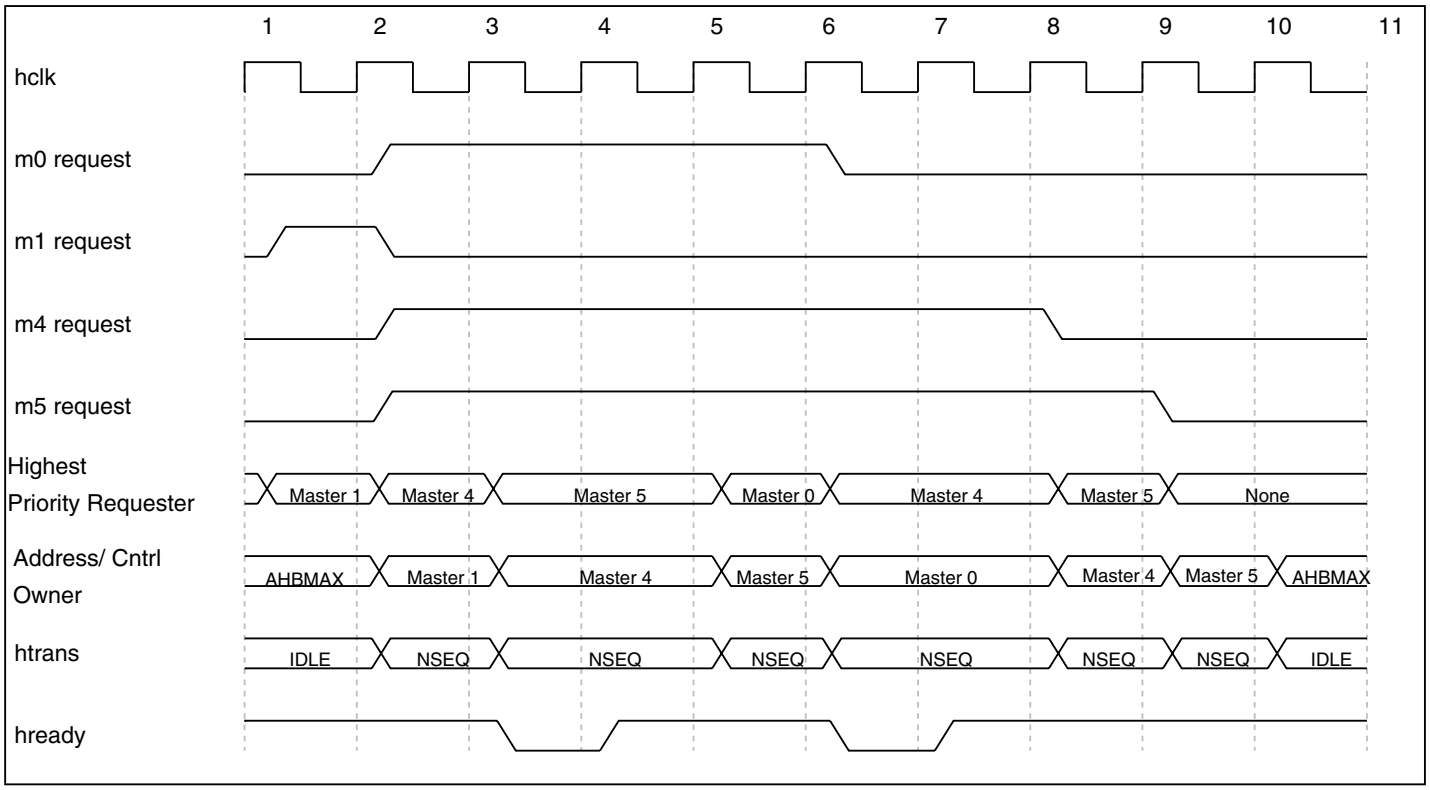


Figure 10-6. Round-robin Mastership Change

10.5.4.4.4 Slave Port State Machine Parking

If no master is currently making a request to the slave port then the slave port will be parked. It will park in one of four places, dictated by the PCTL and PARK bits in the AHBMAX_SGPCR and the locked state of the last master to access it.

If the last master to access the slave port ran a locked cycle and continues to run locked cycles even after leaving the slave port the slave port will park on that master without regard to the bit settings in the AHBMAX_SGPCR and without regard to pending requests from other masters. This is done so a master can run a locked transfer to the slave port, leave it, and return to it and be guaranteed that no other master has had access to it (provided the master maintains all transfers are locked transfers). If locking is not an issue for parking the AHBMAX_SGPCR bits will dictate the parking method.

If the PCTL bits are set for "low power park" mode then the slave port will enter low power park mode. It will not recognize any master as being in control of it and it will not select any master's signals to pass through to the slave bus. In this case all slave bus activity will effectively halt because all slave bus signals being driven from the AHBMAX will be 0. This of course can save quite a bit of power if the slave port will not

be in use for some time. The down side is that when a master does make a request to the slave port it will be delayed by one clock since it will have to arbitrate to acquire ownership of the slave port.

If the PCTL bits are set to "park on last" mode then the slave port will park on the last master to access it, passing all that masters signals through to the slave bus. The AHBMAX will asynchronously force htrans[1:0], hmaster[3:0], hburst[2:0] and hmastlock to 0 for all access that the master does not run to the slave port. When that master access the slave port again it will not pay any arbitration penalty; however, if any other master wishes to access the slave port a one clock arbitration penalty will be imposed.

If the PCTL bits are set to "use PARK" mode then the slave port will park on the master designated by the PARK bits. The behavior here is the same as for the "park on last" mode with the exception that a specific master will be parked on instead of the last master to access the slave port. If the master designated by the PARK bits tries to access the slave port it will not pay an arbitration penalty while any other master will pay a one clock penalty. Figure 10-7 illustrates parking on a specific master.

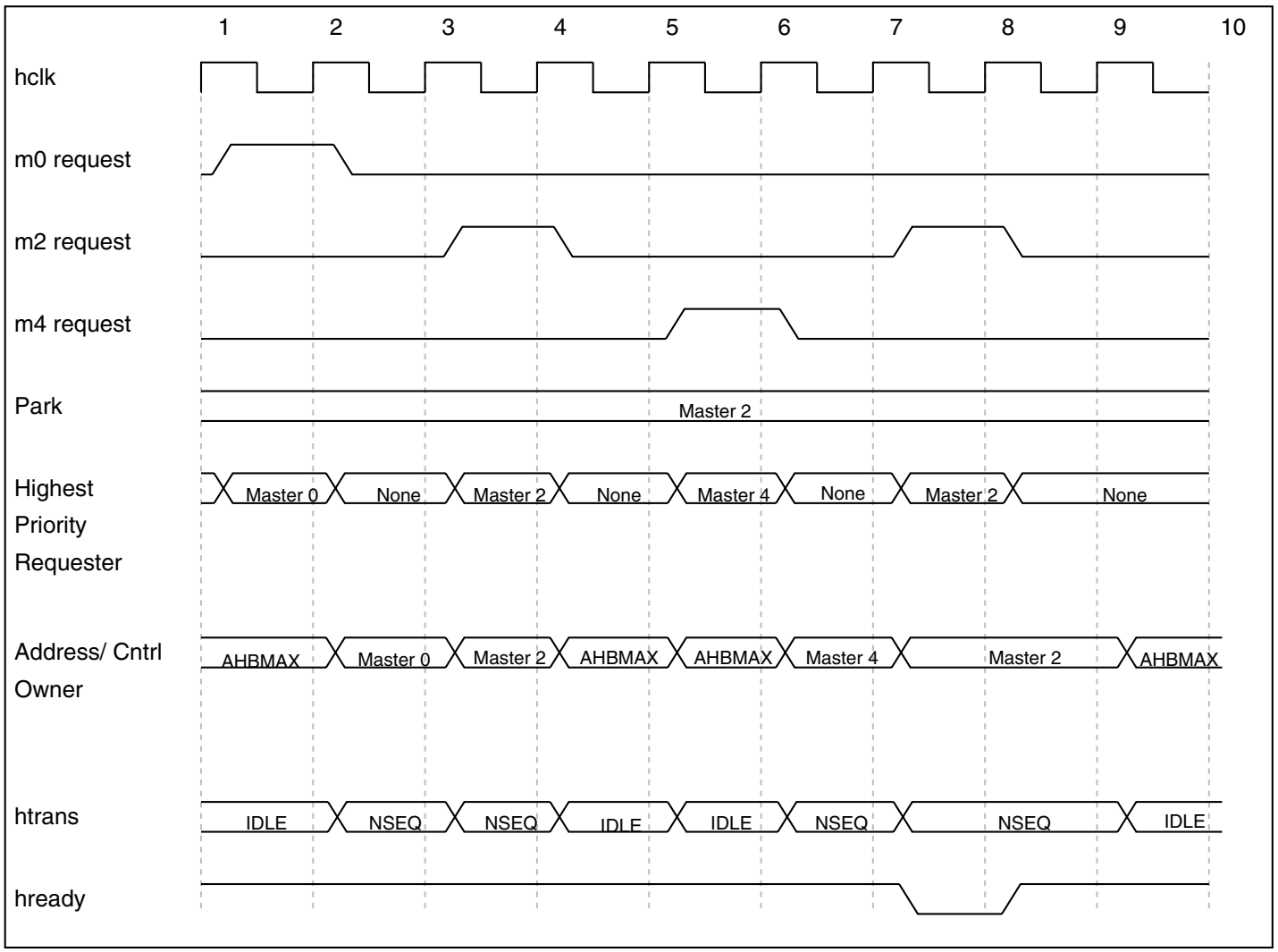


Figure 10-7. Parking on a Specific Master

Figure 10-8 illustrates parking on the last master. Note that in cycle 6 simultaneous requests are made by master 2 and master 4. Although master 2 has higher priority, the slave bus is parked on master 4 so master 4's access will be taken first. The slave port parks on master 2 once it has given control to master 2. This same situation can occur when parking on a specific master as well.

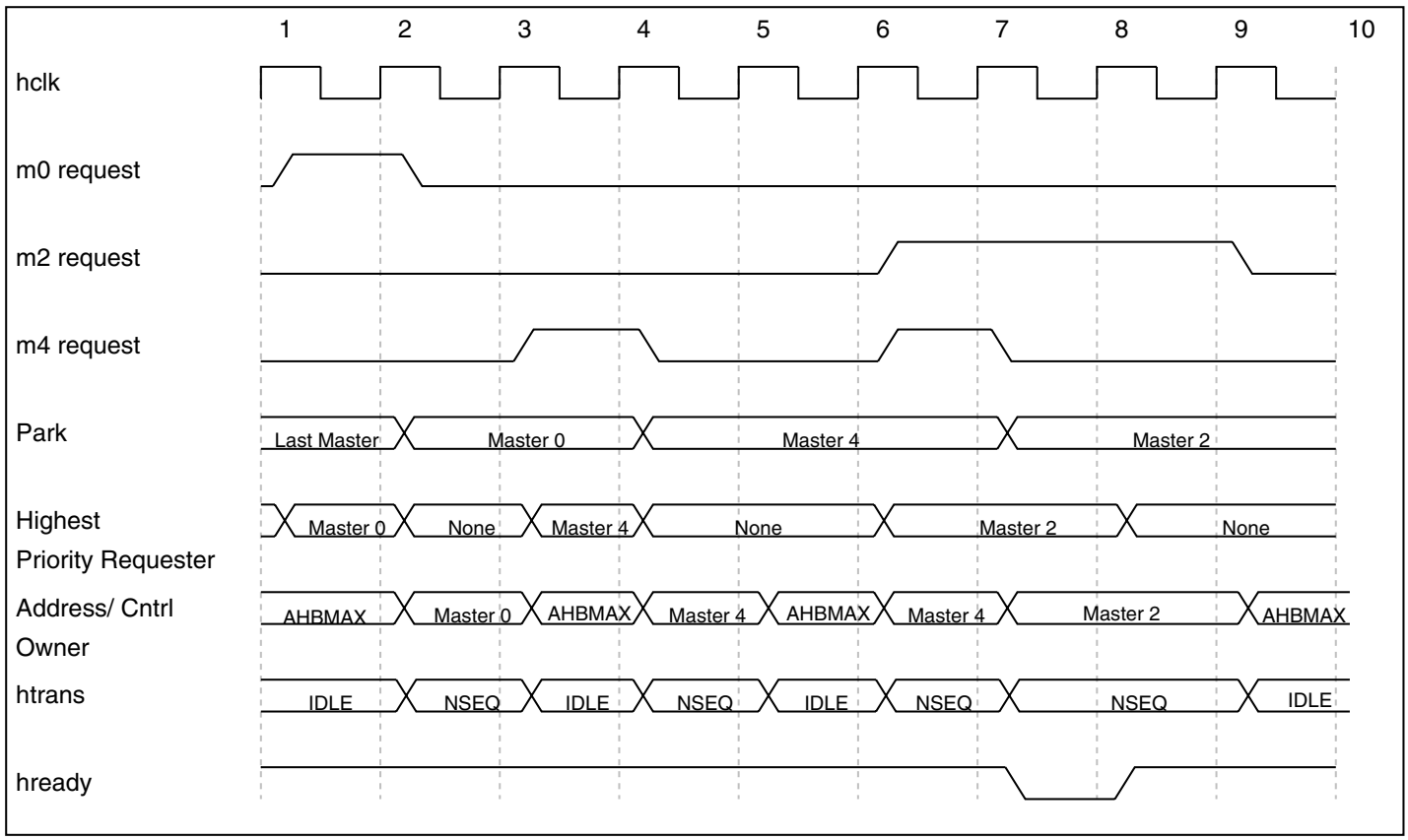


Figure 10-8. Parking on Last Master

10.5.4.4.5 Slave Port State Machine Halt Mode

If the max_halt_request input is asserted the slave port will eventually halt all slave bus activity and go into halt mode, which is almost identical to low power park mode. The HLP bit in the AHBMAX_SGPCR controls the priority level of the max_halt_request in the arbitration algorithm. If the HLP bit is cleared then the max_halt_request will have the highest priority of any master and will gain control of the slave port at the next arbitration point (most likely the next bus cycle, unless the current master is running a locked or fixed length burst transfer). If the HLP bit is set then the slave port will wait until no masters are actively making requests before moving to halt mode.

Regardless of the state of the HLP bit, once the slave port has gone into halt mode as a result of max_halt_request being asserted, it will remain in halt mode until max_halt_request is negated, regardless of the priority level of any masters that may make requests.

In halt mode no master is selected to own the slave port so all the outputs of the slave port are set to 0.

10.6 Initialization/Application Information

No initialization is required by or for the AHBMAX. Hardware reset ensures all the register bits used by the AHBMAX are properly initialized.

10.7 AHBMAX Interface

This section provides information on the AHBMAX interface.

10.7.1 AHBMAX Interface Overview

The main goal of the AHBMAX is to increase overall system performance by allowing multiple masters to communicate in parallel with multiple slaves. In order to maximize data throughput it is essential to keep arbitration delays to a minimum.

This section examines data throughput from the point of view of masters and slaves, detailing when the AHBMAX will stall the masters or insert bubbles on the slave side.

10.7.2 Master Ports-AHBMAX Interface

Master accesses will receive one of four responses from the AHBMAX. They will either be terminated, taken, stalled or responded to with an error.

10.7.2.1 Terminated Accesses

A master access will be terminated if the transfer type is IDLE. The AHBMAX will terminated the access and it will not be allowed to pass through the AHBMAX.

10.7.2.2 Taken Accesses

A master access will be taken if the transfer type is non IDLE and the slave port to which the access decodes is either currently servicing the master or is parked on the master. In this case the AHBMAX will be completely transparent and the master's access will be immediately seen on the slave bus and no arbitration delays will be incurred.

10.7.2.3 Stalled Accesses

A master access will be stalled if the transfer type is non IDLE and the access decodes to a slave port that is busy serving another master, parked on another master or is in low power park mode. The AHBMAX will indicate to the master that the address phase of the access has been taken but will then queue the access to the appropriate slave port to enter into arbitration for access to that slave port.

If the slave port is currently parked on another master or is in low power park mode and no other master is requesting access to the slave port then only one clock of arbitration will be incurred. If the slave port is currently serving another master of a lower priority and the master has a higher priority than all other requesting masters then the master will gain control over the slave port as soon as the data phase of the current access is completed (burst and locked transfers excluded). If the slave port is currently servicing another master of a higher priority then the master will gain control of the slave port once the other master releases control of the slave port if no other higher priority master is also waiting for the slave port.

10.7.2.4 Error Response Terminated Accesses

A master access will be responded to with an error if the transfer type is non IDLE and the access decodes to a location not occupied by a slave port.

10.7.3 Slave Ports-AHBMAX Interface

The goal of the AHBMAX with respect to the slave ports is to keep them 100% saturated when masters are actively making requests. In order to do this the AHBMAX must not insert any bubbles onto the slave bus unless absolutely necessary.

There is only one instance when the AHBMAX will force a bubble onto the slave bus when a master is actively making a request. This occurs when a higher priority master has control of the slave port and is running single clock (zero wait state) accesses while a lower priority master is stalled waiting for control of the slave port. When the higher priority master either leaves the slave port or runs an IDLE cycle to the slave port the AHBMAX will take control of the slave bus and run a single IDLE cycle before giving the slave port to the lower priority master that was waiting for control of the slave port.

The only other times the AHBMAX will have control of the slave port is when the AHBMAX is halting or when no masters are making access requests to the slave port and the AHBMAX is forced to either park the slave port on a specific master or put the slave port into low power park mode.

In most instances when the AHBMAX has control of the slave port it will indicate IDLE for the transfer type, negate all control signals and indicate ownership of the slave bus via the hmaster encoding of b0000. One exception to this rule is when a master running locked cycles has left the slave port but continues to run locked cycles. In this case the AHBMAX will control the slave port and will indicate IDLE for the transfer type but it will not affect any other signals.

NOTE

When a master runs a locked cycle through the AHBMAX, the master will be guaranteed ownership of all slave ports it accesses while running locked cycles for one cycle beyond when the master finishes running locked cycles.

10.8 Programmable Registers

There are four registers that reside in each slave port of the AHBMAX and one register that resides in each master port of the AHBMAX. These registers are IP bus compliant registers. Read and write transfers both require two IP bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

The registers are fully decoded and an error response is returned if an unimplemented location is accessed within the AHBMAX.

The slave registers also feature a bit, which when written with a 1, will prevent the registers from being written to again. The registers will still be readable, but future write attempts will have no effect on the registers and will be terminated with an error response.

AHBMAX memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63F9_4000	Master Priority Register for Slave port n (AHBMAX_MPR0)	32	R/W	0054_3210h	10.8.1/ 584

Table continues on the next page...

AHBMAX memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63F9_4010	General Purpose Control Register for Slave port n (AHBMAX_SGPCR0)	32	R/W	0000_0000h	10.8.2/ 586
63F9_4100	Master Priority Register for Slave port n (AHBMAX_MPR1)	32	R/W	0054_3210h	10.8.1/ 584
63F9_4110	General Purpose Control Register for Slave port n (AHBMAX_SGPCR1)	32	R/W	0000_0000h	10.8.2/ 586
63F9_4200	Master Priority Register for Slave port n (AHBMAX_MPR2)	32	R/W	0054_3210h	10.8.1/ 584
63F9_4210	General Purpose Control Register for Slave port n (AHBMAX_SGPCR2)	32	R/W	0000_0000h	10.8.2/ 586
63F9_4300	Master Priority Register for Slave port n (AHBMAX_MPR3)	32	R/W	0054_3210h	10.8.1/ 584
63F9_4310	General Purpose Control Register for Slave port n (AHBMAX_SGPCR3)	32	R/W	0000_0000h	10.8.2/ 586
63F9_4800	General Purpose Control Register for Master port n (AHBMAX_MGPCR0)	32	R/W	0000_0000h	10.8.3/ 588
63F9_4900	General Purpose Control Register for Master port n (AHBMAX_MGPCR1)	32	R/W	0000_0000h	10.8.3/ 588
63F9_4A00	General Purpose Control Register for Master port n (AHBMAX_MGPCR2)	32	R/W	0000_0000h	10.8.3/ 588
63F9_4B00	General Purpose Control Register for Master port n (AHBMAX_MGPCR3)	32	R/W	0000_0000h	10.8.3/ 588
63F9_4C00	General Purpose Control Register for Master port n (AHBMAX_MGPCR4)	32	R/W	0000_0000h	10.8.3/ 588
63F9_4D00	General Purpose Control Register for Master port n (AHBMAX_MGPCR5)	32	R/W	0000_0000h	10.8.3/ 588
63F9_4E00	General Purpose Control Register for Master port n (AHBMAX_MGPCR6)	32	R/W	0000_0000h	10.8.3/ 588

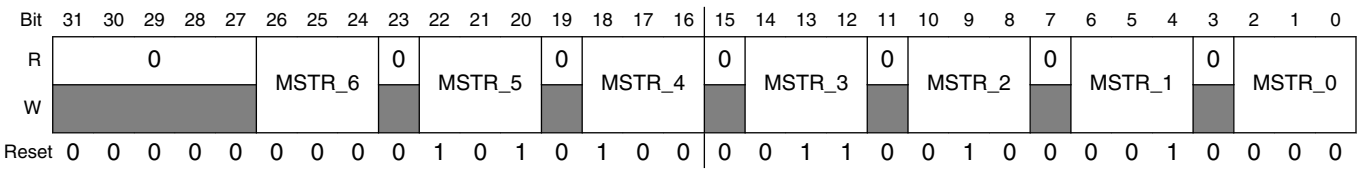
10.8.1 Master Priority Register for Slave port n (AHBMAX_MPR*n*)

The Master Priority Register (MPR) sets the priority of each master port on a per slave port basis and resides in each slave port.

The Master Priority Register can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the Slave General Purpose Control Register the Master Priority Register (MPR) can only be read from, attempts to write to it will have no effect on the MPR and result in an error response.

Additionally, no two available master ports may be programmed with the same priority level. Attempts to program two or more available masters with the same priority level will result in an error response and the MPR will not be updated.

Addresses: AHBMAX_MPR0 is 63F9_4000h base + 0h offset = 63F9_4000h
AHBMAX_MPR1 is 63F9_4000h base + 100h offset = 63F9_4100h
AHBMAX_MPR2 is 63F9_4000h base + 200h offset = 63F9_4200h
AHBMAX_MPR3 is 63F9_4000h base + 300h offset = 63F9_4300h



AHBMAX_MPR*n* field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved. They are read as zero and should be written with zero for upward compatibility.
26–24 MSTR_6	Master 6Priority. These bits set the arbitration priority for master port 6 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
23 Reserved	This read-only field is reserved and always has the value zero. Reserved. They are read as zero and should be written with zero for upward compatibility.
22–20 MSTR_5	Master 5 Priority. These bits set the arbitration priority for master port 5 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
19 Reserved	This read-only field is reserved and always has the value zero. Reserved. They are read as zero and should be written with zero for upward compatibility.
18–16 MSTR_4	Master 4 Priority. These bits set the arbitration priority for master port 4 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
15 Reserved	This read-only field is reserved and always has the value zero. Reserved. They are read as zero and should be written with zero for upward compatibility.
14–12 MSTR_3	Master 3 Priority. These bits set the arbitration priority for master port 3 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
11 Reserved	This read-only field is reserved and always has the value zero. Reserved. They are read as zero and should be written with zero for upward compatibility.

Table continues on the next page...

AHBMAX_MPR*n* field descriptions (continued)

Field	Description
10–8 MSTR_2	Master 2 Priority. These bits set the arbitration priority for master port 2 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
7 Reserved	This read-only field is reserved and always has the value zero. Reserved. They are read as zero and should be written with zero for upward compatibility.
6–4 MSTR_1	Master 1 Priority. These bits set the arbitration priority for master port 1 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
3 Reserved	This read-only field is reserved and always has the value zero. Reserved. They are read as zero and should be written with zero for upward compatibility.
2–0 MSTR_0	Master 0 Priority. These bits set the arbitration priority for master port 0 on the associated slave port. These bits are initialized by hardware reset. 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.

10.8.2 General Purpose Control Register for Slave port n (AHBMAX_SGPCR*n*)

The Slave General Purpose Control Register (AHBMAX_SGPCR*n*) controls several features of each slave port.

The Read Only (RO) bit will prevent any registers associated with this slave port from being written to once set. This bit may be written with 0 as many times as the user desires, but once it is written to a 1 only a reset condition will allow it to be written again.

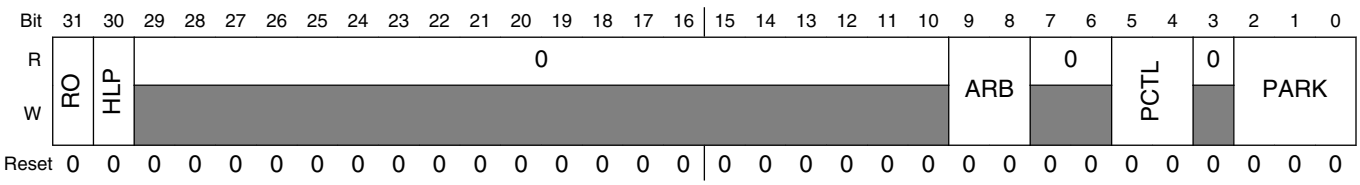
The Halt Low Priority (HLP) bit will set the priority of the max_halt_request input to the lowest possible priority for initial arbitration of the slave ports. By default it is the highest priority. Please note, setting this bit will not effect the max_halt_request from attaining highest priority once it has control of the slave ports.

The PCTL bits determine how the slave port will park when no master is actively making a request. The available options are to park on the master defined by the PARK bits, park on the last master to use the slave port, or go into a low power park mode which will force all the outputs of the slave port to inactive states when no master is requesting an access. The low power park feature can result in an overall power savings if a the slave port is not saturated; however, it will force an extra clock of latency whenever any master tries to access it when it is not in use because it will not be parked on any master.

The PARK bits determine which master the slave will park on when no master is making an active request and the max_halt_request input is negated. Please use caution to only select master ports that are actually present in the design. If the user programs the PARK bits to a master not present in the current design implementation undefined behavior will result.

The AHBMAX_SGPCR*n* can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the AHBMAX_SGPCR*n* the AHBMAX_SGPCR*n* can only be read, attempts to write to it will have no effect on the AHBMAX_SGPCR*n* and result in an error response.

Addresses: AHBMAX_SGPCR0 is 63F9_4000h base + 10h offset = 63F9_4010h
AHBMAX_SGPCR1 is 63F9_4000h base + 110h offset = 63F9_4110h
AHBMAX_SGPCR2 is 63F9_4000h base + 210h offset = 63F9_4210h
AHBMAX_SGPCR3 is 63F9_4000h base + 310h offset = 63F9_4310h



AHBMAX_SGPCR*n* field descriptions

Field	Description
31 RO	Read Only. This bit is used to force all of a slave port's registers to be read only. Once written to 1 it can only be cleared by hardware reset. This bit is initialized by hardware reset. 0 All this slave port's registers can be written. 1 All this slave port's registers are read only and cannot be written (attempted writes have no effect and result in an error response).
30 HLP	Halt Low Priority. This bit is used to set the initial arbitration priority of the max_halt_request input. This bit is initialized by hardware reset. 0 The max_halt_request input has the highest priority for arbitration on this slave port 1 The max_halt_request input has the lowest initial priority for arbitration on this slave port.
29–10 Reserved	This read-only field is reserved and always has the value zero. Reserved. They read as zero and should be written with zero for upward compatibility.
9–8 ARB	Arbitration Mode. These bits are used to select the arbitration policy for the slave port. These bits are initialized by hardware reset. 00 Fixed Priority. 01 Round Robin (rotating) Priority. 10 Reserved 11 Reserved
7–6 Reserved	This read-only field is reserved and always has the value zero. Reserved. They read as zero and should be written with zero for upward compatibility.

Table continues on the next page...

AHBMAX_SGPCR*n* field descriptions (continued)

Field	Description
5–4 PCTL	Parking Control.These bits determine the parking control used by this slave port. These bits are initialized by hardware reset. 00 When no master is making a request the arbiter will park the slave port on the master port defined by the PARK bit field. 01 When no master is making a request the arbiter will park the slave port on the last master to be in control of the slave port. 10 When no master is making a request the arbiter will park the slave port on no master and will drive all outputs to a constant safe state. 11 Reserved
3 Reserved	This read-only field is reserved and always has the value zero. Reserved. They read as zero and should be written with zero for upward compatibility.
2–0 PARK	PARK. These bits are used to determine which master port this slave port parks on when no masters are actively making requests and the PCTL bits are set to 00. These bits are initialized by hardware reset. 000 Park on Master Port 0 001 Park on Master Port 1 010 Park on Master Port 2 011 Park on Master Port 3 100 Park on Master Port 4 101 Park on Master Port 5 110 Park on Master Port 6 111 Reserved

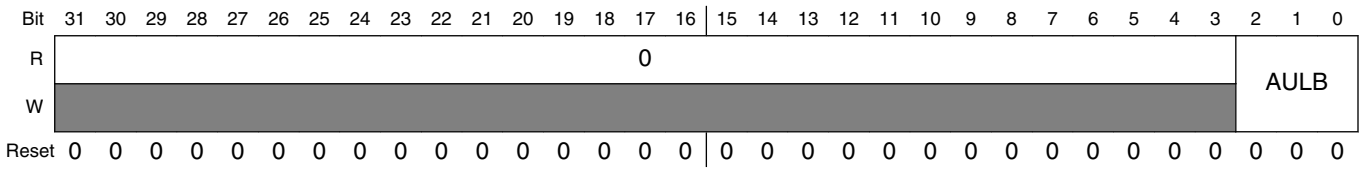
10.8.3 General Purpose Control Register for Master port n (AHBMAX_MGPCR*n*)

The Master General Purpose Control Register (AHBMAX_MGPCR*n*) presently controls only whether or not the master's undefined length burst accesses will be allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters.

The AULB (Arbitrate on Undefined Length Bursts) bit field determines whether (and when) or not the AHBMAX will arbitrate away the slave port the master owns when the master is performing undefined length burst accesses.

The AHBMAX_MGPCR*n* can only be accessed in supervisor mode with 32-bit accesses.

Addresses: AHBMAX_MGPCR0 is 63F9_4000h base + 800h offset = 63F9_4800h
AHBMAX_MGPCR1 is 63F9_4000h base + 900h offset = 63F9_4900h
AHBMAX_MGPCR2 is 63F9_4000h base + A00h offset = 63F9_4A00h
AHBMAX_MGPCR3 is 63F9_4000h base + B00h offset = 63F9_4B00h
AHBMAX_MGPCR4 is 63F9_4000h base + C00h offset = 63F9_4C00h
AHBMAX_MGPCR5 is 63F9_4000h base + D00h offset = 63F9_4D00h
AHBMAX_MGPCR6 is 63F9_4000h base + E00h offset = 63F9_4E00h



AHBMAX_MGPCR*n* field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value zero. Reserved. They read as zero and should be written with zero for upward compatibility.
2–0 AULB	Arbitrate on Undefined Length Bursts. These bits are used to select the arbitration policy during undefined length bursts by this master. These bits are initialized by hardware reset. 000 No arbitration will be allowed during an undefined length burst. 001 Arbitration will be allowed at any time during an undefined length burst. 010 Arbitration will be allowed after four beats of an undefined length burst. 011 Arbitration will be allowed after eight beats of an undefined length burst. 100 Arbitration will be allowed after 16 beats of an undefined length burst. 101 Reserved 110 Reserved 111 Reserved

Chapter 11
AHB-to-APBH Bridge with DMA (APBH-Bridge-DMA)

11.1 Overview

The AHB-to-APBH bridge provides the i.MX50 with an inexpensive peripheral attachment bus running on the AHB's HCLK.

(The H in APBH denotes that the APB*H* is synchronous to HCLK, as compared to APB*X*, which runs on the crystal-derived XCLK.)

As shown in the figure below, the AHB-to-APBH bridge includes the AHB-to-APB PIO bridge for a memory-mapped I/O to the APB devices, as well as a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM Cortex-A8 core. Each one of the APB peripherals, including the vectored interrupt controller, is documented in their respective chapters.

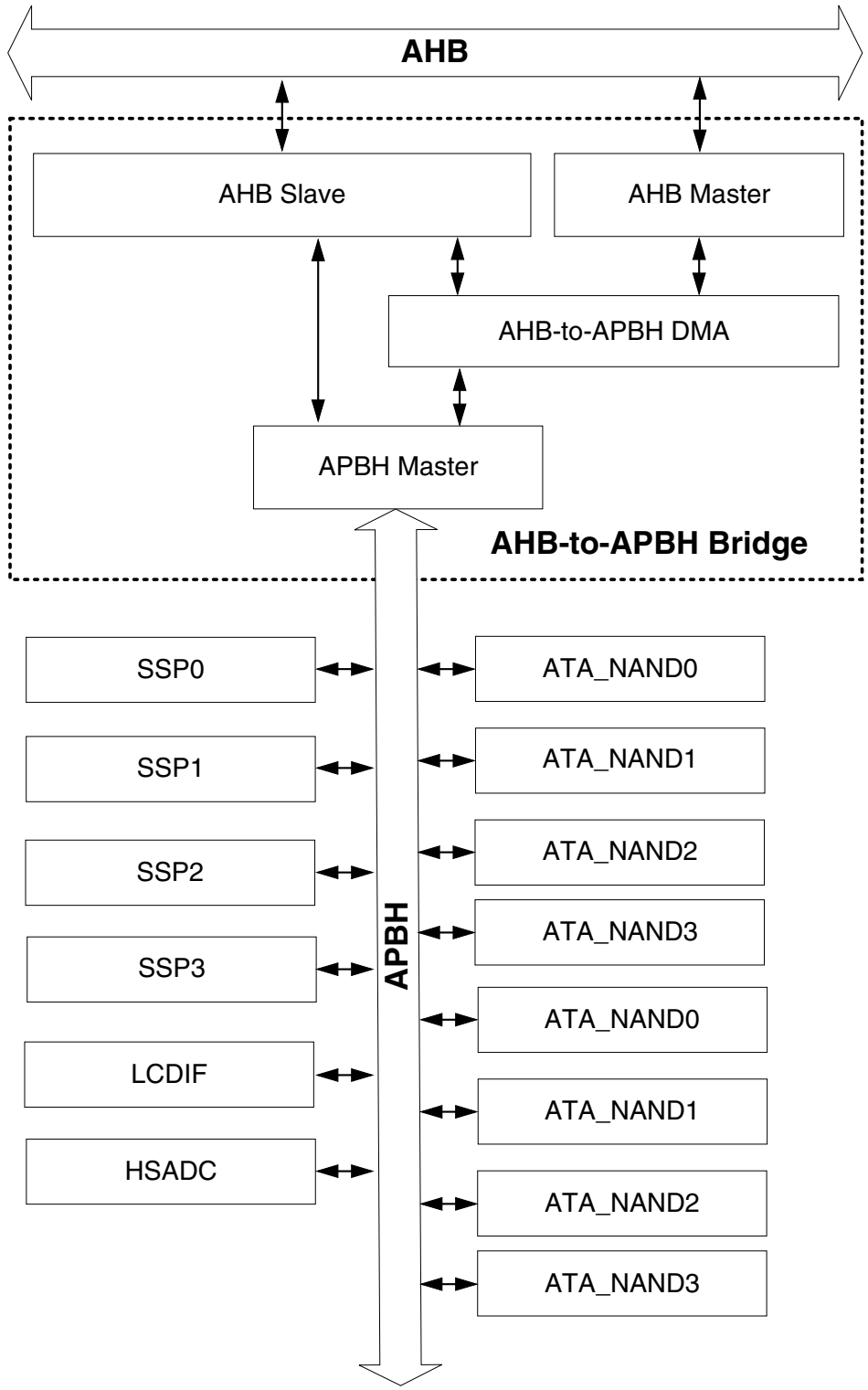


Figure 11-1. AHB-to-APBH Bridge DMA Block Diagram

The DMA controller uses the APBH bus to transfer read and write data to and from each peripheral. There is no separate DMA bus for these devices. Contention between the DMA's use of the APBH bus and the AHB-to-APB bridge functions' use of the APBH is mediated by an internal arbitration logic. For contention between these two units, the

DMA is favored and the AHB slave will report "not ready" through its HREADY output until the bridge transfer can complete. The arbiter tracks repeated lockouts and inverts the priority, guaranteeing the ARM platform every fourth transfer on the APB.

11.2 APBH DMA

The DMA supports sixteen channels of DMA services, as shown in the following table. The shared DMA resource allows each independent channel to follow a simple chained command list.

Command chains are built up using the general structure, as shown in Figure 11-2.

Table 11-1. APBH DMA Channel Assignments

APBH DMA CHANNEL #	USAGE
0	Reserved
1	Reserved
2	Reserved
3	Reserved
4	GPMI0
5	GPMI1
6	GPMI2
7	GPMI3
8	GPMI4
9	GPMI5
10	GPMI6
11	GPMI7
12	HSADC
13	LCDIF
14	Empty
15	Empty

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the ARM platform can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA, and have no further concern for the device until the DMA completion interrupt occurs. The goal is to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1 KHz (arrival intervals longer than 1 ms).

A single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and the controls, it uses to write DMA data bytes to the device. For example, this allows a chain of operations to be issued to the GPMI controller to send NAND command bytes, address bytes, and data transfers where the command and the address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA. Each DMA structure can have 0–15 PIO words appended to it. The #PIOWORDs field, if non-zero, instructs the DMA engine to copy these words to the APB, beginning at PADDR = 0x0000 and incrementing its PADDR for each cycle.

The DMA master generates only normal read/write transfers to the APBH. It does *not* generate set, clear, or toggle (SCT) transfers.

After any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure. Table 11-2 shows the four commands implemented by the DMA.

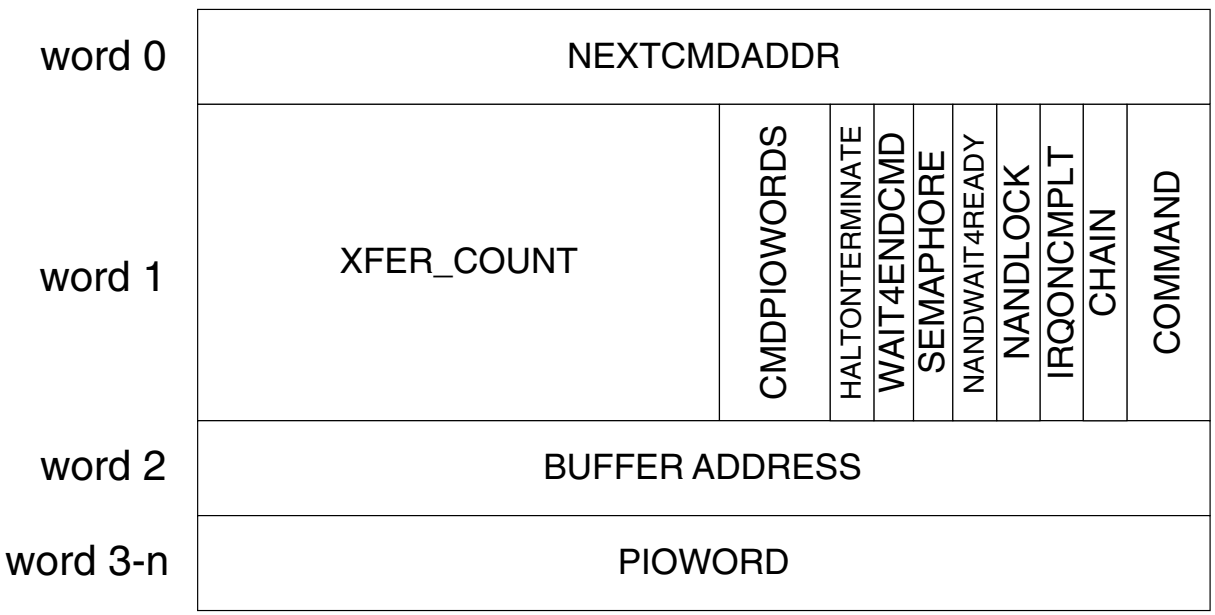


Figure 11-2. AHB-to-APBH Bridge DMA Channel Command Structure

Table 11-2. APBH DMA Commands

DMA COMMAND	USAGE
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, then perform a DMA transfer from the peripheral for the specified number of bytes.
10	DMA_READ. Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

Table continues on the next page...

Table 11-2. APBH DMA Commands (continued)

DMA COMMAND	USAGE
11	DMA_SENSE. Perform any requested PIO word transfers, then perform a conditional branch to the next chained device. Follow the NEXTCMD_ADDR pointer if the peripheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true. This command becomes a no-operation for any channel other than a GPMI channel.

DMA_WRITE operations copy data bytes to the system memory (on-chip RAM or SDRAM) from the associated peripheral. Each peripheral has a target PADDR value that it expects to receive DMA bytes. This association is synthesized in the DMA. The DMA_WRITE transfer uses the BUFFER_ADDRESS word in the command structure to point to the beginning byte to write data from the peripheral.

DMA_READ operations copy data bytes to the APB peripheral from the system memory. The DMA engine contains a shared byte aligner that aligns bytes from system memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers. This command is useful in such applications as activating the NAND devices CHECKSTATUS operation. The check status command in the NAND peripheral reads a status byte from the NAND device, performs an XOR and MASK against an expected value supplied as part of the PIO transfer. Once the read check completes (see NAND Read Status Polling Example),the NO_DMA_XFER command completes. The result in the peripheral is that its PSENSE line is driven by the results of the comparison. The sense flip-flop is only updated by CHECKSTATUS for the device that is executed. At some future point, the chain contains a DMA command structure with the fourth and final command value, that is, the DMA_SENSE command.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. The DMA_SENSE command uses the DMA buffer pointer word of the command structure to point to an alternate DMA command structure chain or list. The DMA_SENSE command examines the sense line of the associated peripheral. If the sense line is false, then the DMA follows the standard list found whose next command is found from the pointer in the NEXTCMD_ADDR word of the command structure. If the sense line is true, then the DMA follows the alternate list whose next command is found from the pointer in the DMA Buffer Pointer word of the DMA_SENSE command structure (see Figure 11-2). The sense command ignores the CHAIN bit, so that both pointers must be valid when the DMA comes to a sense command.

If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, then the DMA channel waits for the device to signal completion of a command by toggling the endcmcd signal before proceeding to load and execute the next command structure. Then, if DECREMENT_SEMAPHORE is set, the semaphore will be decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in the following table, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer-count mechanism is duplicated in the associated peripheral, either as an implied or as a specified count in the peripheral.

Table 11-3. DMA Channel Command Word in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															
Number DMA Bytes to Transfer																Number PIO Words to Write						HALTONTERMINATE	WAIT4ENDCMD	DECREMENT SEMAPHORE	NANDwAIT4READY	NANDLOCK	IRQ_COMPLETE	CHAIN	COMMAND		
DMA Buffer or Alternate CCW																															
Zero or More PIO Words to Write to the Associated Peripheral Starting at its Base Address on the APBH Bus																															

Figure 11-2 also shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1, if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) is loaded into the NEXT_COMMAND_ADDRESS, it is not detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_COMPLETE bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt-status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by the software. It can be used to interrupt the ARM platform.

The NAND_LOCK bit is monitored by the DMA channel arbiter. After a NAND channel (from channel 4 to channel 11) succeeds in the arbiter with its NAND_LOCK bit set, then the arbiter ignores the other seven NAND channels until a command is completed in which the NAND_LOCK is not set. Notice that the semantic here is that the NAND_LOCK state is to limit scheduling of a non-locked DMA. A DMA channel can go from unlocked to locked in the arbiter at the beginning of a command when the

NAND_LOCK bit is set. When the last DMA command of an atomic sequence is completed, the lock should be removed. To accomplish this, the last command does not have the NAND_LOCK bit. It is still locked in the atomic state within the arbiter when the command starts, so that it is the only NAND command that can be executed. At the end, it drops from the atomic state within the arbiter.

The NAND_WAIT4READY bit also has a special use for DMA channels (from channel 4 to channel 11), i.e., the NAND device channels. The NAND device supplies a sample of the ready line for the NAND device. This ready value is used to hold off of a command with this bit set until the ready line is asserted to 1. Once the arbiter sees a command with a wait-for-ready set, it holds off that channel until ready is asserted.

Receiving an IRQ for HALTONTERMINATE (HOT) is a new feature in the APBH DMA descriptor that allows GPMI (as well as SSP and I2C) to signal to the DMA engine that an error has occurred. In prior chips, if a block is stalled due to an error, the only practical way to discover this in software was through a timer of some sort, or to poll the block. Now, an HOT signal is sent from the peripheral to the DMA engine and causes an IRQ after terminating the DMA descriptor being executed. Note not all peripheral block support this termination feature.

Therefore, it is recommended that software use this signal as follows:

- Always set HALTONTERMINATE to 1 in a DMA descriptor. That way, if a peripheral signals HOT, the transfer will end, leaving the peripheral block and the DMA engine synchronized (but at the end of a command).
- When an IRQ from an APBH channel is received, and the IRQ is determined to be due to an error (as opposed to an IRQONCOMPLETE interrupt) the software should:
    - Reset the channel.
    - Determine the error from error reporting in the peripheral block, then manage the error in the peripheral that is attached to that channel in whatever appropriate way exists for that device (software recovery, device reset, block reset, etc).

Each channel has an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run, process commands and perform DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the IDLE state and remains there until the semaphore is incremented by the software. When the semaphore goes to non-zero and the channel is in its IDLE state, then it uses the value in the APBH_CHn_NXTCMDAR register (next command address register) to fetch a

pointer to the next command to process. NOTE: This is a double indirect case. This method allows the software to append to a running command list under the protection of the counting semaphore.

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the APBH_CHn_NXTCMDAR register, and then writes 1 to the counting semaphore in APBH_CHn_SEMA. The DMA channel loads APBH_CHn_CURCMDAR register and then enters the normal state machine processing for the next command. When the software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of APBH_CHn_CURCMDAR at any time to determine the location of the command structure currently being processed.

11.3 NAND Read Status Polling Example

The following figure shows a more complicated scenario. This subset of a NAND device workload shows that the first two command structures are used during the data-write phase of an NAND device write operation (CLE and ALE transfers omitted for clarity).

- After writing the data, one must wait until the NAND device status register indicates that the write charge has been transferred. This is built into the workload using a check status command in the NAND in a loop created from the next two DMA command structures.
- The NO_DMA_TRANSFER command is shown here performing the read check, followed by a DMA_SENSE command to branch the DMA command structure list, based on the status of a bit in the external NAND device.

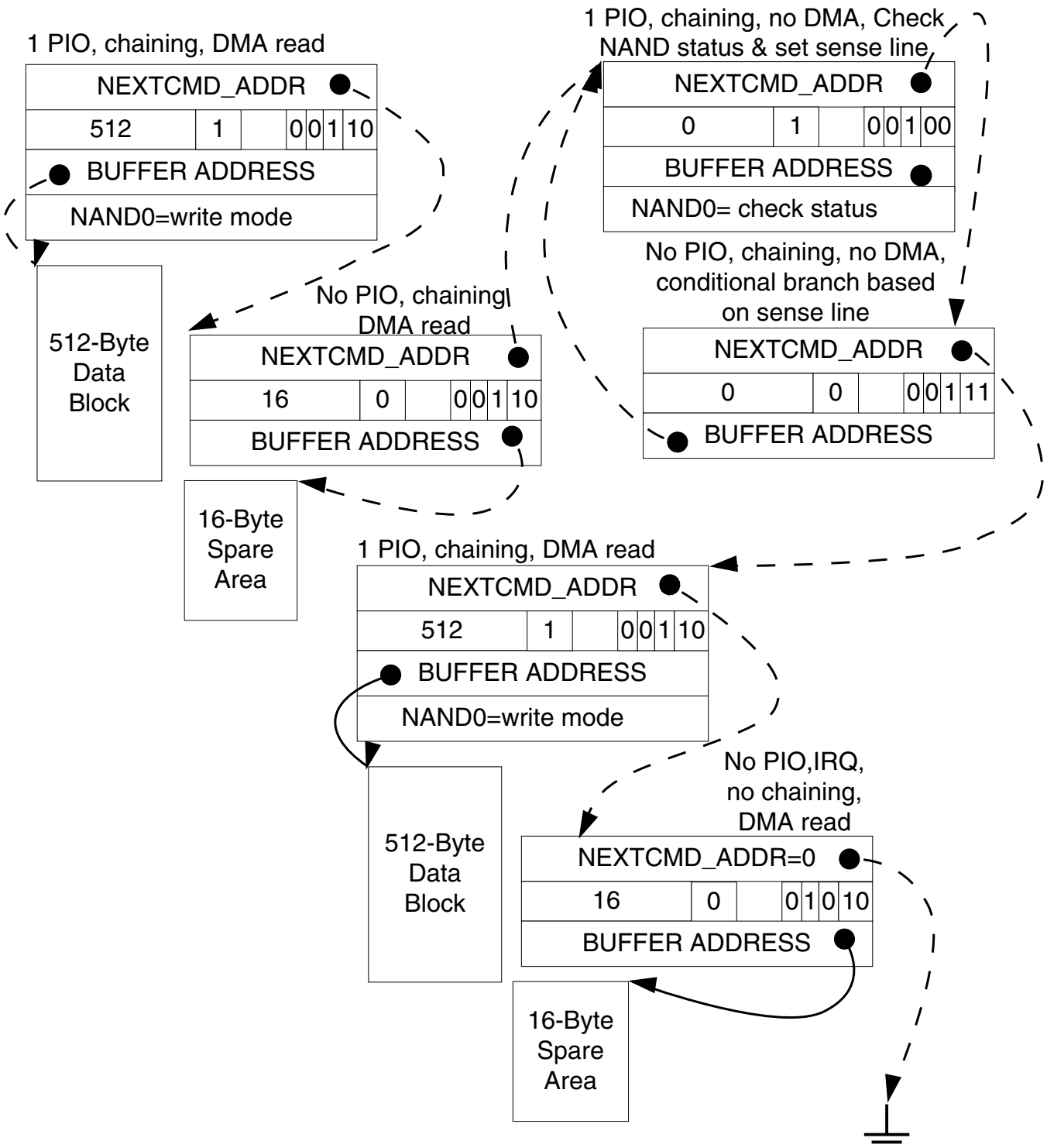


Figure 11-3. AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command

The example in the above figure shows the workload continuing immediately to the next NAND page transfer. However, one could perform a second sense operation to see if an error has occurred after the write. One could then point the sense command alternate branch at a NO_DMA_XFER command with the interrupt bit set. If the CHAIN bit is not

set on this failure branch, then the ARM platform is interrupted immediately, and the channel process is also immediately terminated in the presence of a workload-detected NAND error bit.

Note that each word of the three-word DMA command structure corresponds to a PIO register of the DMA that is accessible on the APBH bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register.

To start DMA processing for the first command, initialize the PIO registers of the desired channel, as follows:

- First, load the next command address register with a pointer to the first command to be loaded.
- Then, write 1 to the counting semaphore register. This causes the DMA to schedule the targeted channel for the DMA command structure load, as if it just finished its previous command.

11.4 Programmable Registers

APBH Hardware Register Format Summary

APBH memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_0000	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0)	32	R/W	E000_0000h	11.4.1/ 607
4100_0004	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_SET)	32	R/W	E000_0000h	11.4.1/ 607
4100_0008	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_CLR)	32	R/W	E000_0000h	11.4.1/ 607
4100_000C	AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0_TOG)	32	R/W	E000_0000h	11.4.1/ 607
4100_0010	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1)	32	R/W	0000_0000h	11.4.2/ 608
4100_0014	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_SET)	32	R/W	0000_0000h	11.4.2/ 608

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_0018	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_CLR)	32	R/W	0000_0000h	11.4.2/ 608
4100_001C	AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1_TOG)	32	R/W	0000_0000h	11.4.2/ 608
4100_0020	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2)	32	R/W	0000_0000h	11.4.3/ 612
4100_0024	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_SET)	32	R/W	0000_0000h	11.4.3/ 612
4100_0028	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_CLR)	32	R/W	0000_0000h	11.4.3/ 612
4100_002C	AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2_TOG)	32	R/W	0000_0000h	11.4.3/ 612
4100_0030	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL)	32	R/W	0000_0000h	11.4.4/ 617
4100_0034	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_SET)	32	R/W	0000_0000h	11.4.4/ 617
4100_0038	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_CLR)	32	R/W	0000_0000h	11.4.4/ 617
4100_003C	AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL_TOG)	32	R/W	0000_0000h	11.4.4/ 617
4100_0040	AHB to APBH DMA Device Assignment Register (APBH_DEVSEL)	32	R/W	0000_0000h	11.4.5/ 618
4100_0050	AHB to APBH DMA burst size (APBH_DMA_BURST_SIZE)	32	R/W	0055_5555h	11.4.6/ 619
4100_0060	AHB to APBH DMA Debug Register (APBH_DEBUG)	32	R/W	0000_0000h	11.4.7/ 620
4100_0100	APBH DMA Channel n Current Command Address Register (APBH_CH_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0110	APBH DMA Channel n Next Command Address Register (APBH_CH0_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0120	APBH DMA Channel n Command Register (APBH_CH0_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0130	APBH DMA Channel n Buffer Address Register (APBH_CH0_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0140	APBH DMA Channel n Semaphore Register (APBH_CH0_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0150	AHB to APBH DMA Channel n Debug Information (APBH_CH0_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0160	AHB to APBH DMA Channel n Debug Information (APBH_CH0_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_0170	APBH DMA Channel n Current Command Address Register (APBH_CH0_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0180	APBH DMA Channel n Next Command Address Register (APBH_CH1_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0190	APBH DMA Channel n Command Register (APBH_CH1_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_01A0	APBH DMA Channel n Buffer Address Register (APBH_CH1_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_01B0	APBH DMA Channel n Semaphore Register (APBH_CH1_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_01C0	AHB to APBH DMA Channel n Debug Information (APBH_CH1_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_01D0	AHB to APBH DMA Channel n Debug Information (APBH_CH1_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_01E0	APBH DMA Channel n Current Command Address Register (APBH_CH1_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_01F0	APBH DMA Channel n Next Command Address Register (APBH_CH2_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0200	APBH DMA Channel n Command Register (APBH_CH2_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0210	APBH DMA Channel n Buffer Address Register (APBH_CH2_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0220	APBH DMA Channel n Semaphore Register (APBH_CH2_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0230	AHB to APBH DMA Channel n Debug Information (APBH_CH2_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0240	AHB to APBH DMA Channel n Debug Information (APBH_CH2_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0250	APBH DMA Channel n Current Command Address Register (APBH_CH2_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0260	APBH DMA Channel n Next Command Address Register (APBH_CH3_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0270	APBH DMA Channel n Command Register (APBH_CH3_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0280	APBH DMA Channel n Buffer Address Register (APBH_CH3_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0290	APBH DMA Channel n Semaphore Register (APBH_CH3_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_02A0	AHB to APBH DMA Channel n Debug Information (APBH_CH3_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_02B0	AHB to APBH DMA Channel n Debug Information (APBH_CH3_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_02C0	APBH DMA Channel n Current Command Address Register (APBH_CH3_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_02D0	APBH DMA Channel n Next Command Address Register (APBH_CH4_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_02E0	APBH DMA Channel n Command Register (APBH_CH4_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_02F0	APBH DMA Channel n Buffer Address Register (APBH_CH4_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0300	APBH DMA Channel n Semaphore Register (APBH_CH4_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0310	AHB to APBH DMA Channel n Debug Information (APBH_CH4_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0320	AHB to APBH DMA Channel n Debug Information (APBH_CH4_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0330	APBH DMA Channel n Current Command Address Register (APBH_CH4_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0340	APBH DMA Channel n Next Command Address Register (APBH_CH5_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0350	APBH DMA Channel n Command Register (APBH_CH5_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0360	APBH DMA Channel n Buffer Address Register (APBH_CH5_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0370	APBH DMA Channel n Semaphore Register (APBH_CH5_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0380	AHB to APBH DMA Channel n Debug Information (APBH_CH5_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0390	AHB to APBH DMA Channel n Debug Information (APBH_CH5_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_03A0	APBH DMA Channel n Current Command Address Register (APBH_CH5_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_03B0	APBH DMA Channel n Next Command Address Register (APBH_CH6_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_03C0	APBH DMA Channel n Command Register (APBH_CH6_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_03D0	APBH DMA Channel n Buffer Address Register (APBH_CH6_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_03E0	APBH DMA Channel n Semaphore Register (APBH_CH6_SEMA)	32	R/W	0000_0000h	11.4.12/ 626

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_03F0	AHB to APBH DMA Channel n Debug Information (APBH_CH6_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0400	AHB to APBH DMA Channel n Debug Information (APBH_CH6_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0410	APBH DMA Channel n Current Command Address Register (APBH_CH6_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0420	APBH DMA Channel n Next Command Address Register (APBH_CH7_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0430	APBH DMA Channel n Command Register (APBH_CH7_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0440	APBH DMA Channel n Buffer Address Register (APBH_CH7_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0450	APBH DMA Channel n Semaphore Register (APBH_CH7_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0460	AHB to APBH DMA Channel n Debug Information (APBH_CH7_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0470	AHB to APBH DMA Channel n Debug Information (APBH_CH7_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0480	APBH DMA Channel n Current Command Address Register (APBH_CH7_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0490	APBH DMA Channel n Next Command Address Register (APBH_CH8_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_04A0	APBH DMA Channel n Command Register (APBH_CH8_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_04B0	APBH DMA Channel n Buffer Address Register (APBH_CH8_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_04C0	APBH DMA Channel n Semaphore Register (APBH_CH8_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_04D0	AHB to APBH DMA Channel n Debug Information (APBH_CH8_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_04E0	AHB to APBH DMA Channel n Debug Information (APBH_CH8_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_04F0	APBH DMA Channel n Current Command Address Register (APBH_CH8_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0500	APBH DMA Channel n Next Command Address Register (APBH_CH9_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0510	APBH DMA Channel n Command Register (APBH_CH9_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0520	APBH DMA Channel n Buffer Address Register (APBH_CH9_BAR)	32	R/W	0000_0000h	11.4.11/ 625

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_0530	APBH DMA Channel n Semaphore Register (APBH_CH9_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0540	AHB to APBH DMA Channel n Debug Information (APBH_CH9_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0550	AHB to APBH DMA Channel n Debug Information (APBH_CH9_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0560	APBH DMA Channel n Current Command Address Register (APBH_CH9_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0570	APBH DMA Channel n Next Command Address Register (APBH_CH10_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0580	APBH DMA Channel n Command Register (APBH_CH10_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0590	APBH DMA Channel n Buffer Address Register (APBH_CH10_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_05A0	APBH DMA Channel n Semaphore Register (APBH_CH10_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_05B0	AHB to APBH DMA Channel n Debug Information (APBH_CH10_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_05C0	AHB to APBH DMA Channel n Debug Information (APBH_CH10_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_05D0	APBH DMA Channel n Current Command Address Register (APBH_CH10_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_05E0	APBH DMA Channel n Next Command Address Register (APBH_CH11_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_05F0	APBH DMA Channel n Command Register (APBH_CH11_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0600	APBH DMA Channel n Buffer Address Register (APBH_CH11_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0610	APBH DMA Channel n Semaphore Register (APBH_CH11_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0620	AHB to APBH DMA Channel n Debug Information (APBH_CH11_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0630	AHB to APBH DMA Channel n Debug Information (APBH_CH11_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0640	APBH DMA Channel n Current Command Address Register (APBH_CH11_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0650	APBH DMA Channel n Next Command Address Register (APBH_CH12_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0660	APBH DMA Channel n Command Register (APBH_CH12_CMD)	32	R/W	0000_0000h	11.4.10/ 623

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_0670	APBH DMA Channel n Buffer Address Register (APBH_CH12_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0680	APBH DMA Channel n Semaphore Register (APBH_CH12_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0690	AHB to APBH DMA Channel n Debug Information (APBH_CH12_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_06A0	AHB to APBH DMA Channel n Debug Information (APBH_CH12_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_06B0	APBH DMA Channel n Current Command Address Register (APBH_CH12_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_06C0	APBH DMA Channel n Next Command Address Register (APBH_CH13_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_06D0	APBH DMA Channel n Command Register (APBH_CH13_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_06E0	APBH DMA Channel n Buffer Address Register (APBH_CH13_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_06F0	APBH DMA Channel n Semaphore Register (APBH_CH13_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0700	AHB to APBH DMA Channel n Debug Information (APBH_CH13_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0710	AHB to APBH DMA Channel n Debug Information (APBH_CH13_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0720	APBH DMA Channel n Current Command Address Register (APBH_CH13_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_0730	APBH DMA Channel n Next Command Address Register (APBH_CH14_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622
4100_0740	APBH DMA Channel n Command Register (APBH_CH14_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_0750	APBH DMA Channel n Buffer Address Register (APBH_CH14_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_0760	APBH DMA Channel n Semaphore Register (APBH_CH14_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_0770	AHB to APBH DMA Channel n Debug Information (APBH_CH14_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_0780	AHB to APBH DMA Channel n Debug Information (APBH_CH14_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0790	APBH DMA Channel n Current Command Address Register (APBH_CH14_CURCMDAR)	32	R/W	0000_0000h	11.4.8/ 621
4100_07A0	APBH DMA Channel n Next Command Address Register (APBH_CH15_NXTCMDAR)	32	R/W	0000_0000h	11.4.9/ 622

Table continues on the next page...

APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_07B0	APBH DMA Channel n Command Register (APBH_CH15_CMD)	32	R/W	0000_0000h	11.4.10/ 623
4100_07C0	APBH DMA Channel n Buffer Address Register (APBH_CH15_BAR)	32	R/W	0000_0000h	11.4.11/ 625
4100_07D0	APBH DMA Channel n Semaphore Register (APBH_CH15_SEMA)	32	R/W	0000_0000h	11.4.12/ 626
4100_07E0	AHB to APBH DMA Channel n Debug Information (APBH_CH15_DEBUG1)	32	R/W	00A0_0000h	11.4.13/ 627
4100_07F0	AHB to APBH DMA Channel n Debug Information (APBH_CH15_DEBUG2)	32	R/W	0000_0000h	11.4.14/ 630
4100_0800	APBH Bridge Version Register (APBH_VERSION)	32	R/W	0301_0000h	11.4.15/ 630

11.4.1 AHB to APBH Bridge Control and Status Register 0 (APBH_CTRL0*n*)

The APBH CTRL 0 provides overall control of the AHB to APBH bridge and DMA.

This register contains module softreset, clock gating, channel clock gating/freeze bits.

Addresses: APBH_CTRL0 is 4100_0000h base + 0h offset = 4100_0000h

APBH_CTRL0_SET is 4100_0000h base + 4h offset = 4100_0004h

APBH_CTRL0_CLR is 4100_0000h base + 8h offset = 4100_0008h

APBH_CTRL0_TOG is 4100_0000h base + Ch offset = 4100_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFTRST	CLKGATE	AHB_ BURST8_EN	APB_ BURST_EN	RSVD0											
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CLKGATE_CHANNEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CTRL0*n* field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal APBH DMA operation. Set this bit to one (default) to disable clocking with the APBH DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBH DMA block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 AHB_BURST8_EN	Set this bit to one (default) to enable AHB 8-beat burst. Set to zero to disable 8-beat burst on AHB interface.
28 APB_BURST_EN	Set this bit to one to enable apb master do a continous transfers when a device request a burst dma. Set to zero will treat a burst dma request as 4/8 individual requests.
27–16 RSVD0	Reserved, always set to zero.
15–0 CLKGATE_CHANNEL	These bits must be set to zero for normal operation of each channel. When set to one they gate off the individual clocks to the channels. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** — 0x0080 **NAND7** — 0x0100 **SSP** —

11.4.2 AHB to APBH Bridge Control and Status Register 1 (APBH_CTRL1*n*)

The APBH CTRL one provides overall control of the interrupts generated by the AHB to APBH DMA. This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
                    BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
                    BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register
struct's bitfield
```

Addresses: APBH_CTRL1 is 4100_0000h base + 10h offset = 4100_0010h

APBH_CTRL1_SET is 4100_0000h base + 14h offset = 4100_0014h

APBH_CTRL1_CLR is 4100_0000h base + 18h offset = 4100_0018h

APBH_CTRL1_TOG is 4100_0000h base + 1Ch offset = 4100_001Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CH15_CMDCMPLT_IRQ_EN	CH14_CMDCMPLT_IRQ_EN	CH13_CMDCMPLT_IRQ_EN	CH12_CMDCMPLT_IRQ_EN	CH11_CMDCMPLT_IRQ_EN	CH10_CMDCMPLT_IRQ_EN	CH9_CMDCMPLT_IRQ_EN	CH8_CMDCMPLT_IRQ_EN	CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH15_CMDCMPLT_IRQ	CH14_CMDCMPLT_IRQ	CH13_CMDCMPLT_IRQ	CH12_CMDCMPLT_IRQ	CH11_CMDCMPLT_IRQ	CH10_CMDCMPLT_IRQ	CH9_CMDCMPLT_IRQ	CH8_CMDCMPLT_IRQ	CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CTRL1*n* field descriptions

Field	Description
31 CH15_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 15.
30 CH14_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 14.
29 CH13_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 13.
28 CH12_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 12.
27 CH11_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 11.
26 CH10_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 10.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
25 CH9_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 9.
24 CH8_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 8.
23 CH7_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 7.
22 CH6_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 6.
21 CH5_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 5.
20 CH4_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 4.
19 CH3_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 3.
18 CH2_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 2.
17 CH1_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 1.
16 CH0_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 0.
15 CH15_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
14 CH14_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
13 CH13_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
12 CH12_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
11 CH11_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
10 CH10_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
9 CH9_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
8 CH8_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
7 CH7_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6 CH6_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
5 CH5_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
4 CH4_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3 CH3_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
2 CH2_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table continues on the next page...

APBH_CTRL1*n* field descriptions (continued)

Field	Description
1 CH1_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0 CH0_ CMDCMPLT_ IRQ	Interrupt request status bit for APBH DMA channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

11.4.3 AHB to APBH Bridge Control and Status Register 2 (APBH_CTRL2*n*)

The APBH CTRL 2 provides channel error interrupts generated by the AHB to APBH DMA. This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
                    BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
                    BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register
struct's bitfield
```

Addresses: APBH_CTRL2 is 4100_0000h base + 20h offset = 4100_0020h

APBH_CTRL2_SET is 4100_0000h base + 24h offset = 4100_0024h

APBH_CTRL2_CLR is 4100_0000h base + 28h offset = 4100_0028h

APBH_CTRL2_TOG is 4100_0000h base + 2Ch offset = 4100_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CH15_ERROR_STATUS	CH14_ERROR_STATUS	CH13_ERROR_STATUS	CH12_ERROR_STATUS	CH11_ERROR_STATUS	CH10_ERROR_STATUS	CH9_ERROR_STATUS	CH8_ERROR_STATUS	CH7_ERROR_STATUS	CH6_ERROR_STATUS	CH5_ERROR_STATUS	CH4_ERROR_STATUS	CH3_ERROR_STATUS	CH2_ERROR_STATUS	CH1_ERROR_STATUS	CH0_ERROR_STATUS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15_ERROR_IRQ	CH14_ERROR_IRQ	CH13_ERROR_IRQ	CH12_ERROR_IRQ	CH11_ERROR_IRQ	CH10_ERROR_IRQ	CH9_ERROR_IRQ	CH8_ERROR_IRQ	CH7_ERROR_IRQ	CH6_ERROR_IRQ	CH5_ERROR_IRQ	CH4_ERROR_IRQ	CH3_ERROR_IRQ	CH2_ERROR_IRQ	CH1_ERROR_IRQ	CH0_ERROR_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CTRL2*n* field descriptions

Field	Description
31 CH15_ERROR_STATUS	Error status bit for APBH DMA Channel 15. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
30 CH14_ERROR_STATUS	Error status bit for APBH DMA Channel 14. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
29 CH13_ERROR_STATUS	Error status bit for APBH DMA Channel 13. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
28 CH12_ERROR_STATUS	Error status bit for APBH DMA Channel 12. Valid when corresponding Error IRQ is set. 1 - AHB bus error

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
	0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
27 CH11_ERROR_ STATUS	Error status bit for APBH DMA Channel 11. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
26 CH10_ERROR_ STATUS	Error status bit for APBH DMA Channel 10. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
25 CH9_ERROR_ STATUS	Error status bit for APBH DMA Channel 9. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
24 CH8_ERROR_ STATUS	Error status bit for APBH DMA Channel 8. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
23 CH7_ERROR_ STATUS	Error status bit for APBX DMA Channel 7. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
22 CH6_ERROR_ STATUS	Error status bit for APBX DMA Channel 6. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
21 CH5_ERROR_ STATUS	Error status bit for APBX DMA Channel 5. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination.

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
	0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
20 CH4_ERROR_STATUS	Error status bit for APBX DMA Channel 4. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
19 CH3_ERROR_STATUS	Error status bit for APBX DMA Channel 3. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
18 CH2_ERROR_STATUS	Error status bit for APBX DMA Channel 2. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
17 CH1_ERROR_STATUS	Error status bit for APBX DMA Channel 1. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
16 CH0_ERROR_STATUS	Error status bit for APBX DMA Channel 0. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
15 CH15_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
14 CH14_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
13 CH13_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
12 CH12_ERROR_IRQ	Error interrupt status bit for APBH DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

Table continues on the next page...

APBH_CTRL2*n* field descriptions (continued)

Field	Description
11 CH11_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
10 CH10_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
9 CH9_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
8 CH8_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
7 CH7_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
6 CH6_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
5 CH5_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
4 CH4_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
3 CH3_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
2 CH2_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
1 CH1_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
0 CH0_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

11.4.4 AHB to APBH Bridge Channel Register (APBH_CHANNEL_CTRL*n*)

The APBH CHANNEL CTRL provides reset/freeze control of each DMA channel. This register contains individual channel reset/freeze bits.

Addresses: APBH_CHANNEL_CTRL is 4100_0000h base + 30h offset = 4100_0030h
APBH_CHANNEL_CTRL_SET is 4100_0000h base + 34h offset = 4100_0034h
APBH_CHANNEL_CTRL_CLR is 4100_0000h base + 38h offset = 4100_0038h
APBH_CHANNEL_CTRL_TOG is 4100_0000h base + 3Ch offset = 4100_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RESET_CHANNEL																FREEZE_CHANNEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CHANNEL_CTRL*n* field descriptions

Field	Description
31–16 RESET_ CHANNEL	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** — 0x0080 **NAND7** — 0x0100 **SSP** —
15–0 FREEZE_ CHANNEL	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is deined access to the central DMA resources. 0x0001 **NAND0** — 0x0002 **NAND1** — 0x0004 **NAND2** — 0x0008 **NAND3** — 0x0010 **NAND4** — 0x0020 **NAND5** — 0x0040 **NAND6** — 0x0080 **NAND7** — 0x0100 **SSP** —

11.4.5 AHB to APBH DMA Device Assignment Register (APBH_DEVSEL)

This register allows reassignment of the APBH device connected to the DMA Channels.

In this chip, apbhdma channel resource is enough for high speed peripherals, so this register is of no use and reserved.

Address: APBH_DEVSEL is 4100_0000h base + 40h offset = 4100_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_DEVSEL field descriptions

Field	Description
31–30 CH15	Reserved.
29–28 CH14	Reserved.
27–26 CH13	Reserved.
25–24 CH12	Reserved.
23–22 CH11	Reserved.
21–20 CH10	Reserved.
19–18 CH9	Reserved.
17–16 CH8	Reserved.
15–14 CH7	Reserved.
13–12 CH6	Reserved.
11–10 CH5	Reserved.

Table continues on the next page...

APBH_DEVSEL field descriptions (continued)

Field	Description
9–8 CH4	Reserved.
7–6 CH3	Reserved.
5–4 CH2	Reserved.
3–2 CH1	Reserved.
1–0 CH0	Reserved.

11.4.6 AHB to APBH DMA burst size (APBH_DMA_BURST_SIZE)

This register programs the apbh burst size of the APBH DMA devices when a DMA burst request is issued.

This register provides a mechanism for assigning the device. N/A in this version.

Address: APBH_DMA_BURST_SIZE is 4100_0000h base + 50h offset = 4100_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8	
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

APBH_DMA_BURST_SIZE field descriptions

Field	Description
31–30 CH15	Reserved.
29–28 CH14	Reserved.
27–26 CH13	Reserved.
25–24 CH12	Reserved.

Table continues on the next page...

APBH_DMA_BURST_SIZE field descriptions (continued)

Field	Description
23–22 CH11	Reserved.
21–20 CH10	Reserved.
19–18 CH9	Reserved.
17–16 CH8	DMA burst size for SSP. 0x0 **BURST0** — 0x1 **BURST4** — 0x2 **BURST8** —
15–14 CH7	DMA burst size for GPMI channel 7. Do not change. GPMI only support burst size 4.
13–12 CH6	DMA burst size for GPMI channel 6. Do not change. GPMI only support burst size 4.
11–10 CH5	DMA burst size for GPMI channel 5. Do not change. GPMI only support burst size 4.
9–8 CH4	DMA burst size for GPMI channel 4. Do not change. GPMI only support burst size 4.
7–6 CH3	DMA burst size for GPMI channel 3. Do not change. GPMI only support burst size 4.
5–4 CH2	DMA burst size for GPMI channel 2. Do not change. GPMI only support burst size 4.
3–2 CH1	DMA burst size for GPMI channel 1. Do not change. GPMI only support burst size 4.
1–0 CH0	DMA burst size for GPMI channel 0. Do not change. GPMI only support burst size 4.

11.4.7 AHB to APBH DMA Debug Register (APBH_DEBUG)

This register is for debug purpose.

The debug register is for internal use only. Not recommend for customer useage.

Address: APBH_DEBUG is 4100_0000h base + 60h offset = 4100_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-															GPMI_ONE_FIFO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_DEBUG field descriptions

Field	Description
31–1 -	Reserved, always set to zero.
0 GPMI_ONE_FIFO	Set to 0ne and the 8 GPMI channels will share the DMA FIFO, and when set to zero, the 8 GPMI channels will use its own DMA FIFO.

11.4.8 APBH DMA Channel n Current Command Address Register (APBH_CH_CURCMDAR)

The APBH DMA channel n current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel n is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE

```
pCurCmd = (apbh_chn_cmd_t *) APBH_CHn_CURCMDAR_RD(0);              // read the whole
register, since there is only one field
pCurCmd = (apbh_chn_cmd_t *) BF_RDn(APBH_CHn_CURCMDAR, 0, CMD_ADDR);  // or, use multi-
register bitfield read macro
pCurCmd = (apbh_chn_cmd_t *) APBH_CHn_CURCMDAR(0).CMD_ADDR;        // or, assign from
bitfield of indexed register's struct
```

Addresses: 4100_0000h base + 100h offset + (112d × *n*), where *n* = 0d to 15d

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

APBH_CH*n*_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel n.

11.4.9 APBH DMA Channel n Next Command Address Register (APBH_CH_NXTCMDAR)

The APBH DMA Channel n Next Command Address register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

APBH DMA Channel n is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel n semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE

```
APBH_CHn_NXTCMDAR_WR(0, (reg32_t) pCommandTwoStructure);         // write the entire
register, since there is only one field
BF_WRn(APBH_CHn_NXTCMDAR, 0, (reg32_t) pCommandTwoStructure);       // or, use multi-
register bitfield write macro
APBH_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure;  // or, assign to bitfield
of indexed register's struct
```

Addresses: 4100_0000h base + 110h offset + (112d × *n*), where *n* = 0d to 15d

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

APBH_CH*n*_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel n.

11.4.10 APBH DMA Channel n Command Register (APBH_CH_CMD)

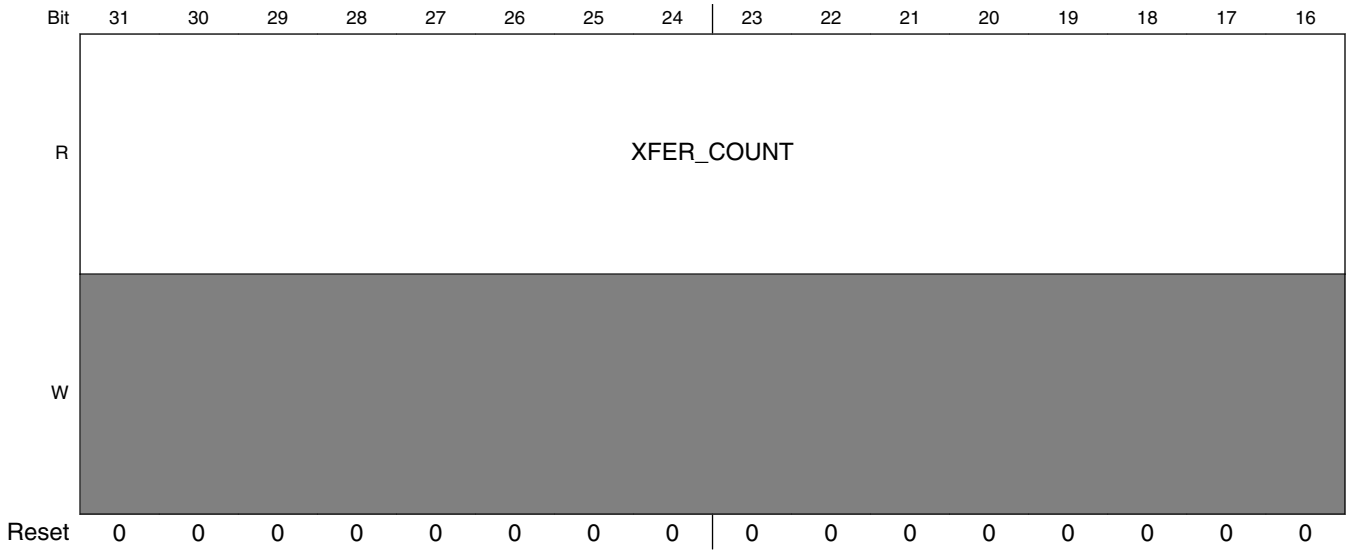
The APBH DMA Channel n command register specifies the DMA transaction to perform for the current command chain item.

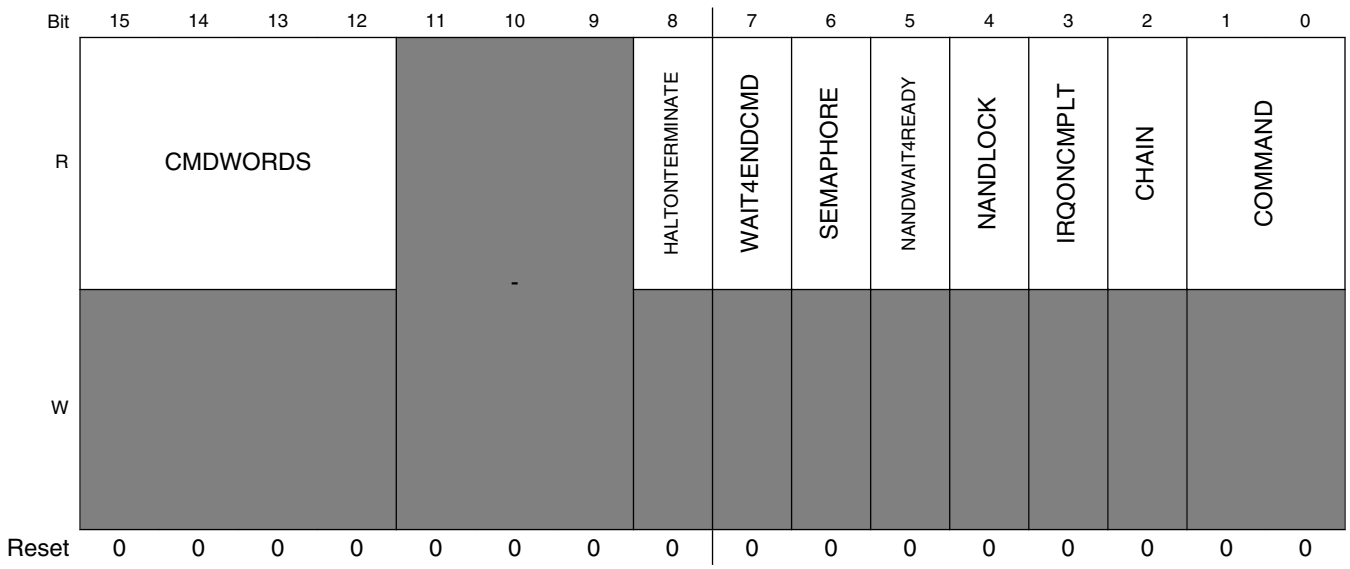
The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE

```
apbh_chn_cmd_t  dma_cmd;
dma_cmd.XFER_COUNT = 512;                                   // transfer 512 bytes
dma_cmd.COMMAND = BV_APBH_CHn_CMD_COMMAND__DMA_WRITE;  // transfer to system memory from
peripheral device
dma_cmd.CHAIN = 1;                                          // chain an additional command
structure on to the list
dma_cmd.IRQONCMPLT = 1;                                     // generate an interrupt on
completion of this command structure
```

Addresses: 4100_0000h base + 120h offset + (112d × *n*), where *n* = 0d to 15d





APBH_CH*n*_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI0 device. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI0, starting with the base PIO address of the GPMI0 control register and incrementing from there. Zero means transfer NO command words
11–9 -	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.

Table continues on the next page...

APBH_CH*n*_CMD field descriptions (continued)

Field	Description
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel will will wait until the NAND device reports "ready" before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel will remain "locked" in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in APBH_CHn_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

11.4.11 APBH DMA Channel n Buffer Address Register (APBH_CH_BAR)

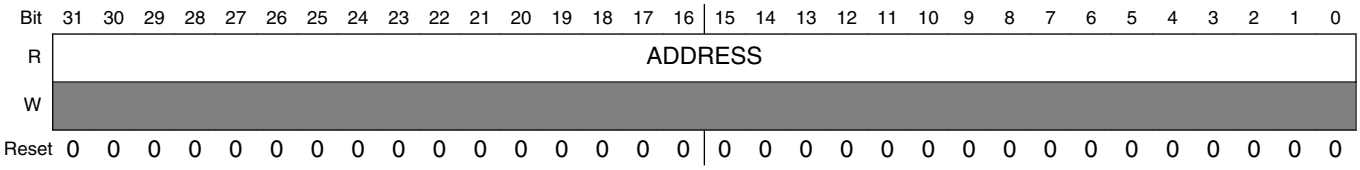
The APBH DMA Channel n buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE

```
apbh_chn_bar_t  dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

Addresses: 4100_0000h base + 130h offset + (112d × *n*), where *n* = 0d to 15d



APBH_CH*n*_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

11.4.12 APBH DMA Channel n Semaphore Register (APBH_CH_SEMA)

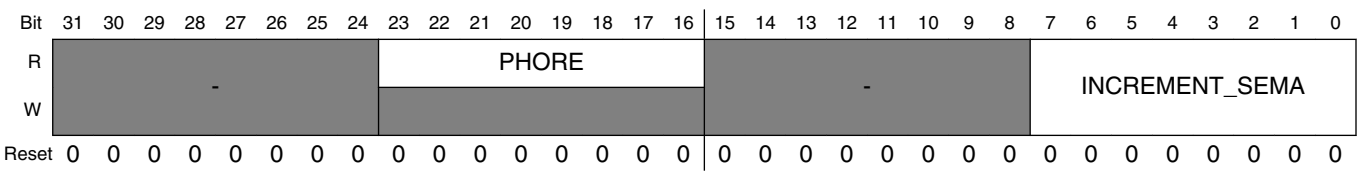
The APBH DMA Channel n semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE

```
BF_WR(APBH_CHn_SEMA, 0, INCREMENT_SEMA, 2);      // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 0, PHORE);  // get instantaneous value
```

Addresses: 4100_0000h base + 140h offset + (112d × *n*), where *n* = 0d to 15d



APBH_CH*n*_SEMA field descriptions

Field	Description
31–24 -	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	Reserved, always set to zero.

Table continues on the next page...

APBH_CH*n*_SEMA field descriptions (continued)

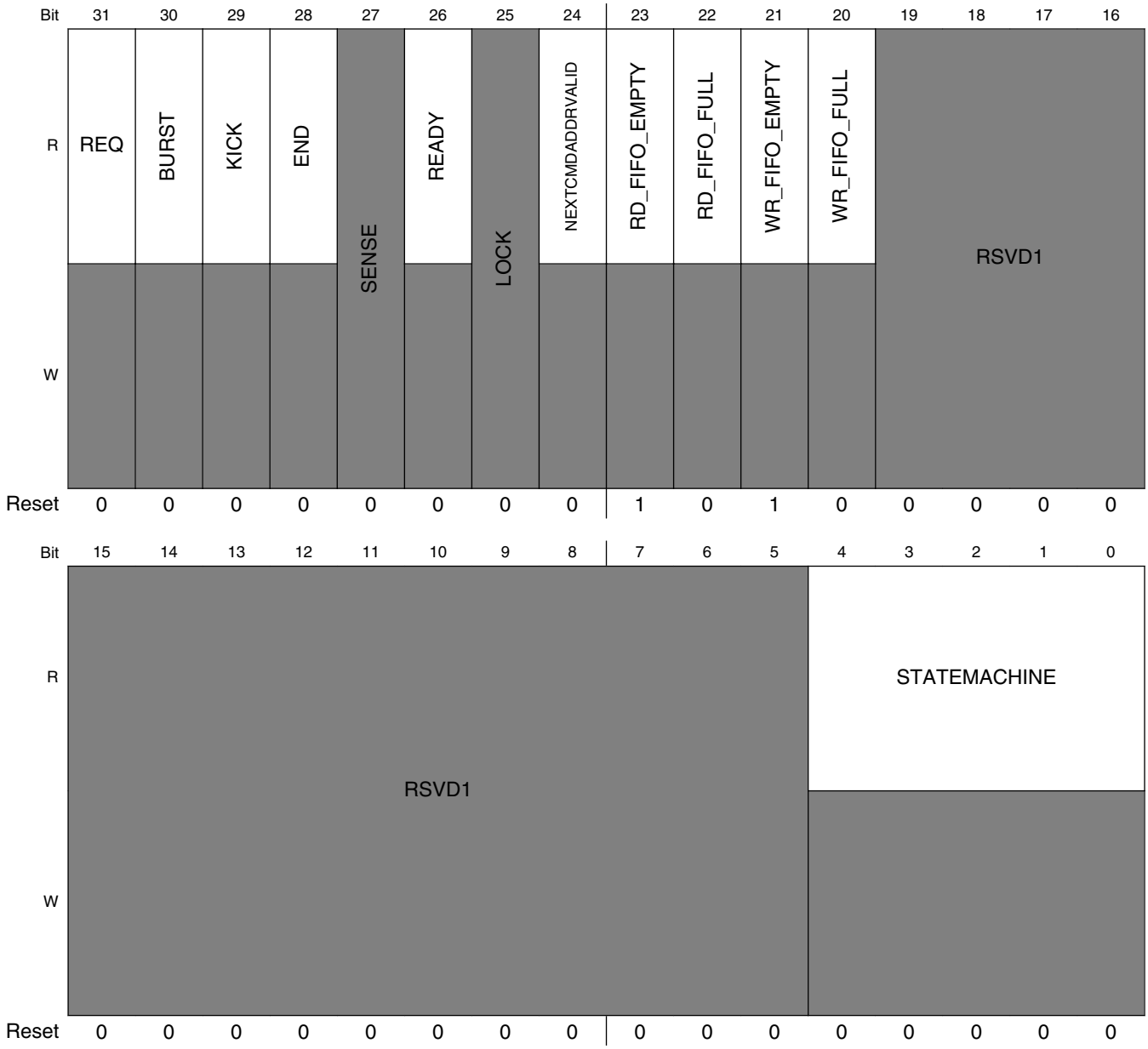
Field	Description
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

11.4.13 AHB to APBH DMA Channel n Debug Information (APBH_CH_DEBUG1)

This register gives debug visibility into the APBH DMA Channel n state machine and controls.

This register allows debug visibility of the APBH DMA Channel n.

Addresses: 4100_0000h base + 150h offset + (112d × *n*), where *n* = 0d to 15d



APBH_CH*n*_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

Table continues on the next page...

APBH_CH*n*_DEBUG1 field descriptions (continued)

Field	Description
27 SENSE	This bit is reserved for this DMA Channel and always reads 0.
26 READY	This bit is reserved for this DMA Channel and always reads 0.
25 LOCK	This bit is reserved for this Channel and always reads 0.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel n state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed.

Table continues on the next page...

APBH_CH*n*_DEBUG1 field descriptions (continued)

Field	Description
	0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

11.4.14 AHB to APBH DMA Channel n Debug Information (APBH_CH_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel n.

This register allows debug visibility of the APBH DMA Channel n.

Addresses: 4100_0000h base + 160h offset + (112d × *n*), where *n* = 0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APBH_CH*n*_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

11.4.15 APBH Bridge Version Register (APBH_VERSION)

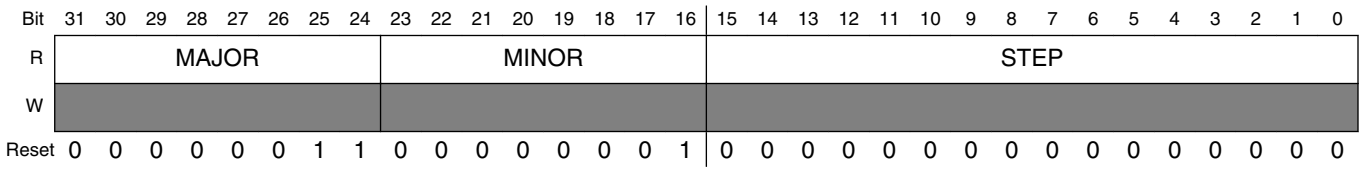
This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
if (APBH_VERSION.B.MAJOR != 3)
   Error();
```

Address: APBH_VERSION is 4100_0000h base + 800h offset = 4100_0800h



APBH_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 12
Digital Audio Multiplexer (AUDMUX)

12.1 Overview

The Digital Audio Multiplexer (AUDMUX) provides a programmable interconnect device for voice, audio, and synchronous data routing between host serial interfaces such as the Synchronous Serial Interface Controller (SSI) and peripheral serial interfaces (audio and voice codecs, also known as coder-decoders).

This section includes a top level diagram that shows the functional organization of the block, including all off-chip signals.

AUDMUX allows the audio system connectivity to be modified through programming, as opposed to altering the PCB schematics of the system. The full description of the block is in Functional Description.

Figure 12-1 shows the block diagram.

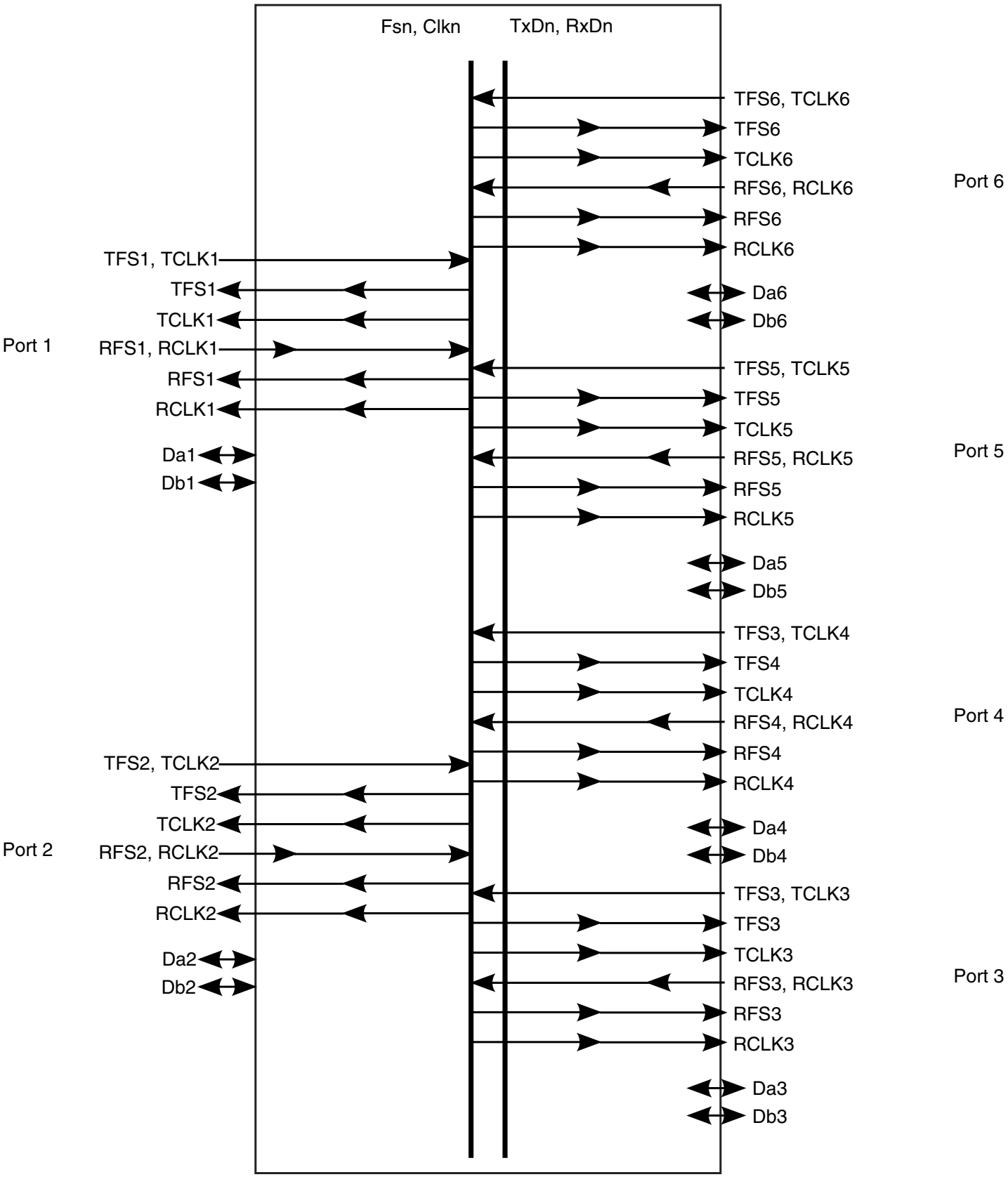


Figure 12-1. AUDMUX Block Diagram

With AUDMUX, resources do not need to be hard-wired and can be effectively shared in different configurations. AUDMUX interconnections allow multiple, simultaneous audio/voice/data flows between the ports in point-to-point or point-to-multipoint configurations.

AUDMUX includes two types of interfaces. Internal ports connect to the processor serial interfaces, and external ports connect to off-chip audio devices and serial interfaces of other processors. The desired connectivity is achieved by configuring the appropriate internal and external ports.

12.1.1 Features

Key features of the block include:

- Two internal ports
- Four external ports
- Full 6-wire SSI interfaces for asynchronous receive and transmit
- Configurable 4-wire (synchronous) or 6-wire (asynchronous) peripheral interfaces
- Independent Tx/Rx Frame sync and clock direction selection for host or peripheral
- Each host interface's capability to connect to any other host or peripheral interface in a point-to-point or point-to-multipoint (network mode)
- Transmit and Receive Data switching to support external network mode

12.1.2 Modes and Operations

The AUDMUX supports the modes described in Operating Modes.

12.2 External Signal Description

The following table lists the off-chip signals of the AUDMUX for the external ports, where P*n* is P3 to P6 and m = (n-2).

Table 12-1. Off-Chip Block Signals

Name	Block Port	I/O	Function	Reset State	Pull-up
AUDm_TXD	P*n*	I/O	Transmit Data from Pn	1	Active
AUDm_RXD	P*n*	I/O	Receive Data at Pn	1	Active
AUDm_TXC	P*n*	I/O	Transmit Clock input/output at Pn	1	-
AUDm_RXC	P*n*	I/O	Receive Clock input/output at Pn	1	-

Table continues on the next page...

Table 12-1. Off-Chip Block Signals (continued)

Name	Block Port	I/O	Function	Reset State	Pull-up
AUDm_TXFS	P*n*	I/O	Transmit Frame sync input/output at Pn	1	-
AUDm_RXFS	P*n*	I/O	Receive Frame sync input/output at Pn	1	-

12.3 Default Register Configuration

There are two configuration registers for each port. Each pair of configuration registers is identical for each port; however, the default values following a reset differ as shown in the Memory Map.

- Default Port Configuration, describes the default configuration of the ports.

12.3.1 Default Port Configuration

After a reset, each port defaults to normal mode (PDCR*n*[MODE] = 0) with synchronous timing mode (PTCR*n*[SYN] = 1) enabled.

The default port-to-port connections are as follows:

- Port 1 to Port 6
  - Port 6 provides the clock and frame sync.
- Port 2 to Port 5
  - Port 5 provides the clock and frame sync.
- Port 3 to Port 4
  - Port 4 provides the clock and frame sync.

12.4 Functional Description

This section provides a complete functional description of the AUDMUX.

12.4.1 Operating Modes

This section describes all functional operation modes of the AUDMUX.

Figure 12-1 shows the AUDMUX block diagram.

All of the ports are essentially identical; there is no functional difference among Ports 1 through 7. The main difference is whether a port is connected to an on-chip serial interface (SSI for example) or connected to the chip's pads to connect to off-chip serial devices (that is, any 4-wire or 6-wire external SSI, voice, I2S, or AC97 codec).

All ports can be configured as four- or six-wire interfaces. When configured as a six-wire interface, the additional RFS and RCLK signals of the interface enable the serial interface to be used in asynchronous mode with separate receive and transmit clocks.

All ports have a Tx/Rx switch to provide flexibility in supporting network mode configurations. The Tx/Rx switch enables the transmit and receive data lines to be swapped so that mastership of the serial bus can be passed among multiple external devices connected to a single port.

In addition to supporting the default external network mode, all ports support an internal network mode:

- With internal network mode, a point-to-multipoint network configuration with an arbitrary number of slaves can be supported if the external slaves are put into the high-impedance state (as defined in the SSI network mode protocol) and have pull-up resistors on their TxD pins. (Alternatively, this can be viewed as requiring a pull-up resistor on the corresponding AUDMUX RxD pin.)

Bit clock direction selection enables each port to be configured as a master or slave in the flow.

Possible scenarios include:

- SSI (internal port) drives a voice codec and a BT (Blue tooth) codec (both on external Port 5) and the Bottom Connector (on external Port 6) simultaneously using network mode. SSI is the master.
- An external processor (external port - Port 4) drives a voice codec and a BT codec (both on external Port 5) and the Bottom Connector (on Port 6) simultaneously using network mode. The external processor is the master.

12.4.1.1 Port Receive Data Modes

Each port has logic to select which data lines are used to create the RxD line for the corresponding host interface.

Figure 12-2 shows the logic used to create the RxD line for Port 1. This logic has the following modes of operation (as determined by MODE):

- Normal
- Internal network mode

The subsequent sections describe the various modes of the port receive data logic. The following terms are used to define the operation of the AUDMUX:

- Network mode- Time-Division Multiplexed protocol for sending unique data to multiple devices on a serial bus.
- Internal network mode-Physical bus configuration where multiple serial buses are effectively connected within the AUDMUX via digital logic to create point-to-multipoint connectivity. An arbitrary number of devices are supported. Devices must be put into the high-impedance state as specified by the network mode protocol. TxDATA lines of devices must be pulled high.
- External network mode-Physical bus configuration where multiple serial buses are electrically connected together on a printed circuit board (that is, external to the AUDMUX). Devices must put their TxDATA lines into the high-impedance state as specified by the network mode protocol.

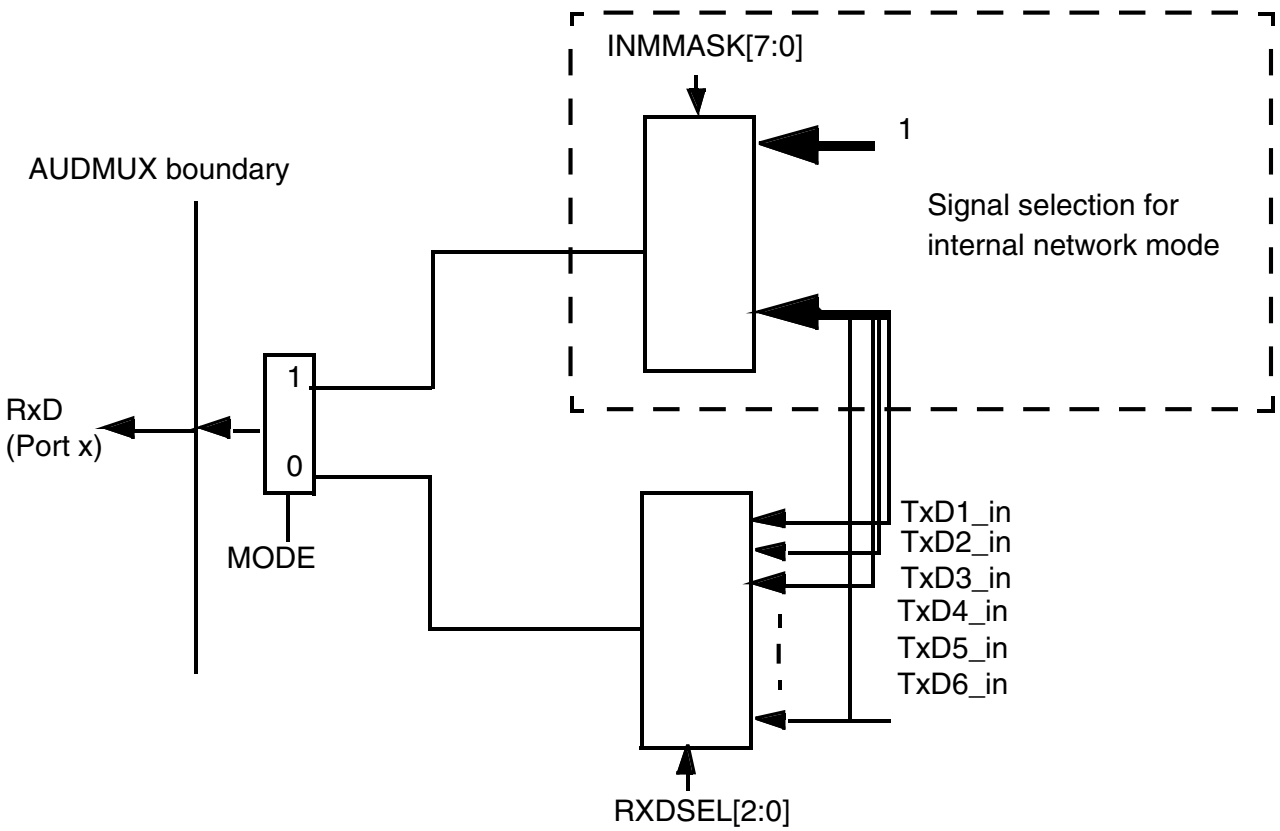


Figure 12-2. Receive Data Logic for Port x

12.4.1.1.1 Normal Mode

In normal mode (MODE = 0) , the port is connected in a point-to-point configuration (as a master or a slave) and the RXDSEL[2:0] setting selects the transmit signal from any port. In normal mode, any data format can be used (that is, SSI normal mode, SSI network mode, AC-97, and others.

12.4.1.1.2 Internal Network Mode

In internal network mode (MODE = 1) , the output of the AND gate is routed (via the output of the port) to the RxD signal of the corresponding host interface.

The INMMASK bit vector selects the transmit signals of the ports that are to be connected in network mode. The transmit signals received at the AUDMUX ports (TxDn_in) are ANDed together to form the output. In internal network mode, only one device can be transmitting in its predesignated timeslot and all other transmit signals must remain high (be in high-impedance state and pulled-up). Therefore, non-active signals in the selection will be high and do not influence the output of the AND gate.

Network mode is a protocol where a master SSI is connected to more than one slave SSI device and communication occurs on a time-slotted frame. Though network mode can allow master-slave and slave-slave communication, internal network mode supports only master-slave communication.

There are two scenarios where internal network mode can be used with external network mode:

1. Slave-only devices are attached to an external port.
2. A master device is attached to an external port and all slave devices connected to the same external port are disabled.

NOTE

When internal network mode is enabled at an external port, RXDSEL[3:0] for RxDn_obe selection is ignored and RxD_obe is always driven high (that is, asserted for all timeslots). All slave devices connected to the same port must be disabled.

Internal Network Mode Example 1

SSI_m and SSI_n are used with Port 4 in internal network mode as shown in Figure 12-3. No pull-up resistors are required because the interfaces combined in internal network mode are on-chip interfaces.

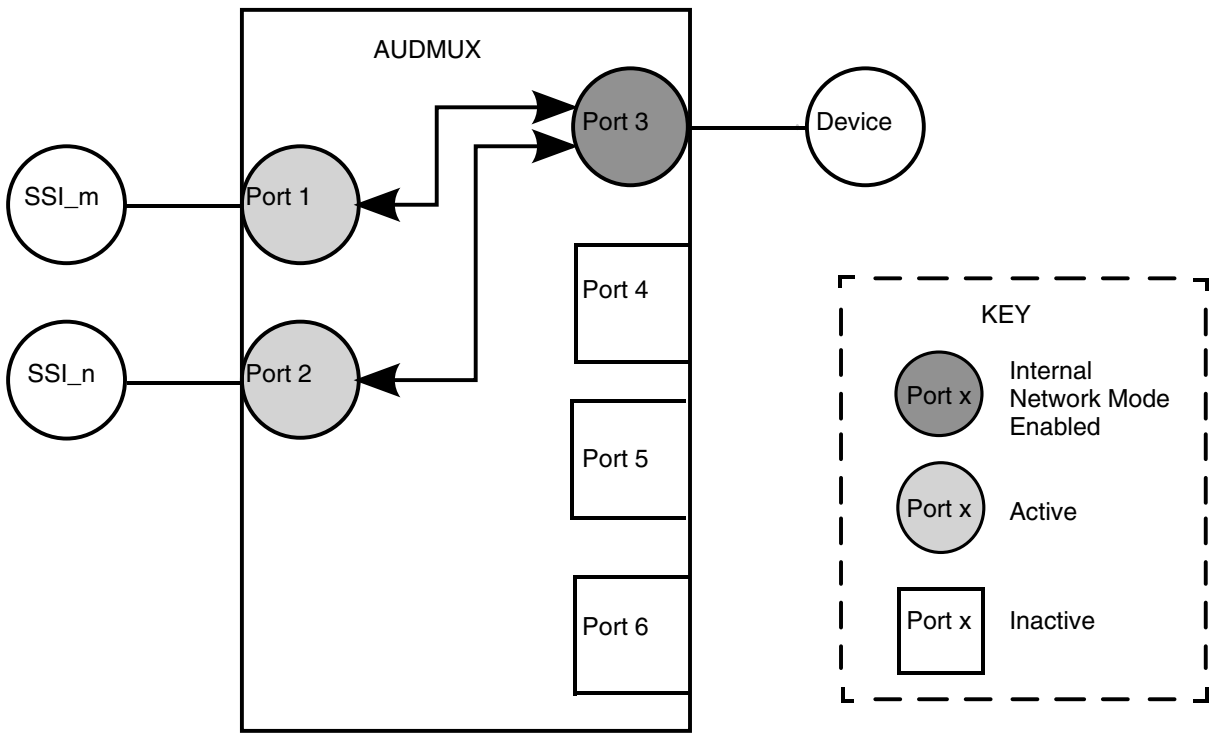


Figure 12-3. Block Diagram For Example 1

See Figure 12-4 for the timing diagram of Example 1. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots.

The data lines for SSI_m and SSI_n (as well as their output enables) are shown. Note that the on-chip interfaces drive a logic '1' when their output enables are logic '0'. The combined TxDATA line, which is the logical AND of the individual TxDATA lines, is used for Port3's TxDATA line.

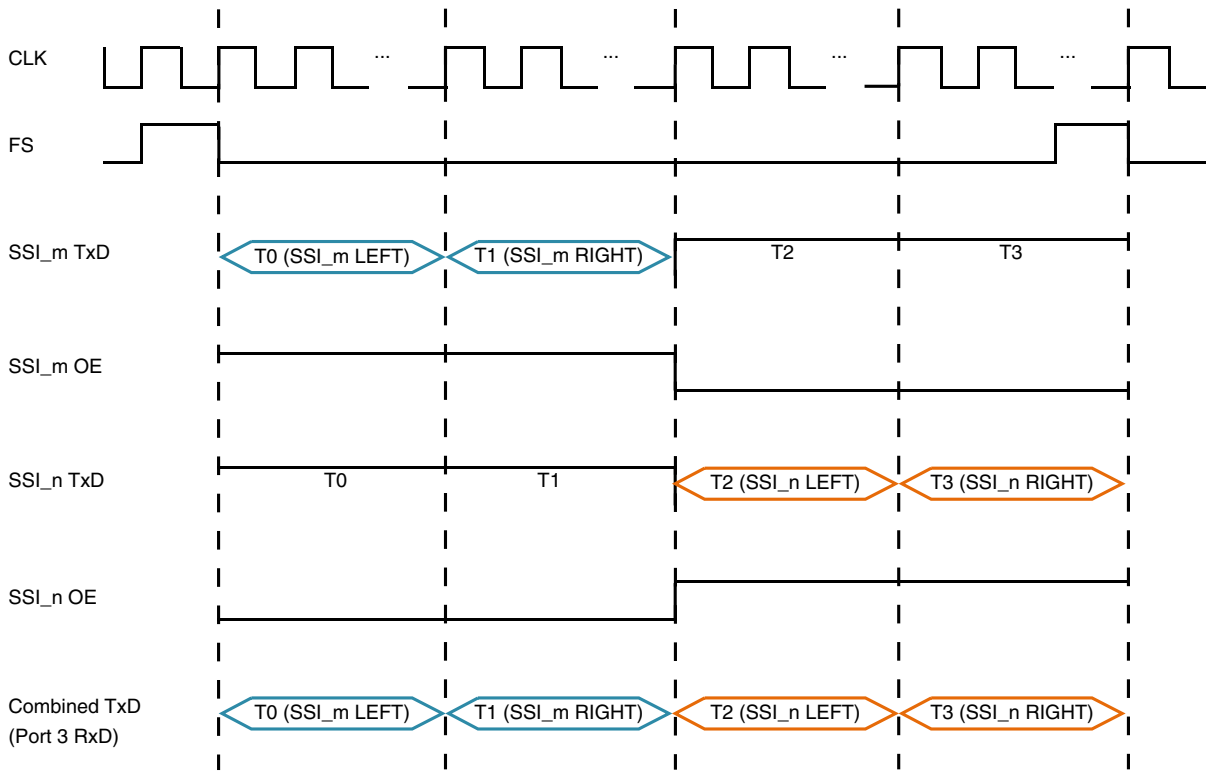


Figure 12-4. Example Using Internal Ports For Transmit Data

Internal Network Mode Example 2

The SSI, Port 3, and Port 4 are used with Port 5 in internal network mode, as shown in the following figure. Note that Port 3 and Port 4 are external ports. Therefore, pull-up resistors are required on the Port 4 RxDATA and Port 5 RxDATA pins. This example shows the timing associated with using adjacent timeslots for the SSI, Port 3 and Port 4.

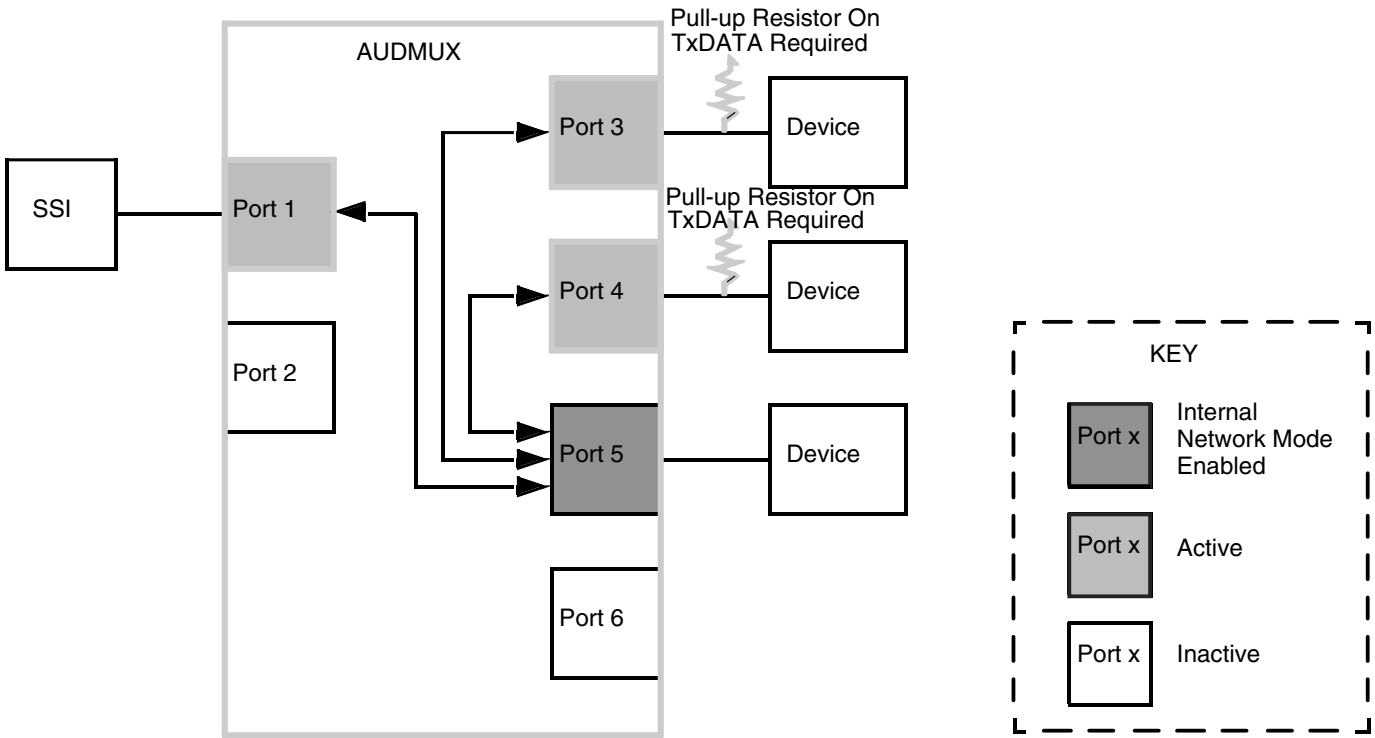


Figure 12-5. Block Diagram For Example 2

The resistance value of the pull-up resistors must be sufficiently high such that a value of '0' can be pulled up to logic '1' within half of a period of the bitclock. The required resistance must be no larger than:

Rmax = 1 / (2 * f_{bc} * C) where:

- f_{bc} is the frequency of the bitclock
- C is the total system capacitance (ICs, board traces, and so on)

The following figure shows the timing diagram for this example. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots.

The data lines for the SSI, Port 3 and Port 4 are shown. Note that the SSI transmits a logic '1' when its corresponding output enable is a logic '0'. The data lines from Port 3 and Port 4 at the pad are pulled high by pull-up resistors when they are in the high-impedance state. The data lines from Port 3 and Port 4 at the AUDMUX are pure digital signals and are constantly driven. The combined TxDATA line, which is the logical AND of the SSI, Port 3 and Port 4's TxDATA lines, is used for Port 5's TxDATA line.

Note the highlighted areas in the following figure. This shows the transition time that occurs while a TxDATA line is being pulled high. In this example, this transition time is a maximum of 1/2 the period of the serial bitclock. This prevents corruption of the first data bit of the next timeslot. It is critical that the pull-up resistance is sufficient for the given bitclock frequency and system capacitance.

Note that hysteresis should be enabled at Port 3's RxDATA pad and Port 4's RxDATA pad to prevent the digital signals created by the pad from toggling rapidly during the pull-up period. The pads typically require a transition within 25ns unless hysteresis is enabled. Instead of using hysteresis, one could select a pull-up resistor sufficiently high to pull-up the signal at the pad within 25 ns; however, that would result in a higher resistance value and higher current drain.

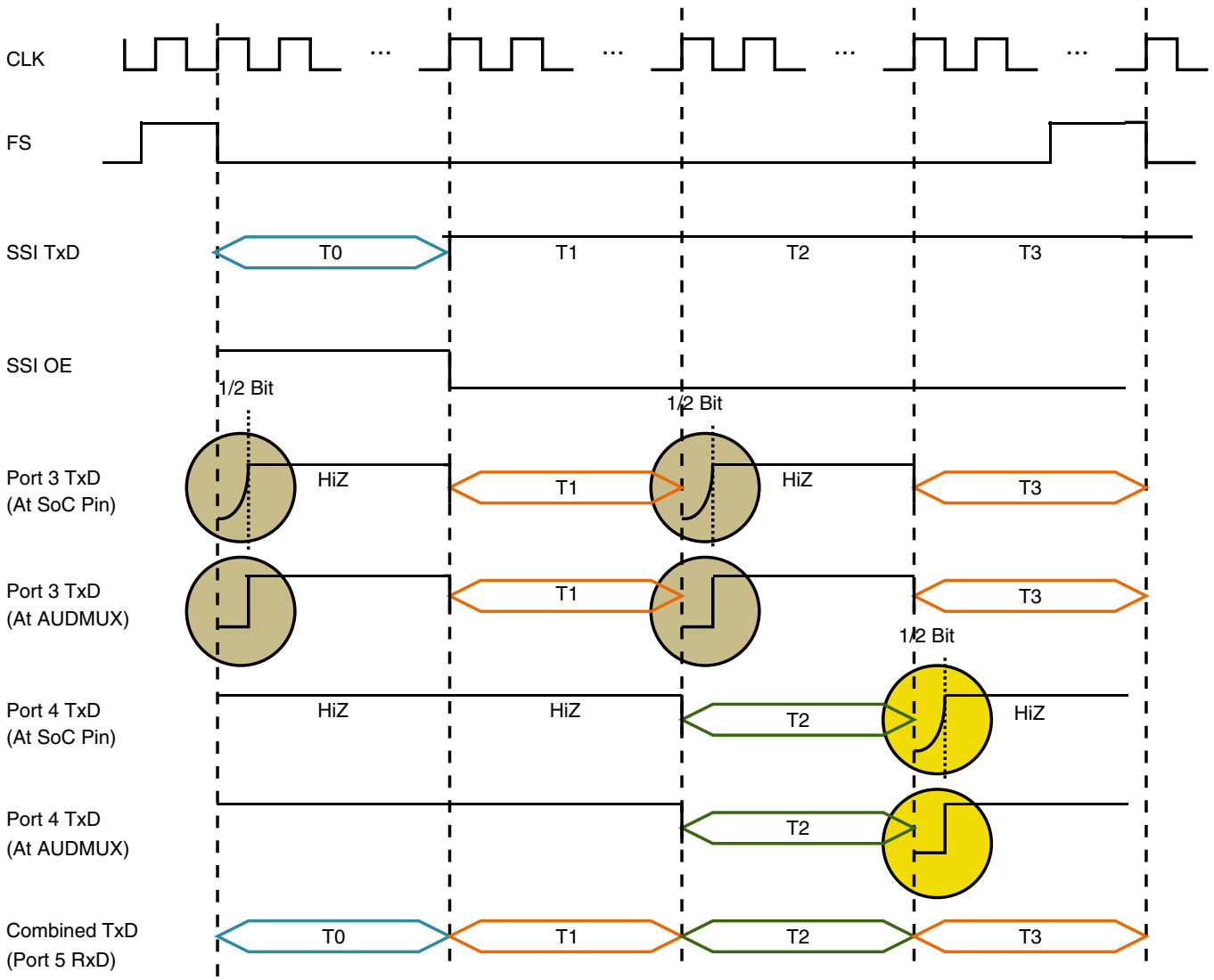


Figure 12-6. Example Using External Ports for Transmit Data in Consecutive Timeslots

Internal Network Mode Example 3

The SSI and Port 3 are used with Port 5 in internal network mode as shown in the following figure. Note that Port 3 is an external port. Therefore, a pull-up resistor is required on the Port 3 TxDATA pin. This example shows the timing associated with inserting empty timeslots after the timeslots have been used by external ports.

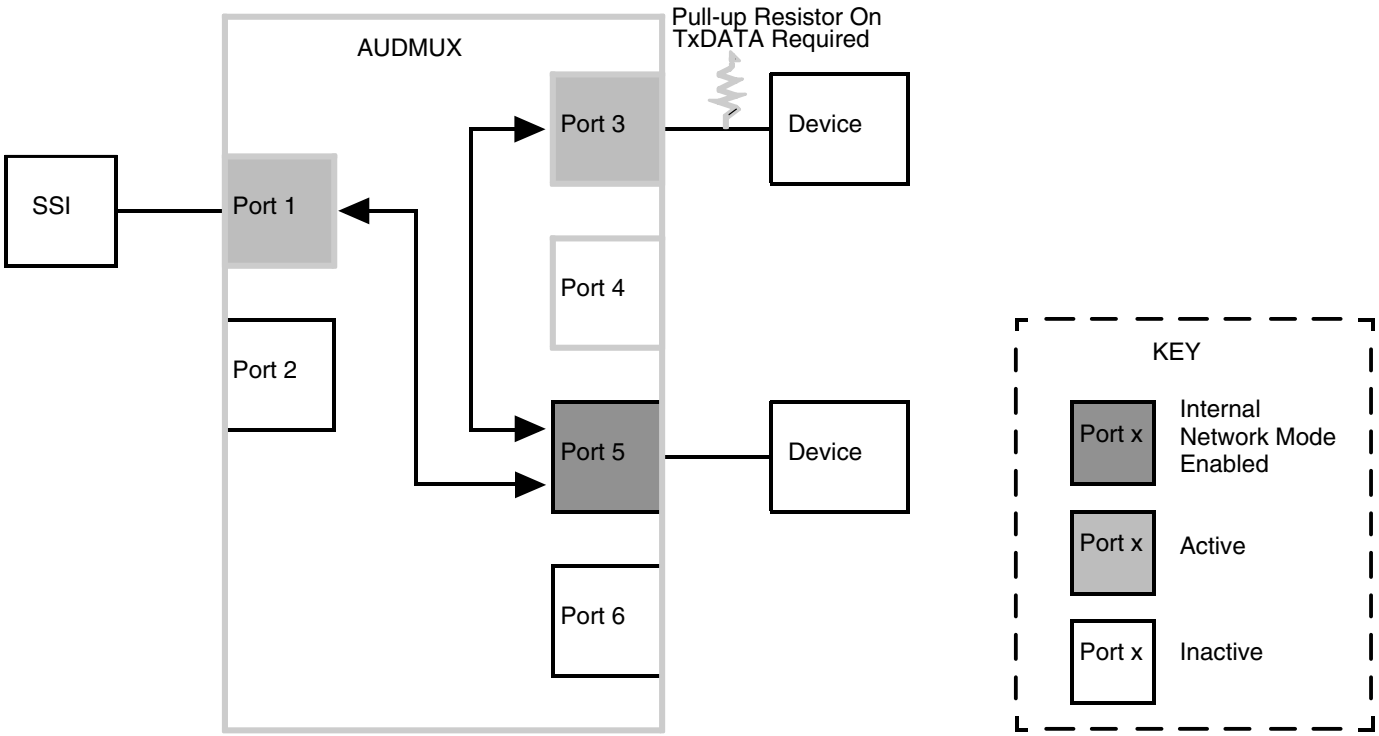


Figure 12-7. Block Diagram For Example 3

The resistance value of the pull-up resistors must be sufficiently high such that a value of '0' can be pulled up to logic '1' by the time that the next occupied timeslot occurs. This allows a much weaker pull-up to be used as compared to Example 2. The required resistance must be no larger than:

Rmax = (4 * n + 1) / (2* f_{bc} * C) where:

- n is the number of bits per timeslot
- f_{bc} is the frequency of the bitclock
- C is the total system capacitance (ICs, board traces, and so on)

The figure below shows the timing diagram for this example. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots.

The data lines for the SSI and Port 3 are shown. Note that the SSI transmits a logic '1' when its corresponding output enable is a logic '0'. The data line from Port 3 at the pad is pulled high by a pull-up resistor when they are in the high-impedance state. The data line from Port 3 at the AUDMUX is a pure digital signal and is constantly driven. The combined TxDATA line, which is the logical AND of the SSI and Port 3's TxDATA lines, is used for Port 5's RxDATA line.

Note the highlighted area in the following figure. This shows the transition time that occurs while Port 3's TxDATA line is being pulled high. In this example, this transition time is a maximum of two timeslots plus 1/2 the period of the serial bitclock. This prevents corruption of the first data bit of the next timeslot. It is critical that the pull-up resistance is sufficient for the given bitclock frequency and system capacitance.

Note that hysteresis must be enabled at Port 3's RxDATA pad to prevent the digital signal created by the pad from toggling rapidly during the extended pull-up period. The pads typically require a transition within 25 ns unless hysteresis is enabled.

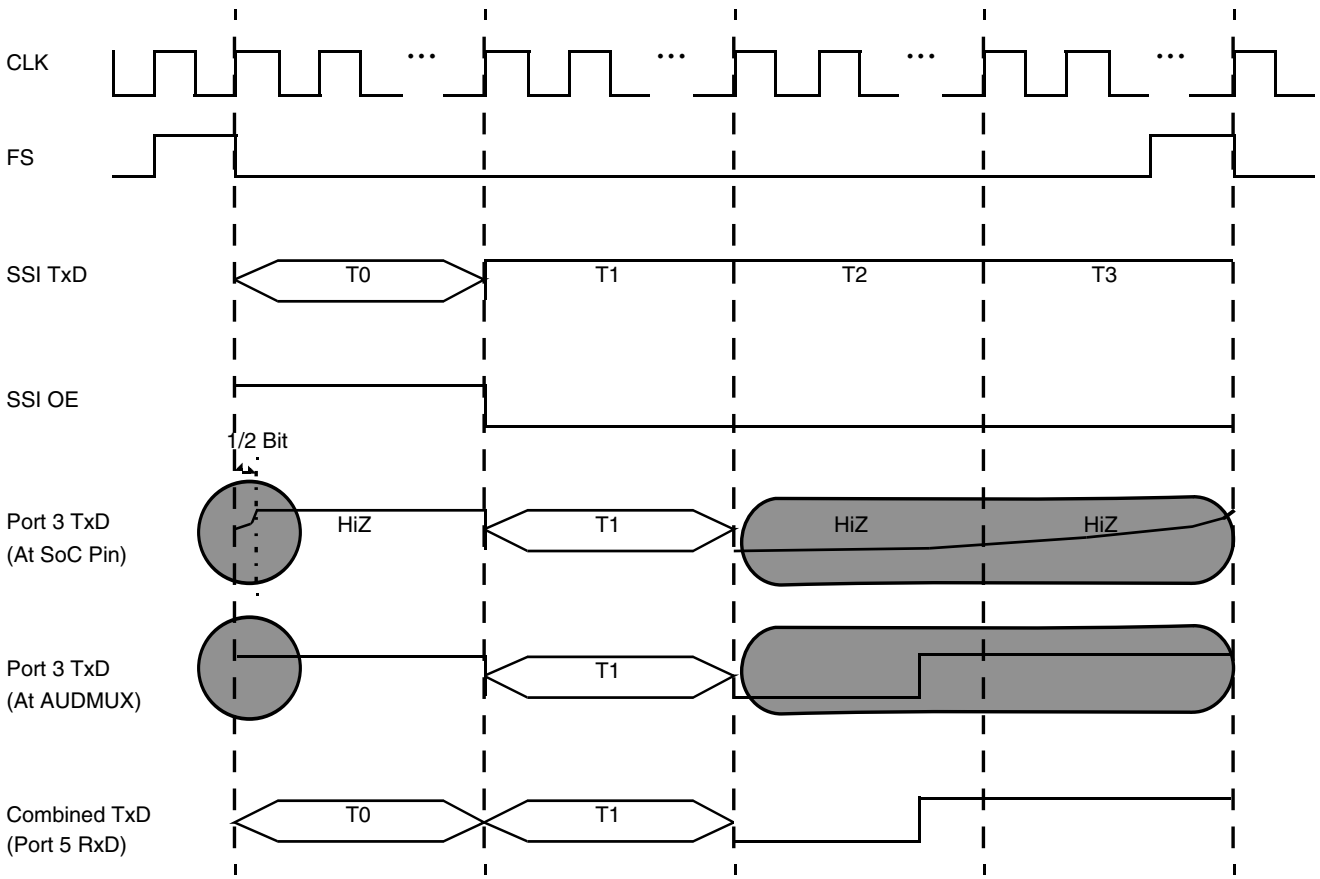


Figure 12-8. Example Using External Ports For Transmit Data In Nonconsecutive Timeslots

12.4.1.1.3 Transmit Data Output Enable Assertion

The TxDATA line from the internal network mode master (connected at any internal port) is put into the high-impedance state at the pad depending upon the assertion or deassertion of TxD_obe, its corresponding output enable generated by the network mode master.

In the case of an external network mode master (connected at an external port), the corresponding TxD_obe is always asserted after the port data register configuration.

12.4.1.2 Tx/Rx Switch and External Network Mode

External network mode is the traditional network mode connection. It is called external network mode to differentiate from the internal network mode. In external network mode, devices are connected to a single external port in a star or multi-drop configuration.

In network mode, there can be only one master (driving the frame sync and clock source) with the other devices configured in normal slave mode or network slave mode. Unlike internal network mode, both master-slave and slave-slave communication can take place in external network mode. Codec devices transmit on a single timeslot while processor serial interfaces (that is, SSI) can process more than one timeslot of data while in network master or slave mode.

The following figure shows the Tx/Rx data switch. RxD_obe is the output buffer enable signal and RxD_out is the data transmit signal from the serial interface. The TxD_in signal is the receive data signal going towards the RXDSEL muxes of all ports.

D_TxRx is the data pin which serves as the chip-level transmit data pin when the TxRx switch is not enabled. D_RxTx is the data pin which serves as the chip-level receive data pin when the TxRx switch is not enabled. The roles of these pins are reversed when the TxRx switch is enabled.

When TXRXEN is disabled (TXRXEN=0), RxD_out is routed to D_TxRx and D_RxTx is routed to TxD_in. The output buffer enable, selected by RXDSEL[2:0], is routed to Db_obe.

When the Tx/Rx switch is enabled (TXRXEN=1), RxD_out is routed to D_RxTx and D_TxRx is routed to TxD_in. The output buffer enable, selected by RXDSEL[2:0], is routed to Da_obe.

If the RXDSELn[2:0] field for any Port *n* is configured to select data from an internal port, the output buffer enable is selected by RXDSELn[2:0] and is routed to Da*n*_obe/ Db*n*_obe. In the case when the RXDSELn[2:0] field for Port *n* is configured to select data from an external port, the output buffer enable is always high and routed to Dan_obe/Dbn_obe, depending on the TXRXENn switch configuration.

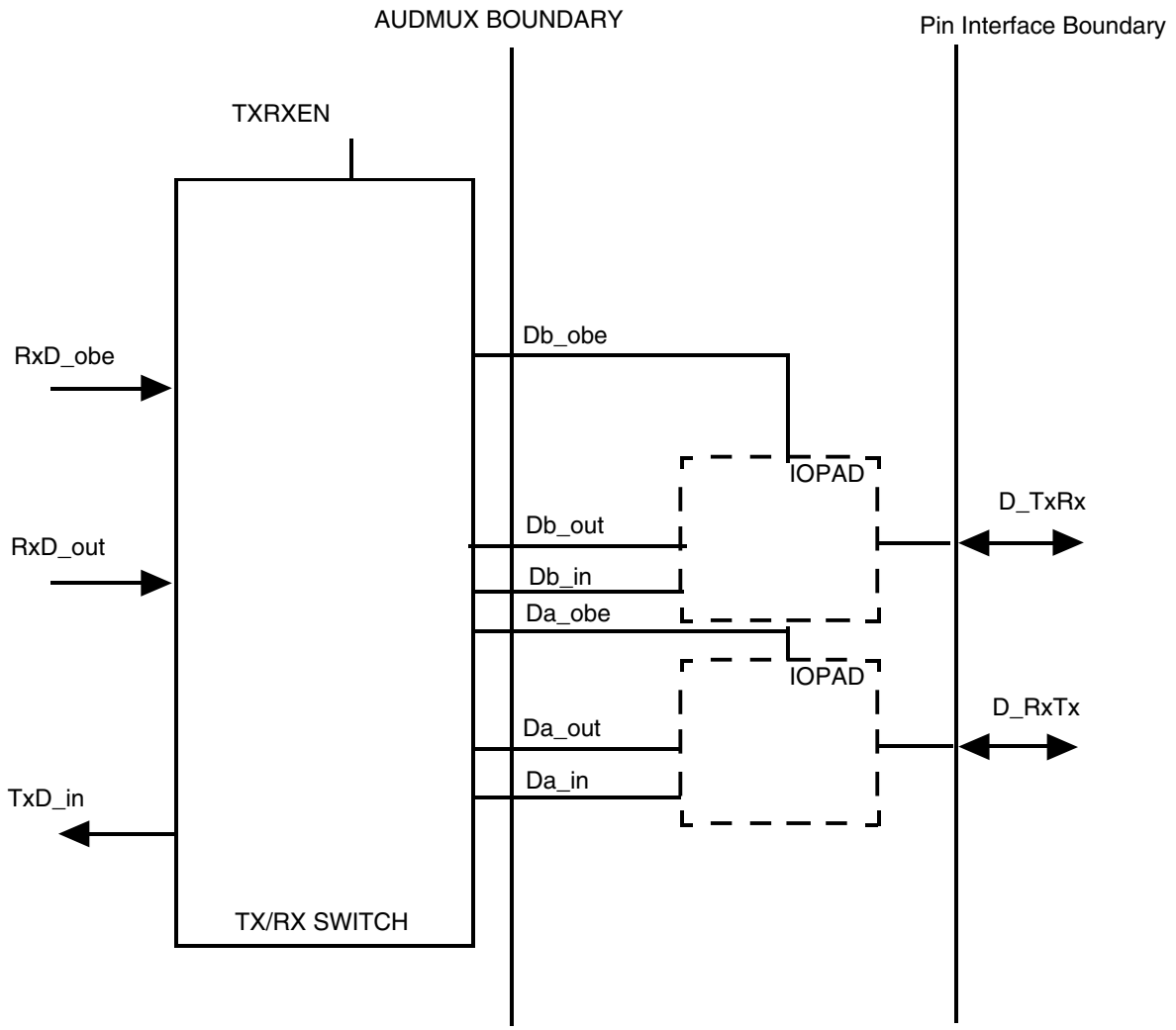


Figure 12-9. Tx/Rx Switch

12.4.1.3 Timing Modes

The AUDMUX ports are constructed as 6-wire interfaces. However, they can be used either in synchronous or asynchronous modes as determined by the SYN bit.

12.4.1.3.1 Synchronous Mode (4-Wire Interface)

In Synchronous mode, the port has a 4-wire interface (that is, RxD, TxD, TxCLK, TxFS). The receive data timing is determined by TxCLK and TxFS.

As shown in the following figure, Port x signals can be routed to Port y, producing 6-wire to 4-wire port connectivity.

TFS_in, RFS_in, TCLK_in, and RCLK_in are the input frame sync and bit clocks from the serial interface (Port x) with their corresponding output buffer enable signals (_obe). TFS_out, RFS_out, TCLK_out, and RCLK_out are the frame sync and bit clocks that are transmitted to the serial interface from the other ports.

The TFS_out and TCLK_out are selected at Port x by the TFSEL and TCSEL mux settings, respectively. RFS_out and RCLK_out are selected at Port x by the RFSEL and RCSEL mux settings, respectively. Similarly, in the external direction, Port y is configured as a 4-wire port; TFSEL selects the FS_obe and FS_out signals. In this mode, the configuration of RFSEL and RCSEL is not used, since the RFS_out and RCLK_out pins at Port y are not available.

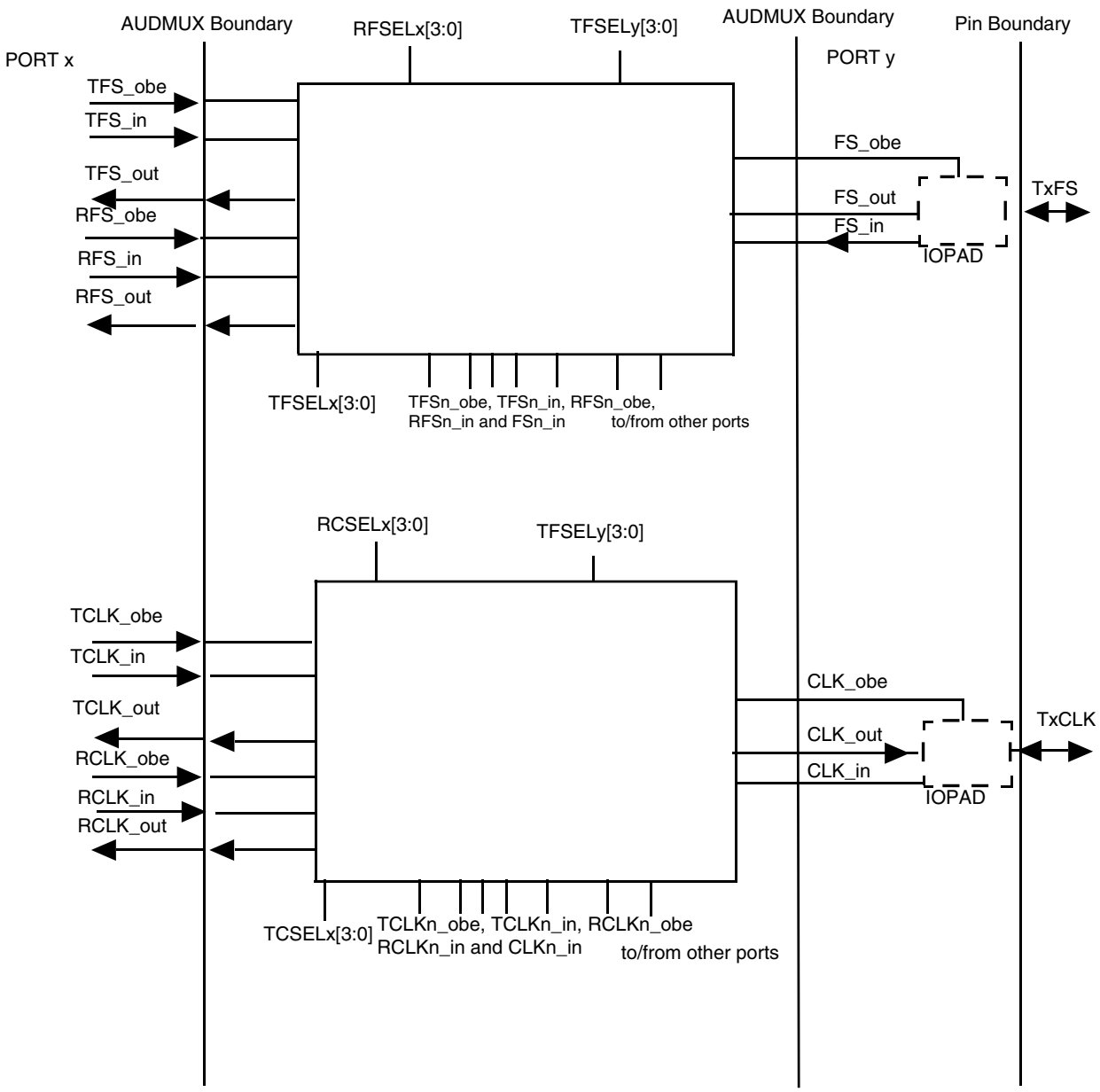


Figure 12-10. Frame Sync and Clock Routing When External Port Is 4-Wire

12.4.1.3.2 Asynchronous Mode (6-Wire Interface)

In Asynchronous mode, the port has a 6-wire interface (meaning RxD, TxD, TxCLK, TxFS, RxCLK, RxFS). This mode has additional receive clock (RxCLK) and frame sync (RxFS) signals as compared to the synchronous or 4-wire interface.

As shown in the figures below, Port x signals can be routed to Port y, producing 6-wire to 6-wire port connectivity.

TFS_in, RFS_in, TCLK_in, and RCLK_in are input frame sync and bit clocks from the serial interface (Port x) with their corresponding output buffer enable signals (_obe). TFS_out, RFS_out, TCLK_out, and RCLK_out are the frame sync and bit clocks that are transmitted to the serial interface from the other ports.

TFS_out and TCLK_out are selected by the TFSEL and TCSEL mux settings, respectively. RFS_out and RCLK_out are selected by the RFSEL and RCSEL mux settings, respectively. Similarly, in the external direction, the TFSEL selects the TxFS_obe and TxFS_out signals and TCSEL selects the TxCLK_obe and TxClk_out signals. The RFSEL selects the RxFS_obe and RxFS_out signals and RCSEL selects the RxCLK_obe and RxCLK_out signals.

NOTE

Because FS_in and CLK_in from external interfaces are also routed to the TFSEL and TCSEL muxes of the external ports, these signals do not have corresponding buffer enable signals. Consequently, their corresponding inputs to the TFSEL and TCSEL mux of the external ports have to be tied high.

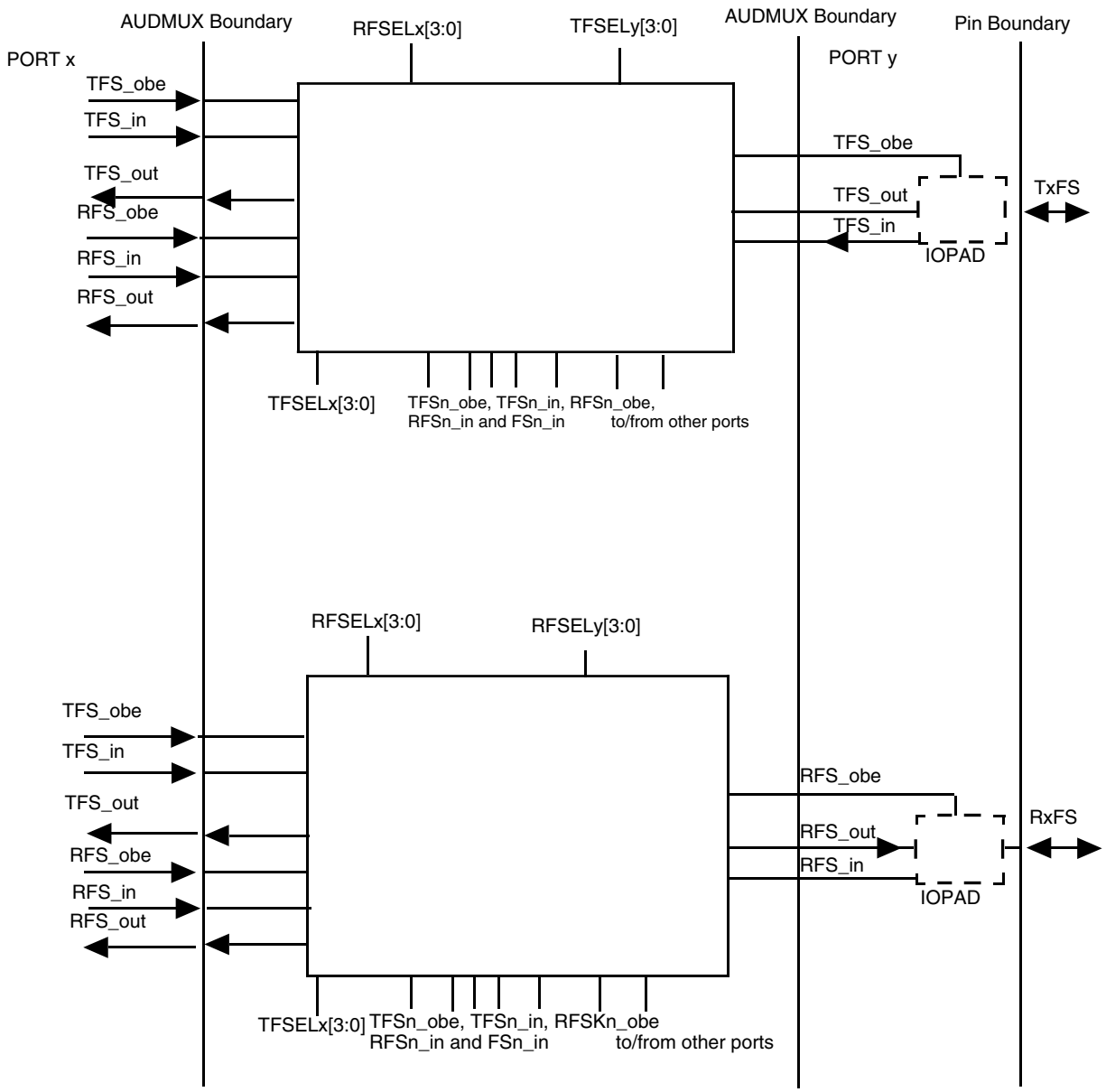


Figure 12-11. Frame Sync Routing When External Port Is 6-Wire

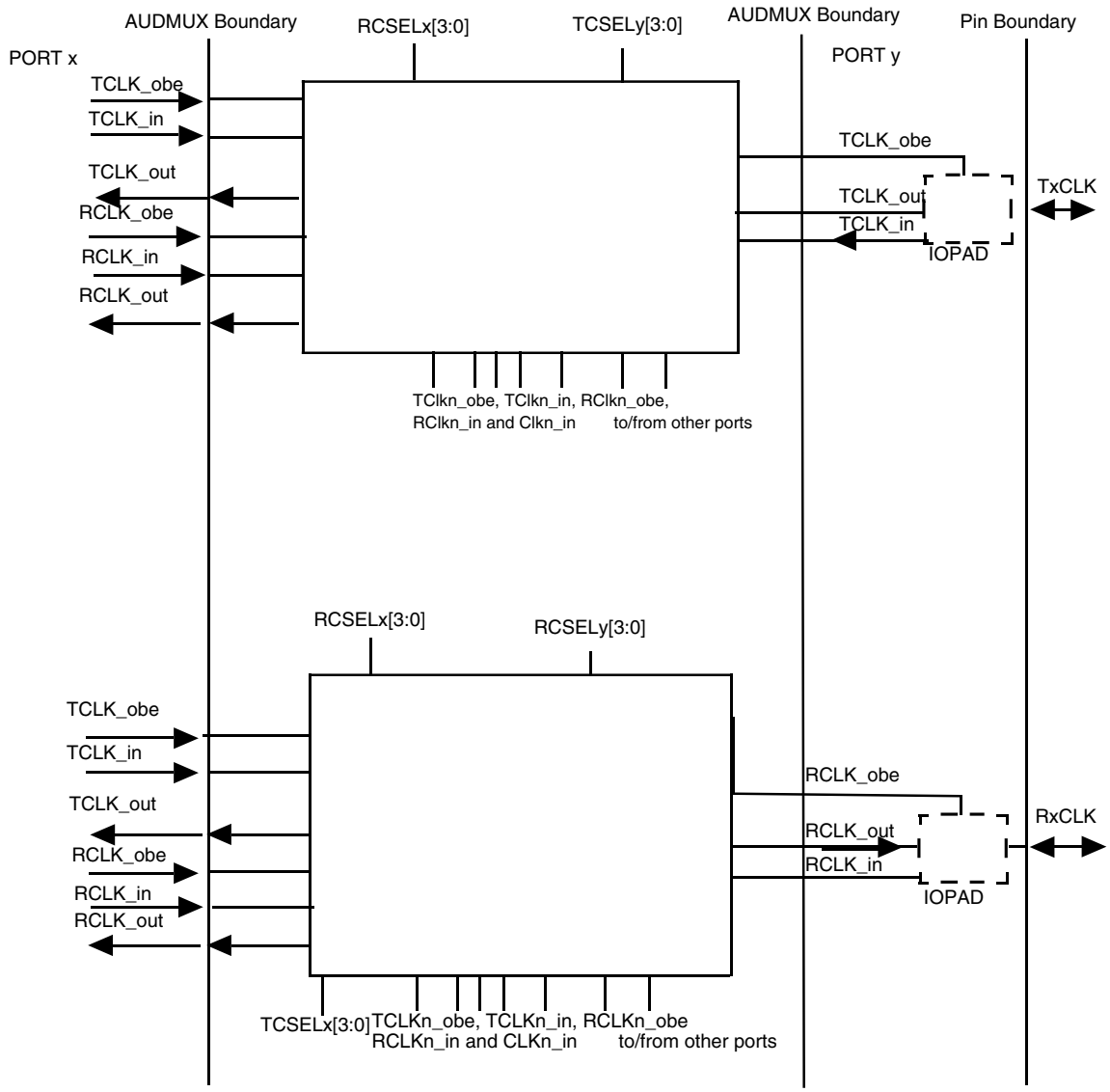


Figure 12-12. Clock Routing When External Port Is 6 Wire

12.4.2 Connectivity Between Ports

Four basic types of connections are provided by the AUDMUX:

- Internal port to external port
- External port to external port
- Internal port to internal port
- Loopback

The corresponding data connections are described in the following sections.

12.4.2.1 Internal Port to External Port Connectivity

The internal port is connected to a processor's serial interface. TxD_obe is the buffer enable signal from the serial interface, TxD_in is the input transmit data from the serial interface to the AUDMUX, and RxD_out is the receive data output from the AUDMUX to the serial interface.

RXDSEL[2:0] of the external port selects the buffer enable signal (TxD_obe) and transmit data output (TxD_out) signal from the TxD_obe and RxD_in signals. RXDSEL[2:0] is a common signal to both selection muxes.

NOTE

Because buffer TxD_in signals from external interfaces do not have corresponding buffer enable signals, their buffer enable signals into the selection mux are tied high. This will ensure that selection of TxD_in, as RxD_out will also drive the RxD_obe output high.

Transmit Data from the serial interface goes into the RXDSEL data mux and comes out as RxD_out. RxD_out is routed to Da_TxRx when TXRXEN is disabled and to D_RxTx when TXRXEN is enabled. Similarly, D_RxTx is routed to TxD_in when TXRXEN is disabled and D_TxRx is routed to TxD_in when TXRXEN is enabled. The routing of frame syncs is shown in Figure 12-11and the routing of interface clocks is shown in Figure 12-9.

If internal network mode is disabled, then RXDSEL selects the TxD_in, which is sent from the AUDMUX to the serial interface connected at Port x. When the internal network mode is selected, RxD_out is constructed by ANDing selected TxD_in signals from the ports (as determined by INMMASK).

If there is more than one device attached to the external port at D_TxRx and D_RxTx and one of the devices is a network master, then two conditions must be noted:

1. When the external master is enabled in network mode, then the serial interface at Port x must be configured as a slave (normal or network mode). No Tx/Rx switching is required.
2. When the external master is disabled and the serial interface at Port x and other slave devices must communicate, then the serial interface at Port x must be configured as a network mode master and the Tx/Rx switch at Port y must be enabled (TXRXEN=1). This will ensure that the transmit and receive paths are connected appropriately.

To communicate with more than one port, internal network mode can be enabled at Port x. In internal network mode, it is possible to communicate with any device attached to the other ports. Internal network mode shall be enabled at the port that is the SSI network mode master.

12.4.2.2 External Port to External Port Connectivity

External ports can communicate with external ports directly.

External ports can communicate together in three ways:

1. Each port's receive logic is configured in normal mode (MODE = 0) . Each port's RXDSEL[2:0] field is configured to select the other port's transmit data. Bit fields associated with clock/frame sync selection and direction are configured for each port. Either port can be the master.
2. One port is configured in internal network mode (MODE = 1) . All desired data lines are combined by the AND gate as determined by INMMASK[7:0]. Since an external port is being used as the internal network mode master, all other devices on the same AUDMUX port as the internal network mode master must be disabled. This configuration can be used with a combination of internal and external ports. All external ports must have a pull-up resistor on its RxDATA pin. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any port can be the master.

12.4.2.3 Internal Port to Internal Port Connectivity

Internal ports can communicate with other internal ports directly.

Internal ports can communicate together in two ways:

1. Each port's receive logic is configured in normal mode (MODE = 0) . Each port's RXDSEL[2:0] field is configured to select the other port's transmit data. Bit fields associated with clock/frame sync selection and direction are configured for each port. Either port can be the master.
2. One port is configured in internal network mode (MODE = 1) . All desired data lines are combined by the AND gate as determined by INMMASK[7:0]. This configuration can be used with a combination of internal and external ports. All external ports must have a pull-up resistor on its RxDATA pin. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any port can be the master.

12.4.2.4 Loopback Connectivity

AUDMUX ports can communicate with themselves in order to provide loopback functionality. Port x can route its TxDATA signal to its own RxD_out signal by setting RXDSELx[2:0] to its own port number. This is supported by all ports in the AUDMUX.

In addition, ports can provide loopback support in internal network mode. With internal network mode, the internal network mode master can loop its TxDATA signal (combined with those of other ports, if desired) back into its RxD_out signal. Port x's INMMASK should be set such that bit (x - 1) is clear in order to enable the loopback.

12.4.3 AUDMUX Clocking

This section provides information about AUDMUX clocking including clock inputs and the clock diagram.

12.4.3.1 AUDMUX Clock Inputs

The IP Bus read/write clock-peripheral clock-is an input to the AUDMUX. It is used for all AUDMUX register accesses. It is driven only when there is an AUDMUX access on the IP Bus.

12.4.3.2 AUDMUX Clock Diagram

The figure below shows the clocking used in the AUDMUX.

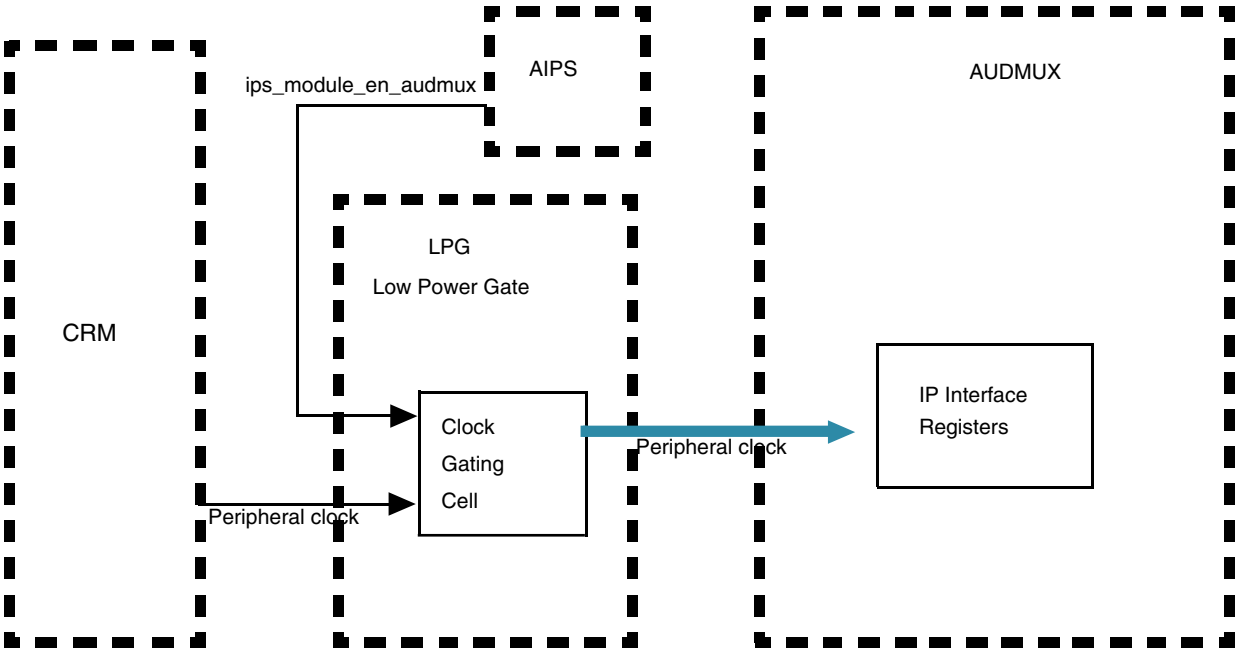


Figure 12-13. AUDMUX Clocking Scheme

12.4.3.3 Clocking Restrictions

- Since the AUDMUX requires only peripheral clock, the AUDMUX places no restrictions on the bus frequency.
- All registers in the AUDMUX are control registers so their values will not change frequently. Their values will be programmed when changing between use cases (not during use cases).

12.5 Programmable Registers

This section includes the block memory map and detailed descriptions of all registers. For the base address of a specific sub-block instantiation, see the system memory map in this manual.

The AUDMUX memory map is shown in the following table.

AUDMUX memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FD_0000	Port Timing Control Register 1 (AUDMUX_PTCR1)	32	R/W	AD40_0800h	12.5.1/ 658
63FD_0004	Port Data Control Register 1 (AUDMUX_PDCR1)	32	R/W	0000_A000h	12.5.2/ 660
63FD_0008	Port Timing Control Register 2 (AUDMUX_PTCR2)	32	R/W	A500_0800h	12.5.3/ 661
63FD_000C	Port Data Control Register 2 (AUDMUX_PDCR2)	32	R/W	0000_8000h	12.5.4/ 663
63FD_0010	Port Timing Control Register 3 (AUDMUX_PTCR3)	32	R/W	9CC0_0800h	12.5.5/ 664
63FD_0014	Port Data Control Register 3 (AUDMUX_PDCR3)	32	R/W	0000_6000h	12.5.6/ 666
63FD_0018	Port Timing Control Register n (AUDMUX_PTCR4)	32	R/W	0000_0800h	12.5.7/ 667
63FD_001C	Port Data Control Register 4 (AUDMUX_PDCR4)	32	R/W	0000_4000h	12.5.8/ 670
63FD_0020	Port Timing Control Register n (AUDMUX_PTCR5)	32	R/W	0000_0800h	12.5.7/ 667
63FD_0024	Port Data Control Register 5 (AUDMUX_PDCR5)	32	R/W	0000_2000h	12.5.9/ 671
63FD_0028	Port Timing Control Register n (AUDMUX_PTCR6)	32	R/W	0000_0800h	12.5.7/ 667
63FD_002C	Port Data Control Register 6 (AUDMUX_PDCR6)	32	R/W	0000_0000h	12.5.10/ 672

12.5.1 Port Timing Control Register 1 (AUDMUX_PTCR1)

PTCR1 is the Port Timing Control Register for Port 1.

Address: AUDMUX_PTCR1 is 63FD_0000h base + 0h offset = 63FD_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFS DIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFS DIR	RFSEL[3:0]				RCLKDIR
Reset	1	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RCSEL[3:0]				SYN	0										
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

AUDMUX_PTCR1 field descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 x101 Port 6 x11x Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 x101 Port 6 x11x Reserved
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 x101 Port 6 x11x Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. **NOTE:** RCLKDIR and SYN should not be changed at the same time. 0 RxClk is an input. 1 RxClk is an output.

Table continues on the next page...

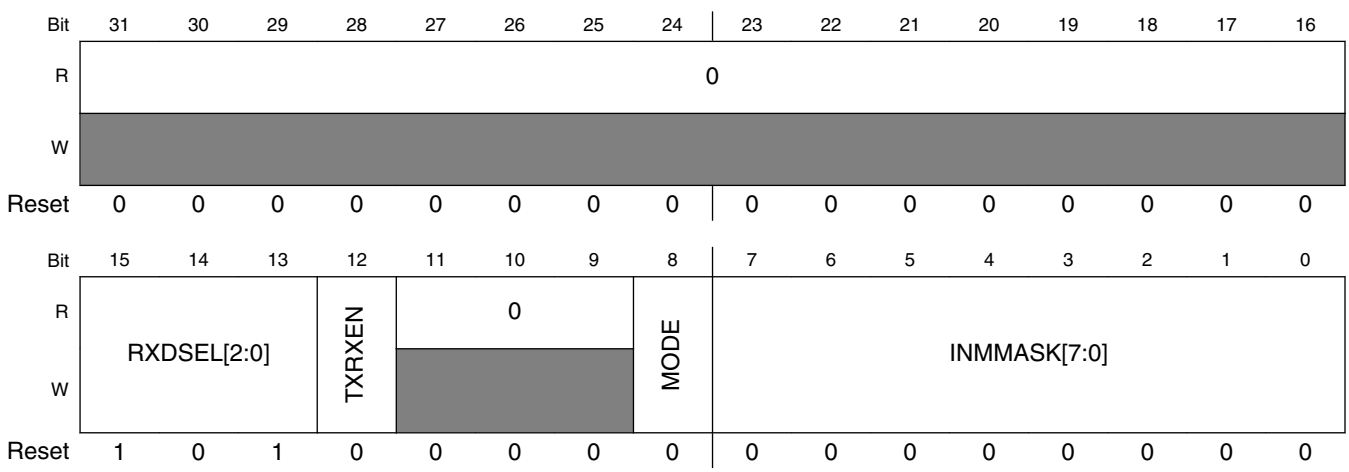
AUDMUX_PTCR1 field descriptions (continued)

Field	Description
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 x101 Port 6 x11x Reserved
11 SYN	Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). **NOTE:** RCLKDIR and SYN should not be changed at the same time. 0 Asynchronous mode 1 Synchronous mode (default)
10–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

12.5.2 Port Data Control Register 1 (AUDMUX_PDCR1)

PDCR1 is the Port Data Control Register for Port 1.

Address: AUDMUX_PDCR1 is 63FD_0000h base + 4h offset = 63FD_0004h



AUDMUX_PDCR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

AUDMUX_PDCR1 field descriptions (continued)

Field	Description
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE = 1 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 101 Port 6 11x Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 MODE	Mode Select. This field selects the mode in which the port is to operate. The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDed together. RXDSEL is ignored. INMMASK determines which RxD signals are ANDed together. 0 Normal mode 1 Internal Network mode
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which the RxD signals are to be ANDed together for internal network mode. Bit 6 represents RxD from Port 6 and bit0 represents RxD from Port 1. 0 Includes RxDn for ANDing 1 Excludes RxDn from ANDing

12.5.3 Port Timing Control Register 2 (AUDMUX_PTCR2)

PTCR2 is the Port Timing Control Register for Port 2.

Address: AUDMUX_PTCR2 is 63FD_0000h base + 8h offset = 63FD_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFS DIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFS DIR	RFSEL[3:0]				RCLKDIR
Reset	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RCSEL[3:0]				SYN	0										
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

AUDMUX_PTCR2 field descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 x101 Port 6 x11x Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 x101 Port 6 x11x Reserved
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 x101 Port 6 x11x Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. **NOTE:** RCLKDIR and SYN should not be changed at the same time. 0 RxClk is an input. 1 RxClk is an output.

Table continues on the next page...

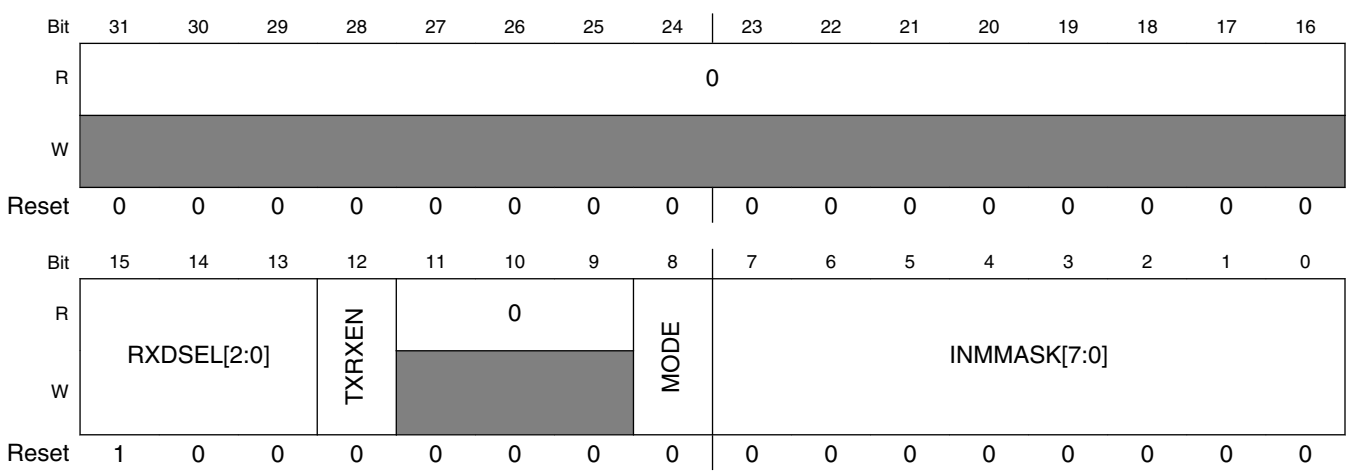
AUDMUX_PTCR2 field descriptions (continued)

Field	Description
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 x101 Port 6 x11x Reserved
11 SYN	Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). **NOTE:** RCLKDIR and SYN should not be changed at the same time. 0 Asynchronous mode 1 Synchronous mode (default)
10–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

12.5.4 Port Data Control Register 2 (AUDMUX_PDCR2)

PDCR2 is the Port Data Control Register for Port 2.

Address: AUDMUX_PDCR2 is 63FD_0000h base + Ch offset = 63FD_000Ch



AUDMUX_PDCR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

AUDMUX_PDCR2 field descriptions (continued)

Field	Description
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE = 1 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 101 Port 6 11x Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 MODE	Mode Select. This field selects the mode in which the port is to operate. The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDed together. RXDSEL is ignored. INMMASK determines which RxD signals are ANDed together. 0 Normal mode 1 Internal Network mode
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which the RxD signals are to be ANDed together for internal network mode. Bit 6 represents RxD from Port 6 and bit0 represents RxD from Port 1. 0 Includes RxDn for ANDing 1 Excludes RxDn from ANDing

12.5.5 Port Timing Control Register 3 (AUDMUX_PTCR3)

PTCR3 is the Port Timing Control Register for Port 3.

Address: AUDMUX_PTCR3 is 63FD_0000h base + 10h offset = 63FD_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFS DIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFS DIR	RFSEL[3:0]				RCLKDIR
Reset	1	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RCSEL[3:0]				SYN	0										
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

AUDMUX_PTCR3 field descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 x101 Port 6 x11x Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 x101 Port 6 x11x Reserved
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 x101 Port 6 x11x Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. **NOTE:** RCLKDIR and SYN should not be changed at the same time. 0 RxClk is an input. 1 RxClk is an output.

Table continues on the next page...

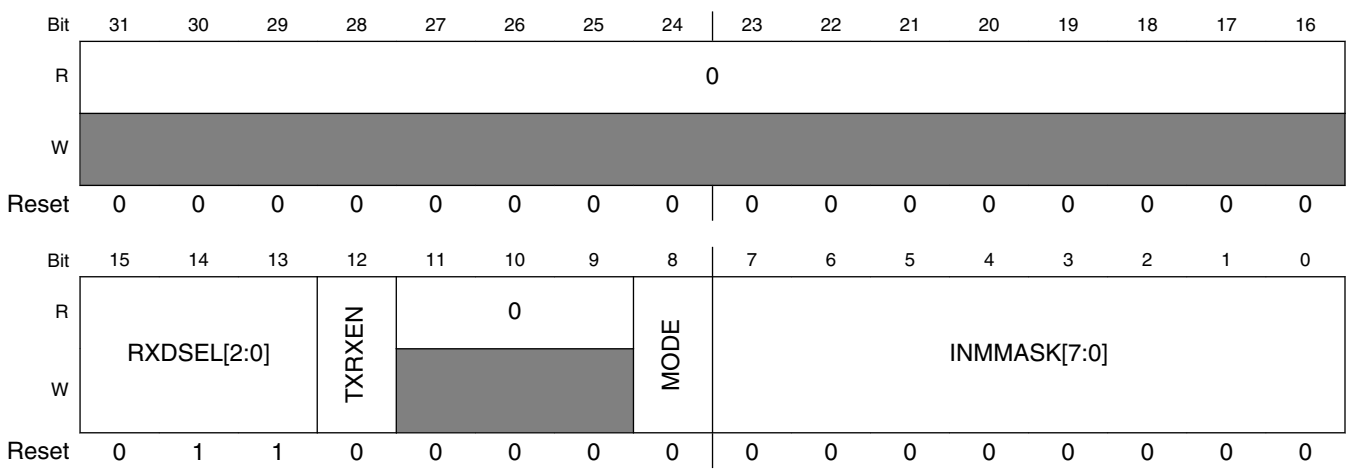
AUDMUX_PTCR3 field descriptions (continued)

Field	Description
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 x101 Port 6 x11x Reserved
11 SYN	Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). **NOTE:** RCLKDIR and SYN should not be changed at the same time. 0 Asynchronous mode 1 Synchronous mode (default)
10–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

12.5.6 Port Data Control Register 3 (AUDMUX_PDCR3)

PDCR3 is the Port Data Control Register for Port 3.

Address: AUDMUX_PDCR3 is 63FD_0000h base + 14h offset = 63FD_0014h



AUDMUX_PDCR3 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

AUDMUX_PDCR3 field descriptions (continued)

Field	Description
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE = 1 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 101 Port 6 11x Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 MODE	Mode Select. This field selects the mode in which the port is to operate. The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDed together. RXDSEL is ignored. INMMASK determines which RxD signals are ANDed together. 0 Normal mode 1 Internal Network mode
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which the RxD signals are to be ANDed together for internal network mode. Bit 6 represents RxD from Port 6 and bit0 represents RxD from Port 1. 0 Includes RxDn for ANDing 1 Excludes RxDn from ANDing

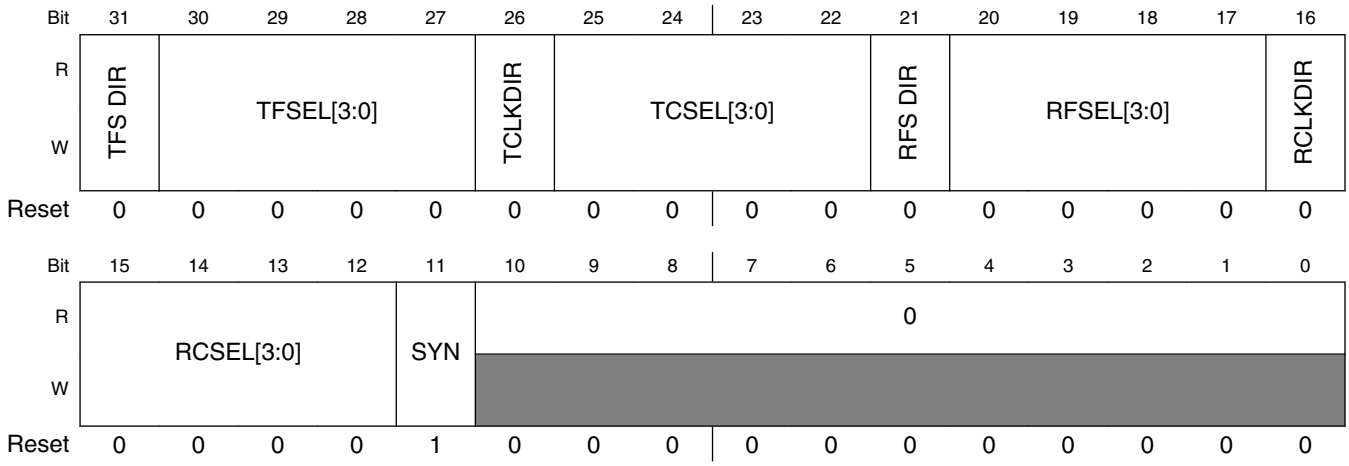
12.5.7 Port Timing Control Register n (AUDMUX_PTCR*n*)

PTCR*n* is the Port Timing Control Register for Port *n*, where *n* ranges from 4 through 7.

Addresses: AUDMUX_PTCR4 is 63FD_0000h base + 18h offset = 63FD_0018h

AUDMUX_PTCR5 is 63FD_0000h base + 20h offset = 63FD_0020h

AUDMUX_PTCR6 is 63FD_0000h base + 28h offset = 63FD_0028h



AUDMUX_PTCR*n* field descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 x101 Port 6 x11x Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 x101 Port 6 x11x Reserved
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync.

Table continues on the next page...

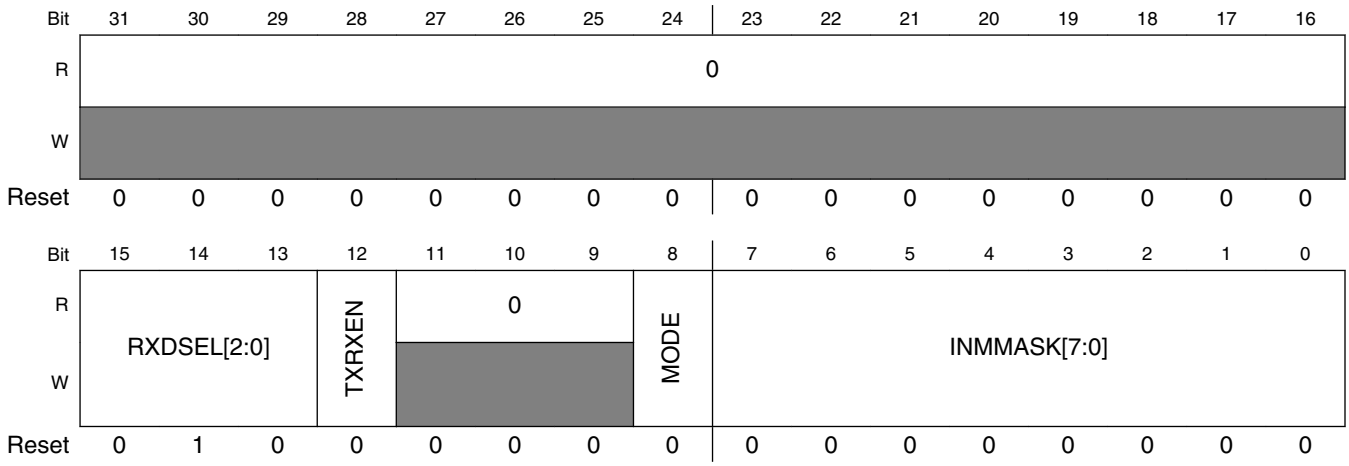
AUDMUX_PTCR*n* field descriptions (continued)

Field	Description
	0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 x101 Port 6 x11x Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. **NOTE:** RCLKDIR and SYN should not be changed at the same time. 0 RxClk is an input. 1 RxClk is an output.
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 x101 Port 6 x11x Reserved
11 SYN	Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). **NOTE:** RCLKDIR and SYN should not be changed at the same time. 0 Asynchronous mode 1 Synchronous mode (default)
10–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

12.5.8 Port Data Control Register 4 (AUDMUX_PDCR4)

PDCR4 is the Port Data Control Register for Port 4.

Address: AUDMUX_PDCR4 is 63FD_0000h base + 1Ch offset = 63FD_001Ch



AUDMUX_PDCR4 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE = 1 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 101 Port 6 11x Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 MODE	Mode Select. This field selects the mode in which the port is to operate. The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDed together. RXDSEL is ignored. INMMASK determines which RxD signals are ANDed together. 0 Normal mode 1 Internal Network mode
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which the RxD signals are to be ANDed together for internal network mode. Bit 5 represents RxD from Port 6 and bit0 represents RxD from Port 1.

Table continues on the next page...

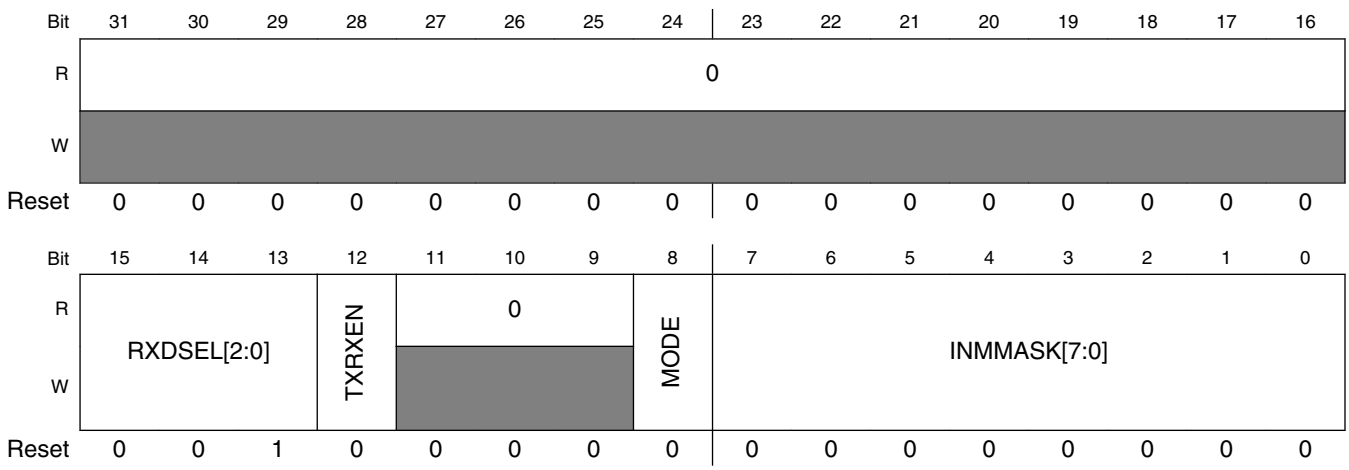
AUDMUX_PDCR4 field descriptions (continued)

Field	Description
	0 Includes RxDn for ANDing 1 Excludes RxDn from ANDing

12.5.9 Port Data Control Register 5 (AUDMUX_PDCR5)

PDCR5 is the Port Data Control Register for Port 5.

Address: AUDMUX_PDCR5 is 63FD_0000h base + 24h offset = 63FD_0024h



AUDMUX_PDCR5 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE = 1 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 101 Port 6 11x Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 MODE	Mode Select. This field selects the mode in which the port is to operate. The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDed together. RXDSEL is ignored. INMMASK determines which RxD signals are ANDed together.

Table continues on the next page...

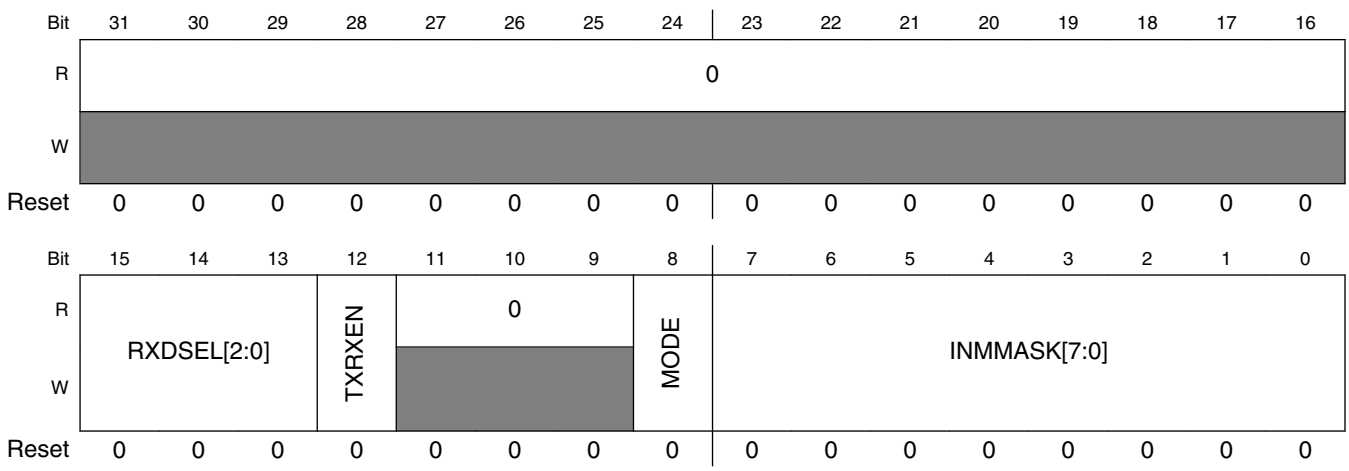
AUDMUX_PDCR5 field descriptions (continued)

Field	Description
	0 Normal mode 1 Internal Network mode
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which the RxD signals are to be ANDed together for internal network mode. Bit 5 represents RxD from Port 6 and bit0 represents RxD from Port 1. 0 Includes RxDn for ANDing 1 Excludes RxDn from ANDing

12.5.10 Port Data Control Register 6 (AUDMUX_PDCR6)

PDCR6 is the Port Data Control Register for Port 6.

Address: AUDMUX_PDCR6 is 63FD_0000h base + 2Ch offset = 63FD_002Ch



AUDMUX_PDCR6 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE = 1 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 101 Port 6 11x Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–9 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

AUDMUX_PDCR6 field descriptions (continued)

Field	Description
8 MODE	Mode Select. This field selects the mode in which the port is to operate. The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDed together. RXDSEL is ignored. INMMASK determines which RxD signals are ANDed together. 0 Normal mode 1 Internal Network mode
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which the RxD signals are to be ANDed together for internal network mode. Bit 5 represents RxD from Port 6 and bit0 represents RxD from Port 1. 0 Includes RxDn for ANDing 1 Excludes RxDn from ANDing

Chapter 13
AHB to IP Bridge (AIPSTZ)

13.1 Introduction

This section provides an overview of the AHB to IP Bridge (AIPSTZ). The peripheral bridge acts as an interface between the system bus and lower bandwidth IP Slave (IPS) bus peripherals.

NOTE

Although we use the mnemonic "AIPSTZ," the AHB to IP Bridge in this chip does not support TrustZone.

13.1.1 Features

The following list summarizes the key features of the bridge:

- The bridge supports the IPS slave bus signals. This interface is only meant for slave peripherals.
- The bridge supports 8-, 16-, and 32-bit IPS peripherals. (Accesses larger than the size of a peripheral are not supported, except to 32-bit memory.)
- The bridge supports a pair of IPS accesses for 64-bit and certain misaligned AHB transfers to 32-bit memory in 64-bit platforms.
- The bridge directly supports up to 32 16-Kbyte external IPS peripherals, and 2 global external IPS peripheral spaces. The bridge occupies 64 MBytes of total address space.
- The bridge provides configurable per-block and per-master access protections.
- Peripheral read transactions require a minimum of 2 hclk clocks, and unbuffered write transactions require a minimum of 3 hclk clocks.
- The bridge uses one single asynchronous reset and one global clock.

13.2 General Operation

The AHB to IP bridge is the interface between the AHB and on-chip IPS peripherals, which are sub-blocks containing readable/writable control and status registers.

The AHB master reads and writes these registers through the AIPSTZ. The bridge generates block enables, the block address, transfer attributes, byte enables and write data as inputs to the IPS peripherals. The bridge captures read data from the IPS interface and drives it on the AHB.

It occupies a 64-Mbyte portion of the address space. 63.5 Mbytes are available for off-platform devices. The register maps of the IPS peripherals are located on 16-Kbyte boundaries. Each IPS peripheral is allocated one 16K-byte block of the memory map, and is activated by one of the block enables from the bridge. Up to thirty-two 16-Kbyte external IPS peripherals may be implemented, occupying contiguous blocks of 16 Kbytes. Two global external IPS block enables are available for the remaining 63 Mbytes of address space to allow for customization and expansion of addressed peripheral devices. In addition, a single "non-global" block enable is also asserted whenever any of the thirty-two non-global block enables is asserted.

The bridge is responsible for indicating to IPS peripherals if an access is in supervisor or user mode. It may block user mode accesses to certain IPS peripherals or it may allow the individual IPS peripherals to determine if user mode accesses are allowed. In addition, peripherals may be designated as write-protected.

All peripheral devices are expected to only require aligned accesses equal to or smaller in size than the peripheral size. An exception to this rule is supported for 32-bit peripherals to allow memory to be placed on the IPS.

13.2.1 AIPSTZ Registers

This section provides information on the registers of the AIPS bridge.

13.2.2 Overview

There are eleven registers that control the AIPS bridge. All registers are 32-bit registers and can only be accessed in supervisor mode by specifically configured bus masters such as the core processor. Additionally, these registers must only be read from or written to by a 32-bit aligned access. The bridge registers are mapped into the PACR0 address space.

Two system clocks are required for read accesses and three system clocks are required for write accesses to the bridge registers.

13.2.3 Control Registers

The memory map for the AIPS program-visible registers is shown in the table below.

The MPROT fields of the AIPSTZ_MPR and the AIPSTZ_PACR and AIPSTZ_OPACR registers are 4 bits in width. Some bits may be reserved depending on device.

Table 13-1. AIPSTZ Register Memory Map

PACR0_ Offset	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0xBASE__0000	MPROT0	MPROT1	MPROT2	MPROT3	MPROT4	MPROT5	MPROT6	MPROT7
0xBASE_0004	MPROT8	MPROT9	MPROT10	MPROT11	MPROT12	MPROT13	MPROT14	MPROT15
0xBASE_0020	Reserved for on-platform Registers							
0xBASE_0024								
0xBASE_0028								
0xBASE_002c								
0xBASE_0040	OPACR0	OPACR1	OPACR2	OPACR3	OPACR4	OPACR5	OPACR6	OPACR7
0xBASE_0044	OPACR8	OPACR9	OPACR10	OPACR11	OPACR12	OPACR13	OPACR14	OPACR15
0xBASE_0048	OPACR16	OPACR17	OPACR18	OPACR19	OPACR20	OPACR21	OPACR22	OPACR23
0xBASE_004c	OPACR24	OPACR25	OPACR26	OPACR27	OPACR28	OPACR29	OPACR30	OPACR31
0xBASE_0050	OPACR32	OPACR33	Reserved					

13.3 Register Descriptions

13.3.1 Master Privilege Registers

Each AIPSTZ_MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the platform, as well as specifying whether write accesses from this master are bufferable.

The registers provide one field per bus master, where field 15 corresponds to master 15, field 14 to master 14,... field 0 to master 0 (typically the processor core). The master index allocation is shown in Table 13-4.

Table 13-2. MPROT Field

Bit	3	2	1	0
Field	MBW	-	-	MPL
Reset	*	*	*	*
Read/Write	Read/Write			

Table 13-3. Master Protection Field Descriptions

Name	Description	Settings
3 **(MBW)**	**Master Buffer Writes** - This bit determines whether the AIPSTZ is enabled to buffer writes from this master.	0 Write accesses from this master are not bufferable 1 Write accesses from this master are allowed to be buffered
2 -	Reserved	0
1 -	Reserved	0
0 **(MPL)**	**Master Privilege Level** - This bit determines how the privilege level of the master is determined.	0 Accesses from this master are forced to user-mode (**ips_supervisor_access** is forced to zero) regardless of the **hprot[1]** access attribute. 1 Accesses from this master are not forced to user-mode. The **hprot[1]** access attribute is used directly to determine **ips_supervisor_access**.

Table 13-4. Master index allocation

Master index	Master name	Comments
Master 0	All masters excluding ARM core, SDMA and CAAM	Share the same number allocation.
Master 1	ARM CORE	
Master 2	CAAM	

Table continues on the next page...

Table 13-4. Master index allocation (continued)

Master index	Master name	Comments
Master 3	SDMA	
Master 4-15	Reserved	

13.3.2 Off-Platform Peripheral Access Control Registers (AIPSTZ_OPACRs)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_PACR has the following format:

Table 13-5. AIPSTZ_OPACR Fields

Bit	3	2	1	0
Field	BW	SP	WP	-
Reset	0	1	0	0
Read/Write	Read/Write			

Table 13-6. Peripheral Access Control Register Field Descriptions

Name	Description	Settings
3 **(BW)**	**Buffer Writes** - This bit determines whether write accesses to this peripheral are allowed to be buffered.[1]	0 Write accesses to this peripheral are not bufferable by the AIPSTZ. 1 Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
2 **(SP)**	**Supervisor Protect** - This bit determines whether the peripheral requires supervisor privilege level for access.	0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.
1 (WP)	**Write Protect** - This bit determines whether the peripheral allows write accesses	0 This peripheral allows write accesses. 1 This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.
0 -	Reserved	0

1. Buffered writes are not available for AIPSTZ. This bit should be set to '0'.

13.4 Functional Description

The AIPS bridge serves as a protocol translator between the AHB system bus and the IP bus. Support is provided for generating a pair of 32-bit IP bus accesses when targeted by a 64-bit system bus access, or a misaligned access which crosses a 32-bit boundary. No other bus-sizing access support is provided.

13.5 Access Protections

The AIPS bridge provides programmable access protections for both masters and peripherals. It allows the privilege level of a master to be overridden, forcing it to user-mode privilege. Peripherals may require supervisor privilege level for access and may be write-protected.

13.6 Access Support

Aligned 64-bit accesses, aligned and misaligned word and half word accesses, as well as byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the IPS. Peripheral registers must not be misaligned, although no explicit checking is performed by the AIPS bridge.

The bridge will perform two IPS transfers for 64-bit accesses, word accesses with byte offsets of 1, 2, or 3, and for half word accesses with a byte offset of 3. All other accesses will be performed with a single IPS transfer.

Only aligned half word and byte accesses are supported for 16-bit peripherals. All other accesses types are unsupported, and results of such accesses are undefined. They are not terminated with an error response.

Only byte accesses are supported for 8-bit peripherals. All other accesses types are unsupported, and results of such accesses are undefined. They are not terminated with an error response.

13.7 Initialization Information

The AIPS bridge should be programmed before use. The following registers should be initialized: the Master Privilege Registers (AIPSTZ_MPRs), the Peripheral Access Control registers (AIPSTZ_PACRs), and the Off-platform Peripheral Access Control registers (AIPSTZ_OPACRs) described in 3.2 Control Registers.

Chapter 14
32-Bit Correcting ECC Accelerator (BCH)

14.1 Introduction

This chapter describes the hardware Bose Ray-Choudhury Hocquenghem (BCH) ECC accelerator. It provides detailed descriptions of how to use the accelerator and programmable registers (described in the Programmable Registers section).

From a programming standpoint, all data transfer to/from memory is handled by the BCH block directly instead of using DMA for data write operations. DMA is still used in programming the GPMI control registers.

14.2 Overview

The hardware ECC accelerator provides a forward error-correction function for improving the reliability of various storage media that may be attached to the chip. For example, modern high-density NAND flash devices presume the existence of forward error-correction algorithms to correct some soft and/or hard bit errors within the device, allowing for higher device yields and, therefore, lower NAND device costs.

The Bose, Ray-Chaudhuri, Hocquenghem (BCH) Encoder and Decoder module is capable of correcting from 2 to 32 single bit errors within a block of data no larger than about 1900 bytes for GF(2^14) (1024 bytes is typical), and about 900 bytes for GF(2^13) (512 bytes is typical). BCH is capable of correcting these errors in applications such as protecting data and resources stored on modern NAND flash devices. The correction level in the BCH block is programmable to provide flexibility for varying applications and configurations of flash page size. The design can be programmed to encode protection of 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, or 32-bit errors when writing flash and to correct the corresponding number of errors on decode. The correction level when decoding MUST be programmed to the same correction level as was used during the encode phase.

BCH-codes are a type of block-code, which implies that all error-correction is performed over a block of N-symbols. The BCH operation will be performed over GF(2^{13} = 8192) or GF(2^{14} = 16384), which is the Galois Field consisting of 8191 or 16383 one-bit symbols. BCH-encoding (or encode for any block-code) can be performed by two algorithms: systematic encoding or multiplicative encoding. Systematic encoding is the process of reading all the symbols which constitute a block, dividing continuously these symbols by the generator polynomial for the GF(8192) or GF(16384) and appending the resulting *t* parity symbols to the block to create a BCH codeword (where *t* is the number of correctable bits).

The BCH encode process creates *t**13 (or *t**14)-bit parity symbols for each data block when the data is written to the flash device. The parity symbols are written to the flash device after the corresponding data block, and together these are collectively called the codeword. The codeword can be used during the decode process to correct errors that occur in either the data or parity blocks.

The BCH decoder processes codewords in a 4-step fashion:

- Syndrome Calculation (SC): This is the process of reading in all of the symbols of the codeword and continuously dividing by the generator polynomial for the field. 2**t* syndromes must be calculated for each codeword and inspection of the syndromes determines if there are errors: a non-zero set of syndromes indicates one or more errors. This process is implemented parallel hardware to minimize processing time since it must be done every time the decode is performed.
- Key Equation Solver (KES): The syndromes represent 2t-linear equations with 2t-unknown variables. The process of solving these equations and selecting from the numerous solutions constitutes the KES module. When the KES block completes its operations, it generates an error locator polynomial (sigma) that is used in the proceeding block to determine the locations and values of the errors.
- Chien Search (CS): This block takes input from the KES block and uses the Chien Algorithm for finding the locations of the errors based on the error locator polynomial. The method basically involves substituting all 8191 symbols from the GF(8192) or 16383 symbols from the GF(16383) into the locator polynomial. All evaluations that produce a zero solution indicate locations of the various errors. Since each located error corresponds to a single bit, the bit in the original data may be corrected by simply flipping the polarity of the incorrect location.
- Correction: this block has to convert the symbol index and mask information to memory byte indexes and masks.

The BCH block was designed to operate in a pipelined fashion to maximize throughput. Aside from the initial latency to fill the pipeline stages, the BCH throughput is about 7/4 cycles/byte. Thus, the bottleneck in performing NAND reads and error corrections is the BCH rate. Current GPMI read rates are approximately 1/2 cycles/byte maximally for the

current generation of NANDs. Fortunately, BCH has a different master clock from GPMI, this gives some flexibity to match the throughput rate. The ARM platform is not directly involved in generating parity symbols, checking for errors, or correcting them.

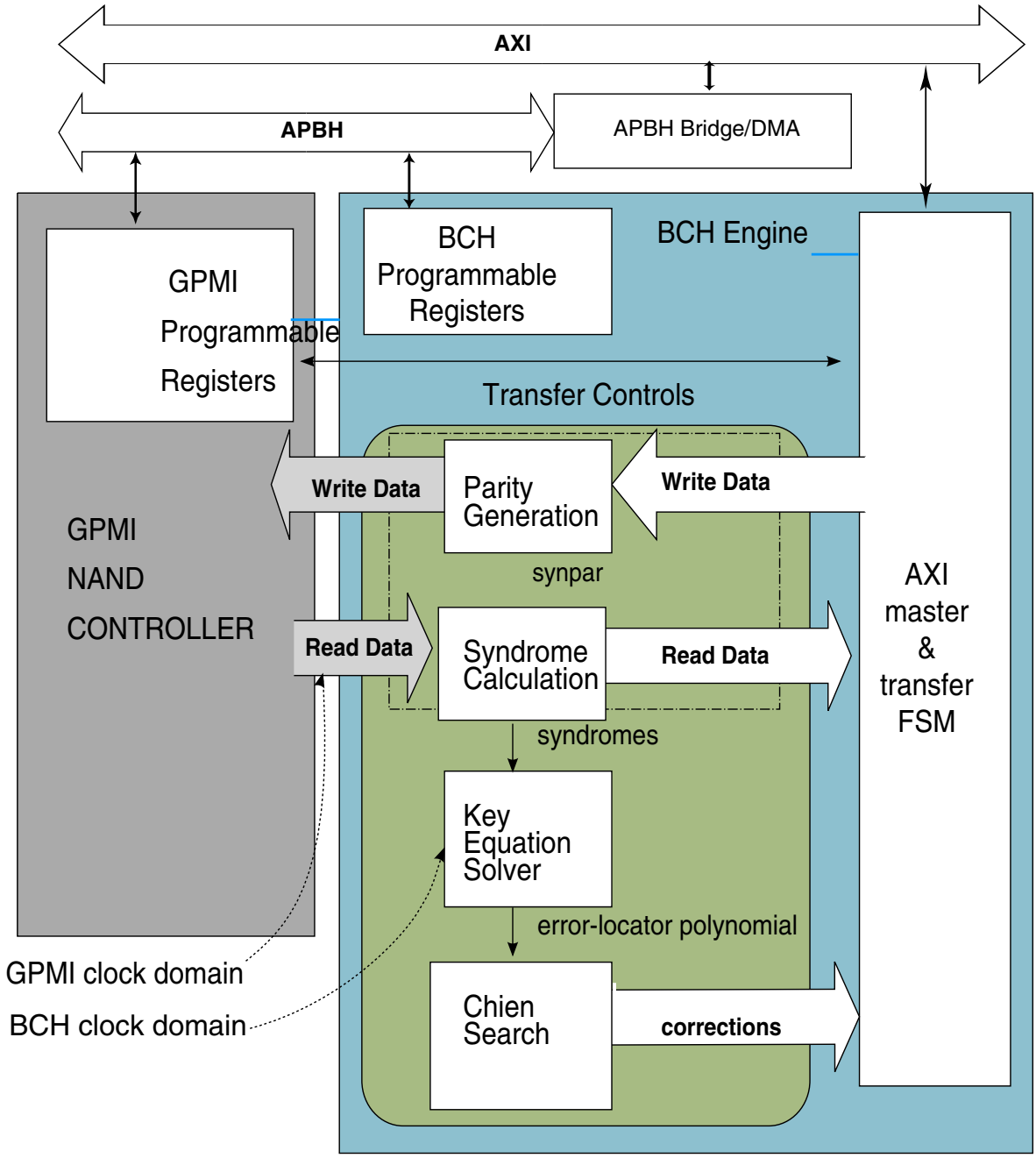


Figure 14-1. Hardware BCH Accelerator

14.3 Operation

Before performing any NAND flash read or write operations, software should first program the BCH's flash layout registers (see Flash Page Layout) to specify how data is to be formatted on the flash device. The BCH hardware allows full programmability over the flash page layout to enable users flexibility in balancing ECC correction levels and ever-changing flash page sizes.

To initiate a NAND flash write, software will program a GPMI DMA operation. The DMA need only program the GPMI control registers (and handle the requisite flash addressing handshakes) since the BCH will handle all data operations using its AXI bus interface. The BCH will then send the data to the GPMI controller to be written to flash as it computes the parity symbols. At the end of each data block the BCH will insert the parity symbols into the data stream so that the GPMI sees only a continuous stream of data to be written.

NAND flash read operations operate in a similar manner. As the GPMI controller reads the device, all data is sent to the BCH hardware for error detection/correction. The BCH controller writes all incoming read data to system memory and in parallel computes the syndromes used to detect bit errors. If errors are detected within a block, the BCH hardware activates the error correction logic to determine where bit errors have occurred and ultimately correct them in the data buffer in system memory. After an entire flash page has been read and corrected, the BCH will signal an interrupt to the ARM platform.

Figure 14-2 indicates how data read from the GPMI is operated on within the BCH hardware. As the BCH receives data from the GPMI (top row) it is written to memory by the BCH's Bus Interface Unit (BIU) (second row). For blocks requiring correction, the KES logic will be activated after the entire block has been received. Once the error locator polynomial has been computed, the corrections are determined by the Chien Search and fed back to the BIU, which performs a read/modify/write operation on the buffer in memory to correct the data

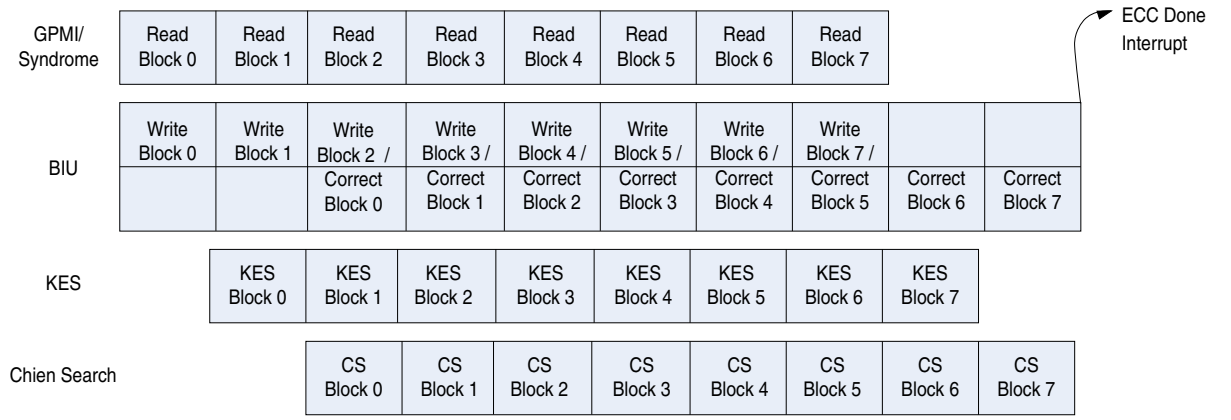


Figure 14-2. Block Pipeline while Reading Flash

14.3.1 BCH Limitations and Assumptions

BCH limitations and assumptions are as follows:

- The BCH is programmable to support 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, and 32-bit error correction. ECC0 is supported as a passthrough, non-correcting mode.
- Data block sizes must be a multiple of 4 bytes and be aligned in system memory.
- The BCH supports a programmable number of metadata/auxiliary data bytes, from 0 to 255.
- Metadata will be written at the beginning of the flash page to facilitate fast access for filesystem operations.
- Metadata may be treated as an independent block for ECC purposes or combined with the first data block to conserve bits in the flash.
- The BCH does not support a partial page write (this can be accomplished by programming the BCH layout registers such that the BCH only sees a portion of the page).
- Flash read operations can read the entire page or the first block on the page.
- The BCH also supports a memory-to-memory mode of operation that does not require the use of DMA or the GPMI.

14.3.2 Flash Page Layout

The BCH supports a fully programmable flash page layout. The BCH maintains 4 independent layout registers that can describe four completely different NAND devices or layouts. When the BCH initiates an operation, it selects one of the layouts by using the chip select as an index into the BCH_LAYOUTSELECT register that determines which layout should be used for the operation.

Three possible (generic) flash layout schemes are supported, as indicated in Figure 14-3. (In each case, the metadata size may also be programmed to 0 bytes). Metadata may either be combined with the first block of data or the size of the first data block can be programmed to 0 to allow the metadata to be protected by its own ECC parity bits.

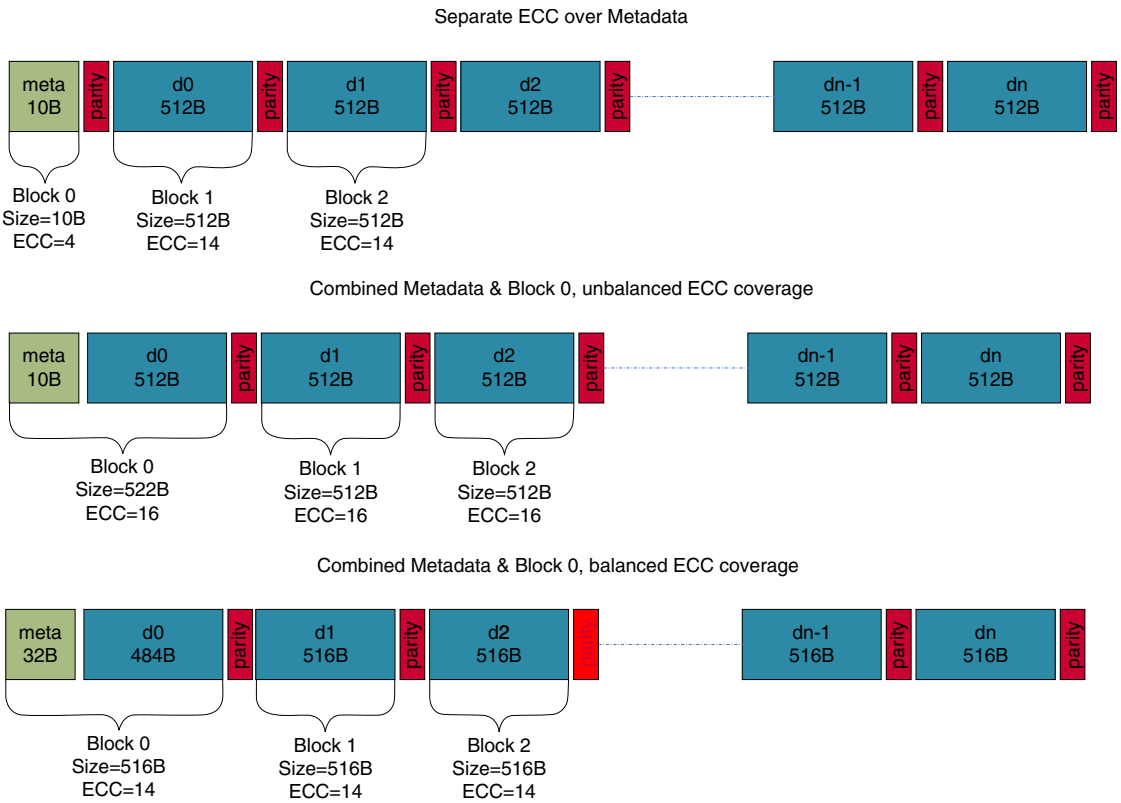


Figure 14-3. FLASH Page Layout Options

Each layout is determined by a pair of registers that define the following parameters:

- DATA0_SIZE: Indicates the number of data bytes in the first block on the page (this should not include parity or metadata bytes). This should be set to 0 when the metadata is to be covered separately with its own ECC. This MUST be a multiple of 4 bytes.
- ECC0: Indicates the ECC level to be used for the first block on the flash (data0+metadata).
- META_SIZE: indicates the number of bytes (from 0-255) that are stored as metadata.
- NBLOCKS: Indicates the number of subsequent "DATAN" blocks on the flash, or the number of blocks following the DATA0 block.
- DATAN_SIZE: Indicates the number of data bytes in all subsequent data blocks. This MUST be a multiple of 4 bytes.
- ECCN: Indicates the ECC level to be used for the subsequent data blocks.

- GF0 or GFN: Indicates the Galois field the meta / data blocks are using.
- PAGE_SIZE: Indicates the total number of bytes available per page on the physical flash device. This includes the spare area and is typically 4096 + 128, 4096 + 218, or 2048 + 64 bytes.

14.3.3 Determining the ECC layout for a Device

Since the BCH is programmable, a system can trade off ECC levels for flash size and layout configurations. The following examples indicate how to determine a valid layout based on the required storage space and flash size. For all cases, the size of the parity will be 13 (or 14 for GF(2^{14}))*ECC level *bits*-- thus for ECC8, 13 (or 14) bytes are required (per block).

14.3.3.1 4K + 218 Flash, 10 bytes Metadata, 512 byte Data Blocks, Separate Metadata, Assuming GF(2^13)

In this case, we have 8 data blocks each consisting of 512 bytes. Since the flash has 218 "spare" bytes (1744 bits), we first estimate an ECC level for the data blocks by first subtracting the number of metadata bytes from the spare bytes (218 - 10 = 208 bytes = 1664 bits) then dividing the number of bits by 8 (number of blocks) and then by 13 (bits per ECC level).

$$(218 - 10) \times 8 = \frac{1664}{13(8)} = 16$$

Thus all the data blocks could be covered by ECC16 if the metadata had no parity. This isn't acceptable, so assume ECC14 for all the data blocks. We now calculate the number of free bits for the metadata parity as

$$1664 - (14) \times 13 \times 8 = 208$$

Thus 208 bits remain for metadata parity. Dividing by 13 (bits/ECC) gives 16, thus the metadata can be covered with ECC16. The settings for this device would then be

Table 14-1. Settings for 4K+218 FLASH

Setting	Value
PAGE_SIZE	4096+218=4314=0x10DA
META_SIZE	10=0x0A

Table continues on the next page...

Table 14-1. Settings for 4K+218 FLASH (continued)

Setting	Value
DATA0_SIZE	0
ECC0	16=0x10
GF0	GF(2^13)
DATAN_SIZE	512=0x200 (in register interface, assigned as 0x80)
ECCN	14=0x0E
GFN	GF(2^13)
NBLOCKS	8

14.3.3.2 4K + 128 flash, 10 bytes Metadata, 1024 byte Data Blocks, Separate Metadata, Assuming GF(2^14) for Data and GF(2^13) for Metadata

This flash will have 118 bytes available for ECC (after subtracting the metadata size), thus 944 bits. Dividing by 4 x 14 (number of blocks x ECC level) we get 17.75, thus we can support ECC16 on the data blocks. The number of free spare bits becomes 944 - 16 x 4 x 14 = 944 - 896 = 48, divided by 13 = 3.69, thus the metadata can be covered by ECC2.

Table 14-2. Settings for 4K + 128 FLASH

Setting	Value
PAGE_SIZE	4096+128=4224=0x1080
META_SIZE	10=0x0A
DATA0_SIZE	0
ECC0	2
GF0	GF(2^13)
DATAN_SIZE	1024=0x400 (in register interface, assigned as 0x100)
ECCN	16
GFN	GF(2^14)
NBLOCKS	4

In this case, there will be additional unused spare bits, with the BCH will pad out with zeros.

14.3.4 Data buffers in System Memory

While the data on the flash is interleaved with parity symbols, the BCH assumes that the data buffers in memory are contiguous. Metadata read from the flash will be stored to the location pointed to by the GPMI_AUXILIARY register and data will be written to the address specified in the GPMI_PAYLOAD register as is shown in Figure 14-4 where the block length is 512 bytes for example. Since the number of blocks on a flash page is programmable, the BCH also writes individual block correction status to the auxiliary pointer at the word-aligned address following the end of the metadata. Optionally, the computed syndromes may also be written to the auxiliary area if the BCH_CTRL[DEBUGSYNDROME] bit is set in the control register.

As blocks complete processing, the bus master will accumulate the status for each block and write it to the auxiliary data buffer following the metadata. The metadata area will be padded with 0's until the next word boundary and the status for blocks 0-3 will be written to the next word. Status for subsequent blocks will then be written to the buffer. Status for the first block (metadata block) is also stored in the BCH_STATUS[STATUS_BLK0] bit field. The completion codes for the blocks are indicated in the Table 14-3. Note that the definition of the bytes and their ordering in the auxiliary and payload storage areas are user defined. When this data is read back from the flash and put into memory, it will resemble the original buffer that was written out to the flash.

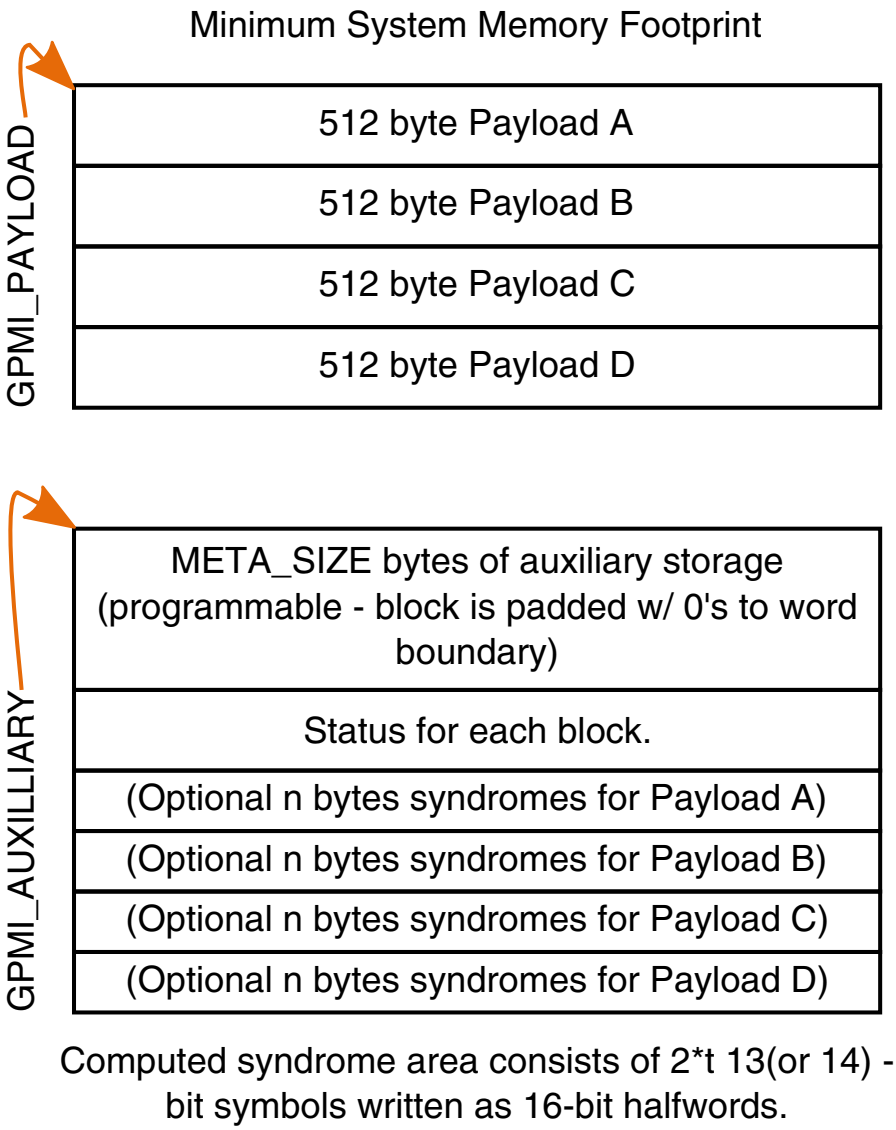
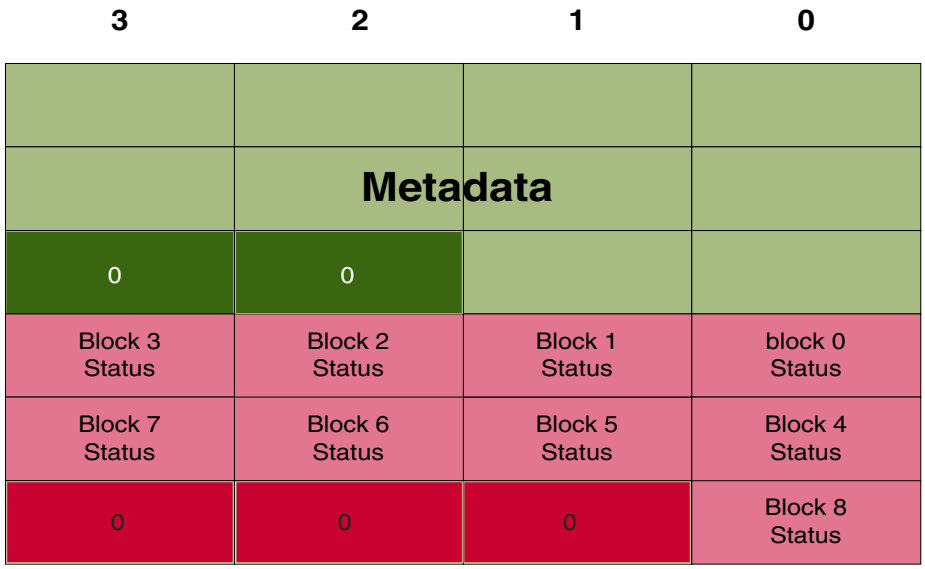


Figure 14-4. BCH Data Buffers in Memory

Table 14-3. Status Block Completion Codes

Code	Description
0xFF	Block is erased
0xFE	Block is uncorrectable
0x00	No errors found
0x01-0x20	Number of errors corrected

Figure 14-5 shows the layout of the bytes within the status field.



Status bytes are allocated based on the NBLOCKS programmed into the flash format register. The number of status bytes will be computed by the NBLOCKS+1. The status area will be padded with zeros to the next word boundary.

Syndrome data written for debug purposes will follow the end of the status block.

Figure 14-5. Memory-to-Memory Operations

14.4 Memory to Memory (Loopback) Operation

The BCH supports a memory-to-memory mode of operation where both the encoded and decoded buffers reside in system memory. This can be useful for applications where data must be protected by ECC, but the storage device does not reside on the GPMI bus.

The BCH operation in memory to memory mode is much simpler than in GPMI mode since DMAs are not required to manage the operation. Instead software simply writes the BCH_DATAPTR and BCH_METAPTR with the addresses of the data and metadata (auxiliary) buffers and the BCH_ENCODEPTR with the address of the buffer for encoded data. To initiate the operation, software simply sets the M2M_ENCODE and M2M_ENABLE bits in the control register. The BCH can be programmed to either issue an interrupt at the end of the operation or software may poll the status bits for completion.

Memory to memory decode operations work in a similar manner. The encoded data address is written to the BCH_ENCODEPTR and the data and meta pointers are written to buffers that correspond to the desired decoded data addresses. To initiate a decode, software must set the M2M_ENCODE bit to 0 while writing the M2M_ENABLE bit.

Note that the addresses written to the BCH_DATAPTR, BCH_METAPTR and BCH_ENCODEPTR registers should always be aligned on a 4 byte boundary, one should note that the address appearing in the interface registers are already removed the 2 always-zero LSBs.

14.5 Programming the BCH/GPMI Interfaces

Programming the BCH for NAND operations consists largely of disabling the soft reset and clock bits (SFTRST and CLKGATE) from the BCH_CTRL register and then programming the flash layout registers to correspond to the format of the attached NAND device(s). The BCH_LAYOUTSELECT register should also be programmed to map the chip select of each attached device into one of the four layout registers.

The bulk of the programming is actually applied to the GPMI via PIO operations embedded in DMA command structures. The DMA will perform all the requisite handshaking with the GPMI interface to negotiate the address portion of the transfer, then the BCH will handle all the movement of data from memory to the GPMI (writes) or the GPMI to memory (reads). The BCH will direct all data blocks to the buffer pointed to by the PAYLOAD_BUFFER and the metadata will be written to the AUXILIARY_BUFFER. Both of these registers are located in the GPMI PIO data space and are communicated to the BCH hardware at the beginning of the transfer. Thus, the normal multi-NAND DMA based device interleaving is preserved, i.e., four NANDs on four separate chip selects can be scheduled for read or write operations using the BCH. Whichever channel finishes its ready wait first and enters the DMA arbiter with its lock bit set will "own" the GPMI command interface and through it will own the BCH resources for the duration of its processing.

14.5.1 BCH Encoding for NAND Writes

The BCH encoder flowchart in Figure 14-6 shows the detailed steps involved in programming and using the BCH encoder. This flowchart shows how to use the BCH block with the GPMI.

To use the BCH encoder with the GPMI's DMA, create a DMA command chain containing ten descriptor structures, as shown in Figure 14-8 and detailed in the DMA structure code example that follows it in DMA Structure Code Example 1. The ten descriptors perform the following tasks:

1. Disable the BCH block (in case it was enabled) and issue NAND write setup command byte (under "CLE") and address bytes (under "ALE").
2. Configure and enable the BCH and GPMI blocks to perform the NAND write.

3. Disable the BCH block and issue NAND write execute command byte (under "CLE").
4. Wait for the NAND device to finish writing the data by watching the ready signal.
5. Check for NAND timeout through "PSENSE".
6. Issue NAND status command byte (under "CLE").
7. Read the status and compare against expected.
8. If status is incorrect/incomplete, branch to error handling descriptor chain.
9. Otherwise, write is complete and emit GPMI interrupt.

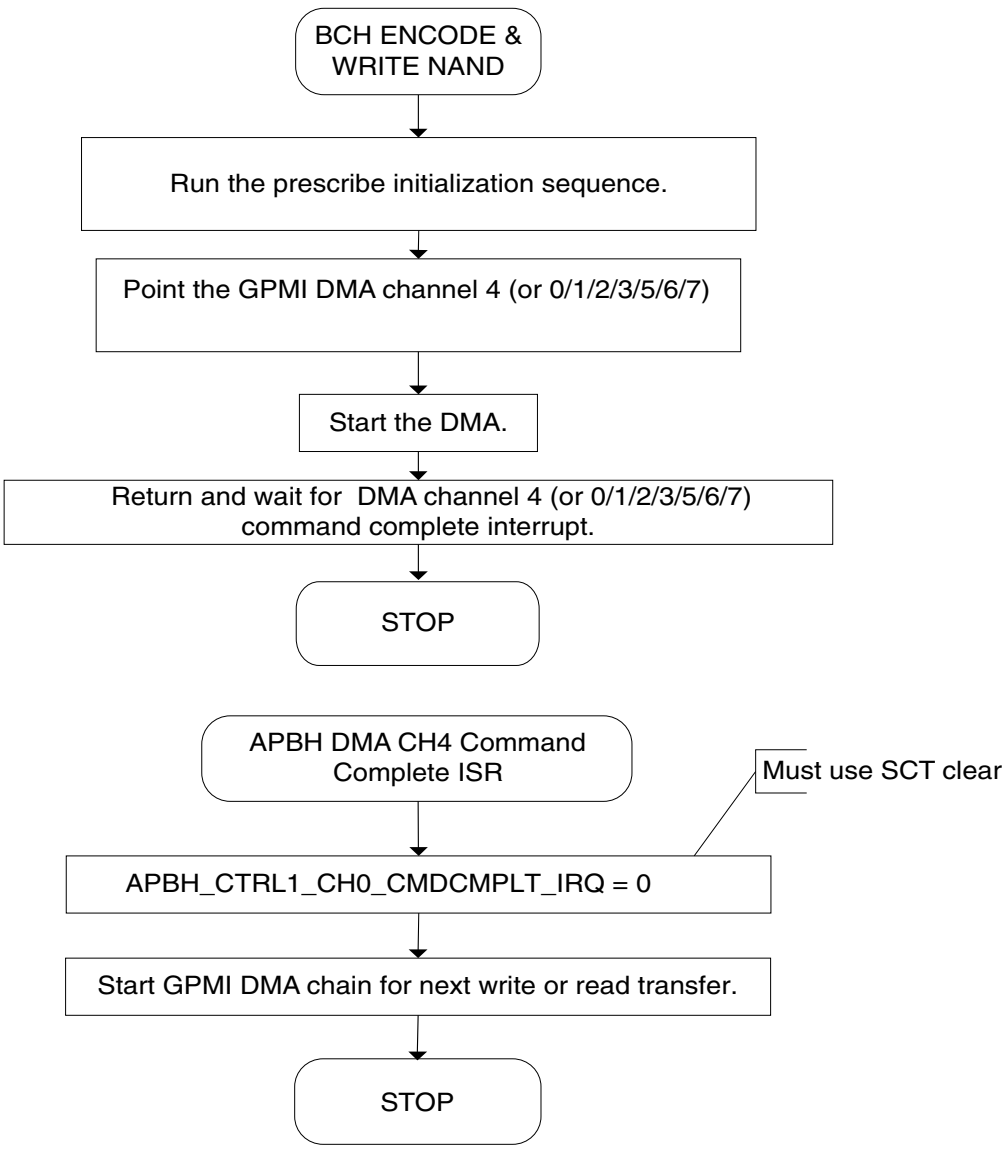


Figure 14-6. BCH Encode Flowchart

Descriptor Legend

NEXT CMD ADDR										
CMD	<=	xfer_count	cmdwords	wait4endcmd	semaphore	nandwait4ready	nandlock	irqoncmplt	chain	command
BUFFER ADDR										
HW_GPMI_CTRL0	<=	command_mode	word_length	lock_cs	CS	address	address_increment	xfer_count		
HW_GPMI_COMPARE	<=	mask				reference				
HW_GPMI_ECCCTRL	<=	ecc_cmd			enable_ecc				buffer_mask	
HW_GPMI_ECCCOUNT										
HW_GPMI_PAYLOAD										
HW_GPMI_AUXILIARY										

Figure 14-7. BCH DMA Descriptor Legend

Refer to this BCH DMA Descriptor Legend when examining Figure 14-8 and Figure 14-10.

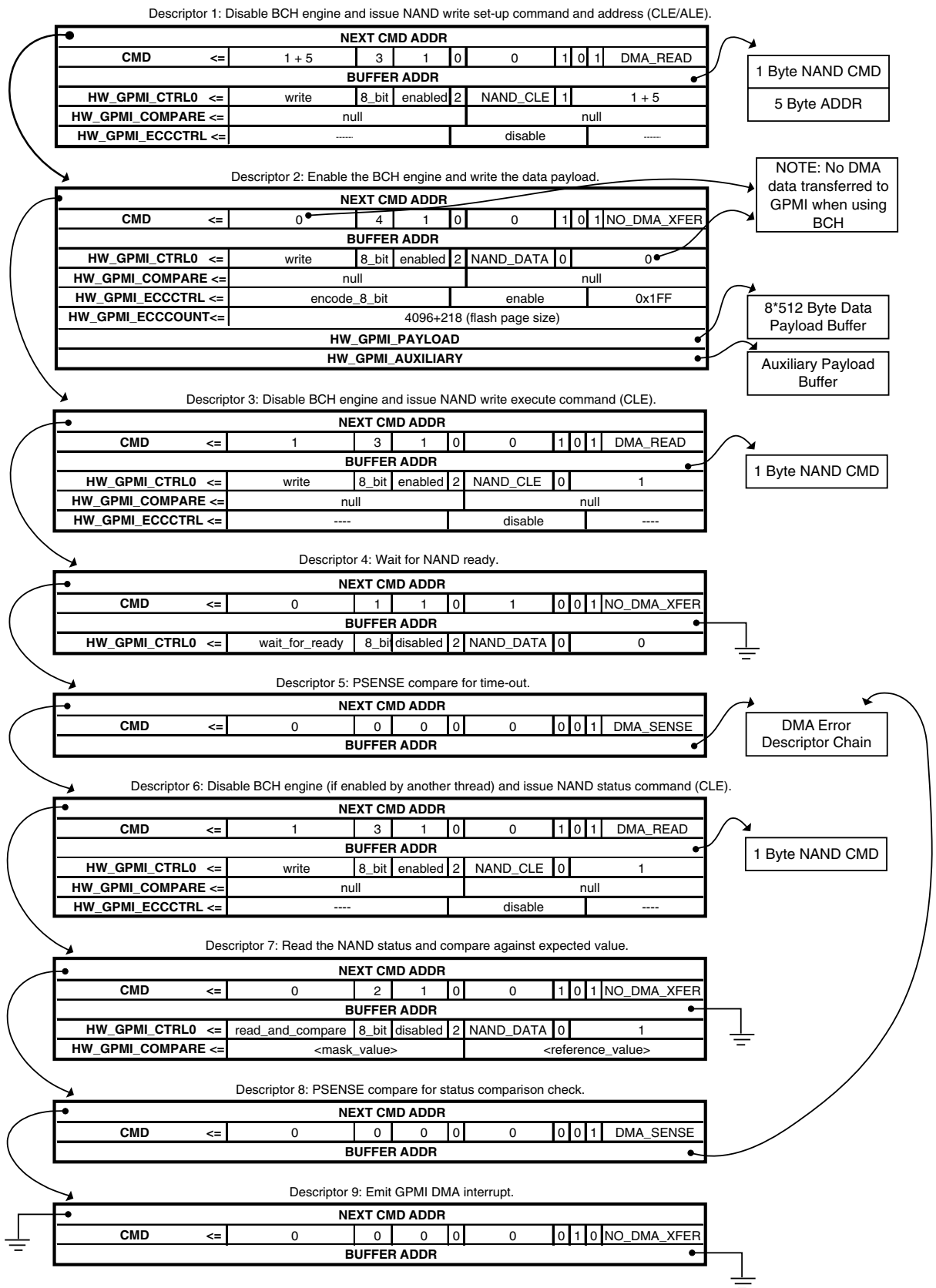


Figure 14-8. BCH Encode DMA Descriptor Chain

To interpret the fields in this BCH Encode DMA Descriptor Chain diagram, see Figure 14-7 for the descriptor legend.

14.5.1.1 DMA Structure Code Example 1

The following code sample illustrates the coding for one write transaction involving 4096 bytes of data payload (eight 512-byte blocks) and 10 bytes of auxiliary payload (also referred to as *metadata*) to a 4 K NAND page sitting on GPMI CS2.

```
//-----------------------------------------------------------------------------
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//-----------------------------------------------------------------------------
typedef struct {
  // DMA related fields
  unsigned int dma_nxtcmdar;
  unsigned int dma_cmd;
  unsigned int dma_bar;
  // GPMI related fields
  unsigned int gpmi_ctrl0;
  unsigned int gpmi_compare;
  unsigned int gpmi_eccctrl;
  unsigned int gpmi_ecccount;
  unsigned int gpmi_data_ptr;
  unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;
//-----------------------------------------------------------------------------
// allocate 10 descriptors for doing a NAND ECC Write
//-----------------------------------------------------------------------------
GENERIC_DESCRIPTOR write[10];
//-----------------------------------------------------------------------------
// DMA descriptor pointer to handle error conditions from psense checks
//-----------------------------------------------------------------------------
unsigned int * dma_error_handler;
//-----------------------------------------------------------------------------
// 8 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
//   byte 0 is write setup command
//   bytes 1-5 is the NAND address
//   byte 6 is write execute command
//   byte 7 is status command
//-----------------------------------------------------------------------------
unsigned char nand_cmd_addr_buffer[8];
//-----------------------------------------------------------------------------
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int write_payload_buffer[(4096/4)];
//-----------------------------------------------------------------------------
// 65 byte meta-data to be written to NAND
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int write_aux_buffer[65];
//-----------------------------------------------------------------------------
// Descriptor 1: issue NAND write setup command (CLE/ALE)
//-----------------------------------------------------------------------------
write[0].dma_nxtcmdar = &write[1];                          // point to the next descriptor
write[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1 + 5)|   // 1 byte command, 5 byte address
                   BF_APBH_CHn_CMD_CMDWORDS      (3)    |   // send 3 words to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |   // wait for command to finish
before
                                                             //    continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
```

```
                    BF_APBH_CHn_CMD_NANDLOCK      (1)    |   // prevent other DMA channels
from
                                                           //    taking over
                    BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                    BF_APBH_CHn_CMD_CHAIN         (1)    |   // follow chain to next command
                    BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to
NAND
write[0].dma_bar = &nand_cmd_addr_buffer;           // byte 0 write setup, bytes 1 - 5 NAND
address
// 3 words sent to the GPMI
write[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                      BF_GPMI_CTRL0_CS                 (2)       | // must correspond to
NAND CS used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (1)       | // send command and
address
                      BF_GPMI_CTRL0_XFER_COUNT         (1 + 5);    // 1 byte command, 5 byte
address
write[0].gpmi_compare = NULL;                          // field not used but necessary to
set eccctrl
write[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 2: write the data payload (DATA)
//-------------------------------------------------------------------------
write[1].dma_nxtcmdar = &write[2];                       // point to the next descriptor
write[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT      (0)|  // NOTE: No DMA data transfer
                   BF_APBH_CHn_CMD_CMDWORDS        (4)|  // send 4 words to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD     (1)|  // Wait to end
                   BF_APBH_CHn_CMD_SEMAPHORE       (0)|
                   BF_APBH_CHn_CMD_NANDWAIT4READY  (0)|
                   BF_APBH_CHn_CMD_NANDLOCK        (1)|  // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT      (0)|
                   BF_APBH_CHn_CMD_CHAIN           (1)|  // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_NO_XFER);  // No data transferred
write[1].dma_bar = &write_payload_buffer;                // pointer for the 4K byte
data area
// 4 words sent to the GPMI
write[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                      BF_GPMI_CTRL0_CS                 (2)       | // must correspond to
NAND CS used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)|
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (0);        // NOTE: this field
contains
                                                                 //  the total amount
                                    // DMA transferred to GPMI via DMA (0)!
write[1].gpmi_compare = NULL;                            // field not used but necessary
to
set eccctrl
write[1].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD,    ENCODE_8_BIT) | // specify t = 8
mode
                        BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       | // enable ECC module
                        BF_GPMI_ECCCTRL_BUFFER_MASK   (0x1FF);           // write all 8 data
blocks
                                                                         //   and 1 aux block
write[1].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(4096+218);       // specify number of bytes
                                                                 //   written to NAND
write[1].gpmi_data_pointer = &write_payload_pointer;             // data buffer address
write[1].gpmi_aux_pointer  = &write_aux_pointer;                 // metadata pointer
//-------------------------------------------------------------------------
// Descriptor 3: issue NAND write execute command (CLE)
//-------------------------------------------------------------------------
write[2].dma_nxtcmdar = &write[3];                         // point to the next descriptor
write[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1) |     // 1 byte command
                   BF_APBH_CHn_CMD_CMDWORDS      (3) |     // send 3 words to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (1) |     // wait for command to finish
```

```
before
                                                        //    continuing
                    BF_APBH_CHn_CMD_SEMAPHORE     (0) |
                    BF_APBH_CHn_CMD_NANDWAIT4READY(0) |
                    BF_APBH_CHn_CMD_NANDLOCK      (1) |     // maintain resource lock
                    BF_APBH_CHn_CMD_IRQONCMPLT    (0) |
                    BF_APBH_CHn_CMD_CHAIN         (1) |     // follow chain to next command
                    BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);   // read data from DMA, write
to NAND
write[2].dma_bar = &nand_cmd_addr_buffer[6];             // point to byte 6, write execute
command
// 3 words sent to the GPMI
write[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)     | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)   |
                      BF_GPMI_CTRL0_CS                 (2)        | // must correspond to
NAND CS used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);        // 1 byte command
write[2].gpmi_compare = NULL;                           // field not used but necessary to set
eccctrl
write[2].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 4: wait for ready (CLE)
//-------------------------------------------------------------------------
write[3].dma_nxtcmdar = &write[4];                        // point to the next descriptor
write[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0) |    // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (1) |    // send 1 word to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (1) |    // wait for command to finish before
                                                         //      continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0) |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(1) |    // wait for nand to be ready
                   BF_APBH_CHn_CMD_NANDLOCK      (0) |    // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0) |
                   BF_APBH_CHn_CMD_CHAIN         (1) |    // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);          // no dma transfer
write[3].dma_bar = NULL;                                                // field not used
// 1 word sent to the GPMI
write[3].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) | // wait for NAND
ready
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)         |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)      |
                      BF_GPMI_CTRL0_CS                 (2)            | // must correspond
to NAND CS used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)     |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)            |
                      BF_GPMI_CTRL0_XFER_COUNT         (0);
//-------------------------------------------------------------------------
// Descriptor 5: psense compare (time out check)
//-------------------------------------------------------------------------
write[4].dma_nxtcmdar = &write[5];                          // point to the next descriptor
write[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)        | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)        |
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                   BF_APBH_CHn_CMD_CHAIN         (1)        | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check
write[4].dma_bar = dma_error_handler;                 // if sense check fails, branch to error
handler
//-------------------------------------------------------------------------
// Descriptor 6: issue NAND status command (CLE)
//-------------------------------------------------------------------------
write[5].dma_nxtcmdar = &write[6];                          // point to the next descriptor
write[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1)      |  // 1 byte command
                   BF_APBH_CHn_CMD_CMDWORDS      (3)      |  // send 3 words to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)      |  // wait for command to finish
```

```
before
continuing
                    BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                    BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                    BF_APBH_CHn_CMD_NANDLOCK      (1)    |  // prevent other DMA channels
from
taking over
                    BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                    BF_APBH_CHn_CMD_CHAIN         (1)    |  // follow chain to next command
                    BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);  // read data from DMA, write to
NAND
write[5].dma_bar = &nand_cmd_addr_buffer[7];                 // point to byte 7, status
command
write[5].gpmi_compare = NULL;                          // field not used but necessary to set
eccctrl
write[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
// 3 words sent to the GPMI
write[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                      BF_GPMI_CTRL0_CS                 (2)       | // must correspond to
NAND CS used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);      // 1 byte command
//--------------------------------------------------------------------------
// Descriptor 7: read status and compare (DATA)
//--------------------------------------------------------------------------
write[6].dma_nxtcmdar = &write[7];                           // point to the next descriptor
write[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)  |    // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (2)  |    // send 2 words to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)  |    // wait for command to finish
before
                                                         //      continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)  |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)  |
                   BF_APBH_CHn_CMD_NANDLOCK      (1)  |    // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)  |
                   BF_APBH_CHn_CMD_CHAIN         (1)  |    // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
write[6].dma_bar = NULL;                                     // field not used
// 2 word sent to the GPMI
write[6].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ_AND_COMPARE) | // read from the
                                                                  //       NAND and
//                                                                // compare to
expect
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)  |
                      BF_GPMI_CTRL0_CS                 (2)        |    // must correspond to
NAND CS used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)        |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);
write[6].gpmi_compare = <MASK_AND_REFERENCE_VALUE>;  // NOTE: mask and reference values are
NAND
                                                   // SPECIFIC to evaluate the NAND status
//--------------------------------------------------------------------------
// Descriptor 8: psense compare (time out check)
//--------------------------------------------------------------------------
write[7].dma_nxtcmdar = &write[8];                             // point to the next descriptor
write[7].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)        | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)        | // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                   BF_APBH_CHn_CMD_CHAIN         (1)        | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check
write[7].dma_bar = dma_error_handler;             // if sense check fails, branch to error
```

```
handler
//-----------------------------------------------------------------------------
// Descriptor 9: emit GPMI interrupt
//-----------------------------------------------------------------------------
write[8].dma_nxtcmdar = NULL;                                   // not used since this is
last
descriptor
write[8].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT     (0)          | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS       (0)          | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD    (0)          | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE      (0)          |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)           |
                   BF_APBH_CHn_CMD_NANDLOCK       (0)          |
                   BF_APBH_CHn_CMD_IRQONCMPLT     (1)          | // emit GPMI interrupt
                   BF_APBH_CHn_CMD_CHAIN          (0)          | // terminate DMA chain
processing
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
```

14.5.1.2 Using the BCH Encoder

To use the BCH encoder, first turn off the module-wide soft reset bit in both the GPMI and BCH blocks before starting any DMA activity. Note that turning off the soft reset must take place by itself, prior to programming the rest of the control registers. Turn off the BCH bus master soft reset bit. Turn off the clock gate bits.

Program the remainder of the GPMI, BCH and APBH DMA as follows:

```
// bring APBH out of reset
HW_APBH_CTRL0_CLR(BM_APBH_CTRL0_SFRST);
HW_APBH_CTRL0_CLR(BM_APBH_CTRL0_CLKGATE);
// bring BCH out of reset
HW_BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
HW_BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);
// bring gpmi out of reset
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST);
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_CLKGATE);
HW_GPMI_CTRL1_SET(BM_GPMI_CTRL1_DEV_RESET | // deassert reset
             BM_GPMI_CTRL1_BCH_MODE  ); // enable BCH mode

// enable pinctrl
HW_PINCTRL_CTRL_WR(0x00000000);
// enable gpmi pins
HW_PINCTRL_MUXSEL0_CLR(0x0000ffff); // data bits
HW_PINCTRL_MUXSEL1_CLR(0x03ffffff); // control bits
HW_PINCTRL_MUXSEL8_CLR(0x0003f3ff); // control bits
HW_PINCTRL_MUXSEL8_SET(0x00015155); // control bits
```

Note that for writing NANDs (ECC encoding), only GPMI DMA command complete interrupts are used. The BCH engine is used for writing to the NAND but may optionally produces an interrupt. From the sample code in DMA Structure Code Example 1.

- DMA descriptor 1 prepares the NAND for data write by using the GPMI to issue a write setup command byte under "CLE", then sends a 5-byte address under "ALE". The BCH engine is disabled and not used for these commands.
- DMA descriptor 2 enables the BCH engine for encoding to begin the initial writing of the NAND data by specifying where the data and auxiliary payload are coming from in system memory.

- DMA descriptor 3 issues the write commit command byte under "CLE" to the NAND.
- DMA descriptor 4 waits for the NAND to complete the write commit/transfer by watching the NAND's ready line status. This descriptor relinquishes the NANDLOCK on the GPMI to enable the other DMA channels to initiate NAND transactions on different NAND CS lines.
- DMA descriptor 6 issues a NAND status command byte under "CLE" to check the status of the NAND device following the page write.
- DMA descriptor 7 reads back the NAND status and compares the status with an expected value. If there are differences, then the DMA processing engine follows an error-handling DMA descriptor path.
- DMA descriptor 8 disables the BCH engine and emits a GPMI interrupt to indicate that the NAND write has been completed.

14.5.2 BCH Decoding for NAND Reads

When a page is read from NAND flash, BCH syndromes will be computed and, if correctable errors are found, they will be corrected on a per block basis within the NAND page. This decoding process is fully overlapped with other NAND data reads and with ARM platform execution. The BCH decoder flowchart in Figure 14-9 shows the steps involved in programming the decoder. The hardware flow of reading and decoding a 4096-byte page is shown in Figure 14-10.

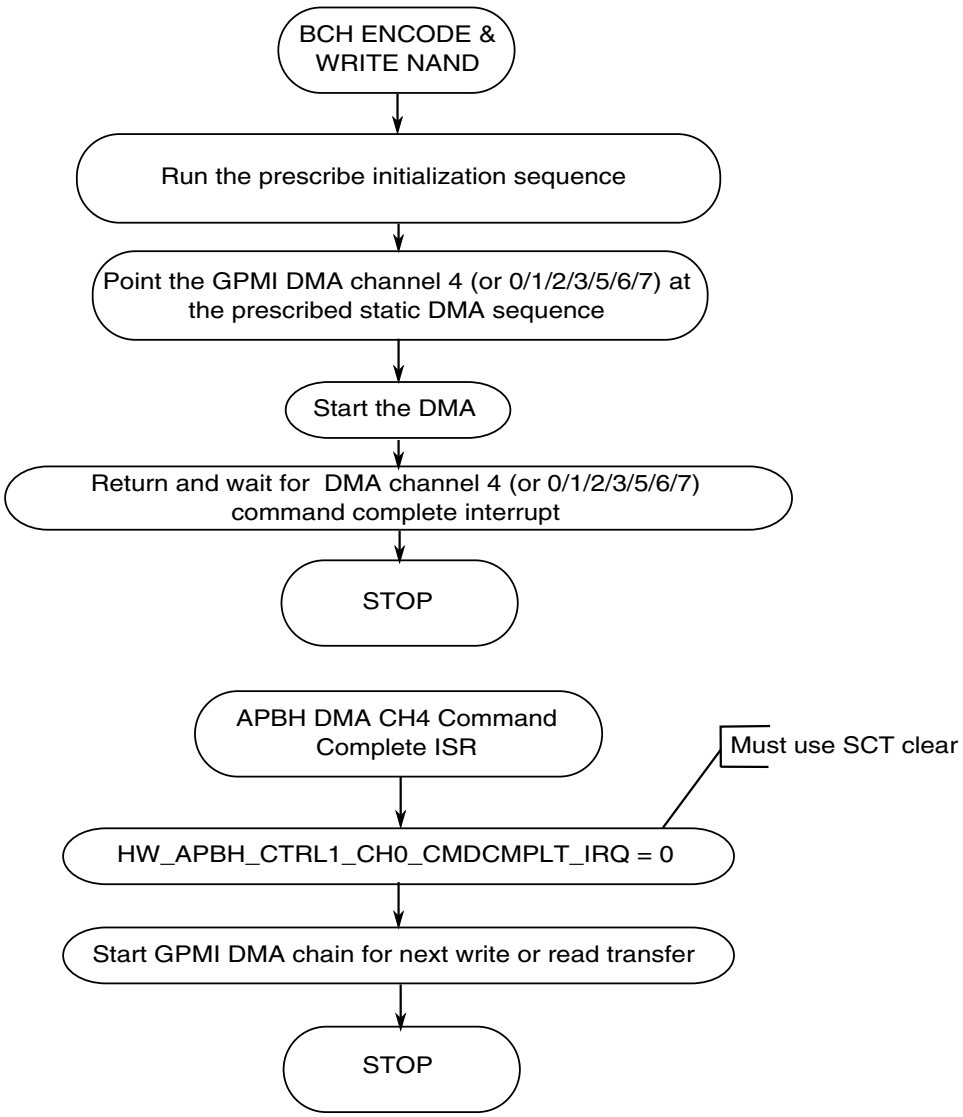


Figure 14-9. BCH Decode Flowchart

Conceptually, an APBH DMA Channel 0, 1, 2, 3, 4, 5, 6, or 7 command chain with seven command structures linked together is used to perform the BCH decode operation (as shown in Figure 14-10). NOTE: The GPMI's DMA command structures controls the BCH decode operation.

To use the BCH decoder with the GPMI's DMA, create a DMA command chain containing seven descriptor structures, as shown in Figure 14-10 and detailed in the DMA structure code example that follows it in DMA Structure Code Example 2. The seven DMA descriptors perform the following tasks:

1. Issue NAND read setup command byte (under "CLE") and address bytes (under "ALE").
2. Issue NAND read execute command byte (under "CLE").

3. Wait for the NAND device to complete accessing the block data by watching the ready signal.
4. Check for NAND timeout through "PSENSE".
5. Configure and enable the BCH block and read the NAND block data.
6. Disable the BCH block.
7. Descriptor NOP to allow NANDLOCK in the previous descriptor to the thread-safe.

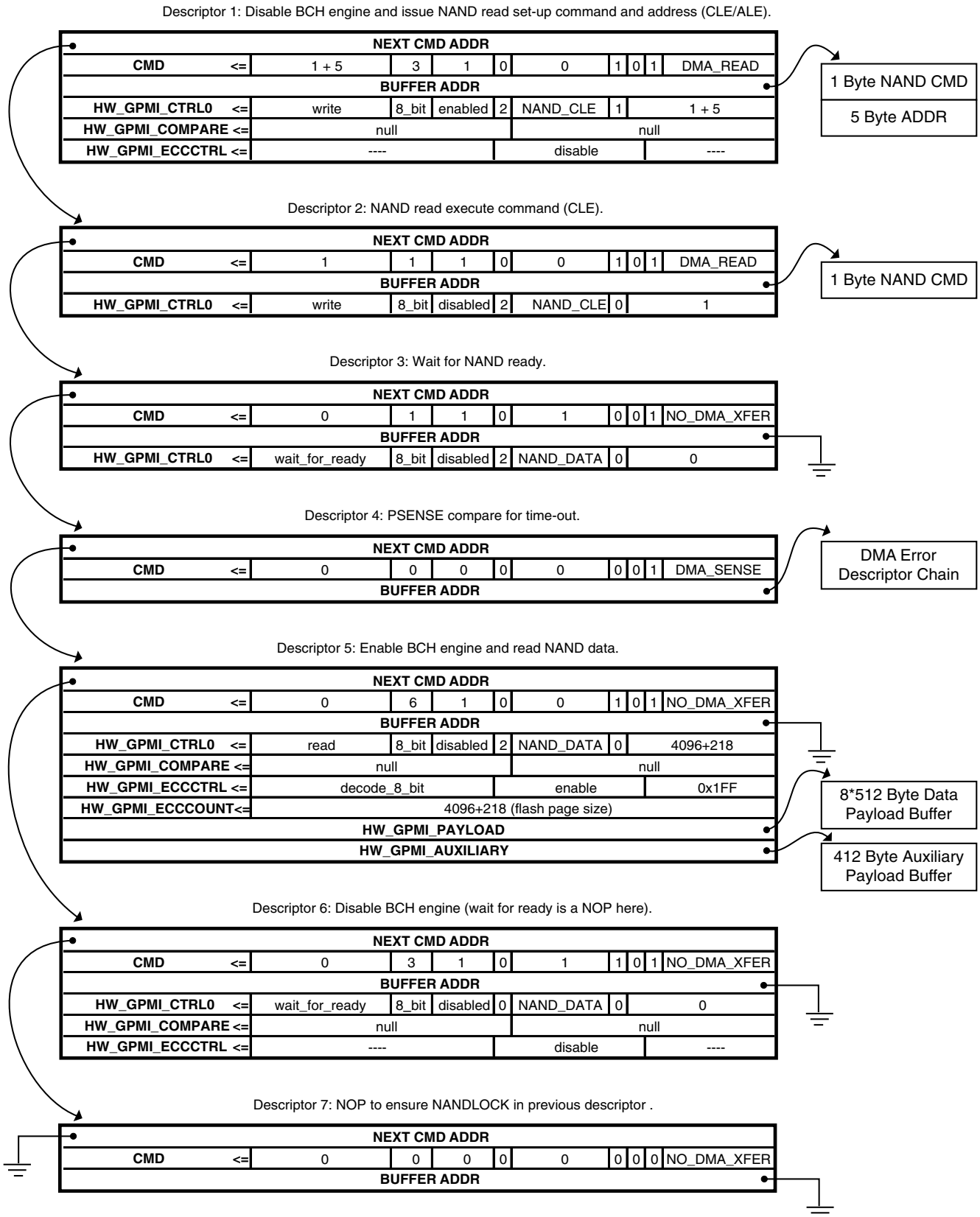


Figure 14-10. BCH Decode DMA Descriptor Chain

NOTE

To interpret the fields in this diagram, see Figure 14-7 for the descriptor legend.

14.5.2.1 DMA Structure Code Example 2

The following sample code illustrates the coding for one read transaction, consisting of a seven DMA command structure chain for reading all 4096 bytes of payload data (eight 512-byte blocks) and 65 bytes of metadata with the associative parity bytes (8 x (18) + 9) from a 4K NAND page sitting on GPMI CS2.

```
//-----------------------------------------------------------------------------
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//-----------------------------------------------------------------------------
typedef struct {
  // DMA related fields
  unsigned int dma_nxtcmdar;
  unsigned int dma_cmd;
  unsigned int dma_bar;
  // GPMI related fields
  unsigned int gpmi_ctrl0;
  unsigned int gpmi_compare;
  unsigned int gpmi_eccctrl;
  unsigned int gpmi_ecccount;
  unsigned int gpmi_data_ptr;
  unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;
//-----------------------------------------------------------------------------
// allocate 7 descriptors for doing a NAND ECC Read
//-----------------------------------------------------------------------------
GENERIC_DESCRIPTOR read[7];
//-----------------------------------------------------------------------------
// DMA descriptor pointer to handle error conditions from psense checks
//-----------------------------------------------------------------------------
unsigned int * dma_error_handler;
//-----------------------------------------------------------------------------
// 7 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
//   byte 0 is read setup command
//   bytes 1-5 is the NAND address
//   byte 6 is read execute command
//-----------------------------------------------------------------------------
unsigned char nand_cmd_addr_buffer[7];
//-----------------------------------------------------------------------------
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int read_payload_buffer[(4096/4)];
//-----------------------------------------------------------------------------
// 412 byte auxiliary buffer used for reads
// needs to be word aligned
//-----------------------------------------------------------------------------
unsigned int read_aux_buffer[(412/4)];
//-----------------------------------------------------------------------------
// Descriptor 1: issue NAND read setup command (CLE/ALE)
//-----------------------------------------------------------------------------
read[0].dma_nxtcmdar = &read[1];                          // point to the next descriptor
read[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1 + 5) |  // 1 byte command, 5 byte address
                  BF_APBH_CHn_CMD_CMDWORDS      (3)     |  // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)     |  // wait for command to finish
                                                           //     before continuing
```

```
                    BF_APBH_CHn_CMD_SEMAPHORE     (0)      |
                    BF_APBH_CHn_CMD_NANDWAIT4READY(0)      |
                    BF_APBH_CHn_CMD_NANDLOCK      (1)      |  // prevent other DMA channels from
                                                           //    taking over
                    BF_APBH_CHn_CMD_IRQONCMPLT    (0)      |
                    BF_APBH_CHn_CMD_CHAIN         (1)      |   // follow chain to next command
                    BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);  // read data from DMA, write to
NAND
read[0].dma_bar = &nand_cmd_addr_buffer;            // byte 0 read setup, bytes 1 - 5 NAND
address
// 3 words sent to the GPMI
read[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)     | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)   |
                     BF_GPMI_CTRL0_CS                 (2)        | // must correspond to NAND
CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (1)        | // send command and address
                     BF_GPMI_CTRL0_XFER_COUNT         (1 + 5);    // 1 byte command, 5 byte
address
read[0].gpmi_compare = NULL;                        // field not used but necessary to set
eccctrl
read[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-------------------------------------------------------------------------
// Descriptor 2: issue NAND read execute command (CLE)
//-------------------------------------------------------------------------
read[1].dma_nxtcmdar = &read[2];                          // point to the next descriptor
read[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1)    |   // 1 byte read command
                  BF_APBH_CHn_CMD_CMDWORDS      (1)    |   // send 1 word to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |   // wait for command to finish
before
                                                        //    continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)    |   // prevent other DMA channels from
                                                        //     taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                  BF_APBH_CHn_CMD_CHAIN         (1)    |   // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);     // read data from DMA, write
to NAND
read[1].dma_bar = &nand_cmd_addr_buffer[6];              // point to byte 6, read execute
command
// 1 word sent to the GPMI
read[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)     | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)  |
                     BF_GPMI_CTRL0_CS                 (2)        | // must correspond to NAND
CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)        |
                     BF_GPMI_CTRL0_XFER_COUNT         (1);        // 1 byte command
//-------------------------------------------------------------------------
// Descriptor 3: wait for ready (DATA)
//-------------------------------------------------------------------------
read[2].dma_nxtcmdar = &read[3];                          // point to the next descriptor
read[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)      | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (1)      | // send 1 word to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)      | // wait for command to finish
before
                                                        //     continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)      |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(1)      | // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK      (0)      | // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)      |
                  BF_APBH_CHn_CMD_CHAIN         (1)      | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);   // no dma transfer
read[2].dma_bar = NULL;                                   // field not used
// 1 word sent to the GPMI
read[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) | // wait for NAND
ready
```

```
                    BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)          |
                    BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)       |
                    BF_GPMI_CTRL0_CS                 (2)             | // must correspond
to NAND CS used
                    BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)      |
                    BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)             |
                    BF_GPMI_CTRL0_XFER_COUNT         (0);
//--------------------------------------------------------------------------
// Descriptor 4: psense compare (time out check)
//--------------------------------------------------------------------------
read[3].dma_nxtcmdar = &read[4];                                    // point to the next
descriptor
read[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)         |       // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (0)         |       // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)         |       // do not wait to continue
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)         |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)         |
                  BF_APBH_CHn_CMD_NANDLOCK      (0)         |
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)         |
                  BF_APBH_CHn_CMD_CHAIN         (1)         |       // follow chain to next
command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);         // perform a sense check
read[3].dma_bar = dma_error_handler;                        // if sense check fails, branch to
error handler
//--------------------------------------------------------------------------
// Descriptor 5: read 4K page plus 65 byte meta-data Nand data
//               and send it to ECC block (DATA)
//--------------------------------------------------------------------------
read[4].dma_nxtcmdar = &read[5];                              // point to the next descriptor
read[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)     | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (6)     | // send 6 words to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)     | // wait for command to finish before
                                                          //       continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)     |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)     |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)     | // prevent other DMA channels from
taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)     | // ECC block generates BCH interrupt
                                                          //      on completion
                  BF_APBH_CHn_CMD_CHAIN         (1)     | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);     // no DMA transfer,
                                                                  // ECC block handles
transfer
read[4].dma_bar = NULL;                                           // field not used
// 6 words sent to the GPMI
read[4].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)       |  // read from the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)      |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)   |
                     BF_GPMI_CTRL0_CS                 (2)         |  // must correspond to
NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)         |
                     BF_GPMI_CTRL0_XFER_COUNT         (4096+218);    // eight 512 byte data
blocks
                                                                     // metadata, and parity

read[4].gpmi_compare = NULL;                                  // field not used but necessary to set
eccctrl
// GPMI ECCCTRL PIO This launches the 4K byte transfer through BCH's
// bus master. Setting the ECC_ENABLE bit redirects the data flow
// within the GPMI so that read data flows to the BCH engine instead
// of flowing to the GPMI's DMA channel.
read[4].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD,    DECODE_8_BIT) |    // specify t = 8
mode
                       BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       |    // enable ECC
module
                       BF_GPMI_ECCCTRL_BUFFER_MASK     (0X1FF);  // read all 8 data blocks
and
1 aux block
read[4].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(4096+218);        // specify number of bytes
```

```
                                                                    //        read from NAND
read[4].gpmi_data_ptr = &read_payload_buffer;                       // pointer for the 4K byte
                                                                    //    data area
read[4].gpmi_aux_ptr = &read_aux_buffer;                            // pointer for the 65 byte
aux area +
                                                                    // parity and syndrome
bytes for both
                                                                   // data and aux blocks.
//--------------------------------------------------------------------------
// Descriptor 6: disable ECC block
//--------------------------------------------------------------------------
read[5].dma_nxtcmdar = &read[6];                                    // point to the next descriptor
read[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)      | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (3)      | // send 3 words to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)      | // wait for command to finish
before
                                                           //     continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)      |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(1)      | // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK      (1)      | // need nand lock to be
                                                           // thread safe while turn-off BCH
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)      |
                  BF_APBH_CHn_CMD_CHAIN         (1)      | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);      // no dma transfer
read[5].dma_bar = NULL;                                             // field not used
// 3 words sent to the GPMI
read[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)       |
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)      |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)   |
                     BF_GPMI_CTRL0_CS                 (2)         | // must correspond to
NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)         |
                     BF_GPMI_CTRL0_XFER_COUNT         (0);
read[5].gpmi_compare = NULL;                                       // field not used but necessary to set
eccctrl
read[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE);  // disable the ECC block
//--------------------------------------------------------------------------
// Descriptor 7: deassert nand lock
//--------------------------------------------------------------------------
read[6].dma_nxtcmdar = NULL;                                        // not used since this is last
descriptor
read[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)    |    // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (0)    |    // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)    |    // wait for command to finish
before
                                                           //        continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                  BF_APBH_CHn_CMD_NANDLOCK      (0)    |    // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |    // BCH engine generates interrupt
                  BF_APBH_CHn_CMD_CHAIN         (0)    |    // terminate DMA chain processing
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);      // no dma transfer
read[6].dma_bar = NULL;                                             // field not used
```

14.5.2.2 Using the Decoder

As illustrated in Figure 14-10 and the sample code in DMA Structure Code Example 2 :

- DMA descriptor 1 prepares the NAND for data read by using the GPMI to issue a NAND read setup command byte under "CLE", then sends a 5-byte address under "ALE". The BCH engine is not used for these commands.

- DMA descriptor 2 issues a one-byte read execute command to the NAND device that triggers its read access. The NAND then goes not ready.
- DMA descriptor 3 performs a wait for ready operation allowing the DMA chain to remain dormant until the NAND device completes its read access time.
- DMA descriptor 5 handles the reading and error correction of the NAND data. This command's PIOs activate the BCH engine to write the read NAND data to system memory and to process it for any errors that need to be corrected. This DMA descriptor contains two PIO values that are system memory addresses pointing to the PAYLOAD data area and to the AUXILIARY data area. These addresses are used by the BCH engine's bus master to move data into system memory and to correct it. While this example is reading an entire 4K page-payload plus metadata-it is equally possible to read just one 512-byte payload block or just the uniquely protected metadata block in a single 7 DMA structure transfer.
- DMA descriptor 6 disables the BCH engine with the NANDLOCK asserted. This is necessary to ensure that the GPMI resource is not arbitrated to another DMA channel when multiple DMA channels are active concurrently.
- DMA descriptor 7 deasserts the NANDLOCK to free up the GPMI resource to another channel.

As the BCH block receives data from the GPMI:

- The decoder transforms the read NAND data block into an BCH code word and computes the codeword syndrome.
- If no errors are present, then the BCH block can immediately report back to firmware. This report is passed as the BCH_CTRL_COMPLETE_IRQ interrupt status bit and the associated status registers in BCH_STATUS0/1 registers.
- If an error is present, then the BCH block corrects the necessary data block or parity block bytes, if possible (not all errors are correctable).

As the BCH decoder reads the data and parity blocks, it records a special condition, that is, that all of the bits of a payload data block or metadata block are one, including any associated parity bytes. The "all-ones" case for both parity and data indicates an erased block in the NAND device.

The BCH_STATUS0 register contains a 4-bit field that indicates the final status of the auxiliary block. A value of 0x0 indicates no errors found for a block.

- A value of 1 to 20 inclusive indicates that many correctable errors were found and fixed.
- A value of 0xFE indicates uncorrectable errors detected on the block.
- A value of 0xFF indicates that the block was in the special ALL ONES state and is therefore considered to be an ERASED block.
- All other values are disallowed by the hardware design.

Recall that up to eight NAND devices can have DMA chains in-flight at once, that is, they can all be contending for access to the GPMI data bus. It is impossible to predict which NAND device will enter the BCH engine with a transfer first, because each chain includes a wait4ready command structure. As a result, firmware should look at the BCH_STATUS0_COMPLETED_CE bit field to determine which block is being reported in the status register. There is also a 16 bit HANDLE field in the GPMI_ECCCTRL register that is passed down the pipeline with each transaction. This handle field can be used to speed firmware's detection of which transaction is being reported.

These examples of reading and writing have focused on full page transfers of 4 K page NAND devices. Other device configurations can be specified by changing the ECCOUNT field in the GPMI registers and reprogramming the BCH's FLASHnLAYOUTm registers.

The BCH and GPMI blocks are designed to be very efficient at reading single 512(or 1024)-byte pages in one transaction. With no errors, the transaction takes less than 20 HCLKs longer than the time to read the raw data from the NAND.

To summarize, the APBH DMA command chain for a BCH decode operation is shown in Figure 14-10. Seven DMA command structures must be present for each NAND read transaction decoded by the BCH. The seven DMA command structures for multiple NAND read transaction blocks can be chained together to make larger units of work for the BCH, and each will produce an appropriate error report in the BCH PIO space. Multiple NAND devices can have such multiple chains scheduled. The results can come back out of order with respect to the multiple chains.

14.5.3 Interrupts

There are two interrupt sources used in processing BCH protected NAND read and write transfers. Since all BCH operations are initiated by GPMI DMA command structures, the DMA completion interrupt for the GPMI is an important ISR. Both of the flow charts of the Programmable Registers section and Programming the BCH/GPMI Interfaces show the GPMI DMA complete ISR skeleton. In both reads and writes, the GPMI DMA completion interrupt is used to schedule work *INTO* the error correction pipeline. As the front end processing completes, the DMA interrupt is generated and additional work, i.e. DMA chains, are passed to the GPMI DMA to keep it "*fed*". For write operations, this is the only interrupt that will be generated for processing the NAND write transfer.

For reads, however, two interrupts are needed. Every read is started by a GPMI DMA command chain and the front end queue is fed as described above. The back end of the read pipeline is "drained" by monitoring the BCH completion interrupt found int BCH_CTRL_COMPLETE_IRQ.

An BCH transaction consists of reading or writing all of the blocks requested in the GPMI_ECCCTRL_BUFFER_MASK bit field. As every read transaction completes, it posts the status of all of the blocks to the BCH_STATUS0 register and sets the completion interrupt. The five stages of the BCH read pipeline completes, one in the GPMI and four in the BCH, are independently stalled as they complete and try to deliver to the next stage in the data flow. Several of these stages can be skipped if no-errors are found or once an uncorrectable error is found in a block.

In any case, the final stage will stall if the status register is busy waiting for the ARM platform to take status register results. The hardware monitors the state of the BCH_CTRL_COMPLETE_IRQ bit. If it is still set when the last pipeline stage is ready to post data, then the stage will stall. It follows that the next previous stage will stall when it is ready to hand off work to the final stage, and so on up the pipeline.

WARNING: It is important that firmware read the STATUS0/1 results and save them before clearing the interrupt request bit otherwise a transaction and its results could be completely lost.

14.6 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. The examplary code is shown below:

```
// A soft reset can take multiple clocks to complete, so do NOT gate the
// clock when setting soft reset. The reset process will gate the clock
// automatically. Poll until this has happened before subsequently
// preparing soft-reset and clock gate
  HW_BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
  HW_BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);
// asserting soft-reset
  HW_BCH_CTRL_SET(BM_BCH_CTRL_SFTRST);
// waiting for confirmation of soft-reset
  while (!HW_BCH_CTRL.B.CLKGATE)
  {
// busy wait
  }
// Done.
  HW_BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
  HW_BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);
```

14.7 Programmable Registers

BCH Hardware Register Format Summary

BCH memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_8000	Hardware BCH ECC Accelerator Control Register (BCH_CTRL)	32	R/W	C000_0000h	14.7.1/ 715
4100_8004	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_SET)	32	R/W	C000_0000h	14.7.1/ 715
4100_8008	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_CLR)	32	R/W	C000_0000h	14.7.1/ 715
4100_800C	Hardware BCH ECC Accelerator Control Register (BCH_CTRL_TOG)	32	R/W	C000_0000h	14.7.1/ 715
4100_8010	Hardware ECC Accelerator Status Register 0 (BCH_STATUS0)	32	R	0000_0010h	14.7.2/ 717
4100_8020	Hardware ECC Accelerator Mode Register (BCH_MODE)	32	R/W	0000_0000h	14.7.3/ 719
4100_8030	Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR)	32	R/W	0000_0000h	14.7.4/ 719
4100_8040	Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR)	32	R/W	0000_0000h	14.7.5/ 720
4100_8050	Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR)	32	R/W	0000_0000h	14.7.6/ 720
4100_8070	Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT)	32	R/W	E4E4_ E4E4h	14.7.7/ 721
4100_8080	Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0)	32	R/W	070A_4080h	14.7.8/ 722
4100_8090	Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1)	32	R/W	10DA_4080h	14.7.9/ 724
4100_80A0	Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0)	32	R/W	070A_4080h	14.7.10/ 725
4100_80B0	Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1)	32	R/W	10DA_4080h	14.7.11/ 727
4100_80C0	Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0)	32	R/W	070A_4080h	14.7.12/ 728
4100_80D0	Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1)	32	R/W	10DA_4080h	14.7.13/ 730
4100_80E0	Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0)	32	R/W	070A_4080h	14.7.14/ 731
4100_80F0	Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1)	32	R/W	10DA_4080h	14.7.15/ 733
4100_8100	Hardware BCH ECC Debug Register0 (BCH_DEBUG0)	32	R/W	0000_0000h	14.7.16/ 735
4100_8104	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_SET)	32	R/W	0000_0000h	14.7.16/ 735

Table continues on the next page...

BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_8108	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_CLR)	32	R/W	0000_0000h	14.7.16/ 735
4100_810C	Hardware BCH ECC Debug Register0 (BCH_DEBUG0_TOG)	32	R/W	0000_0000h	14.7.16/ 735
4100_8110	KES Debug Read Register (BCH_DBGKESREAD)	32	R	0000_0000h	14.7.17/ 736
4100_8120	Chien Search Debug Read Register (BCH_DBGCSFEREAD)	32	R	0000_0000h	14.7.18/ 737
4100_8130	Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD)	32	R	0000_0000h	14.7.19/ 737
4100_8140	Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD)	32	R	0000_0000h	14.7.20/ 738
4100_8150	Block Name Register (BCH_BLOCKNAME)	32	R	2048_4342h	14.7.21/ 738
4100_8160	BCH Version Register (BCH_VERSION)	32	R	0100_0000h	14.7.22/ 739

14.7.1 Hardware BCH ECC Accelerator Control Register (BCH_CTRL*n*)

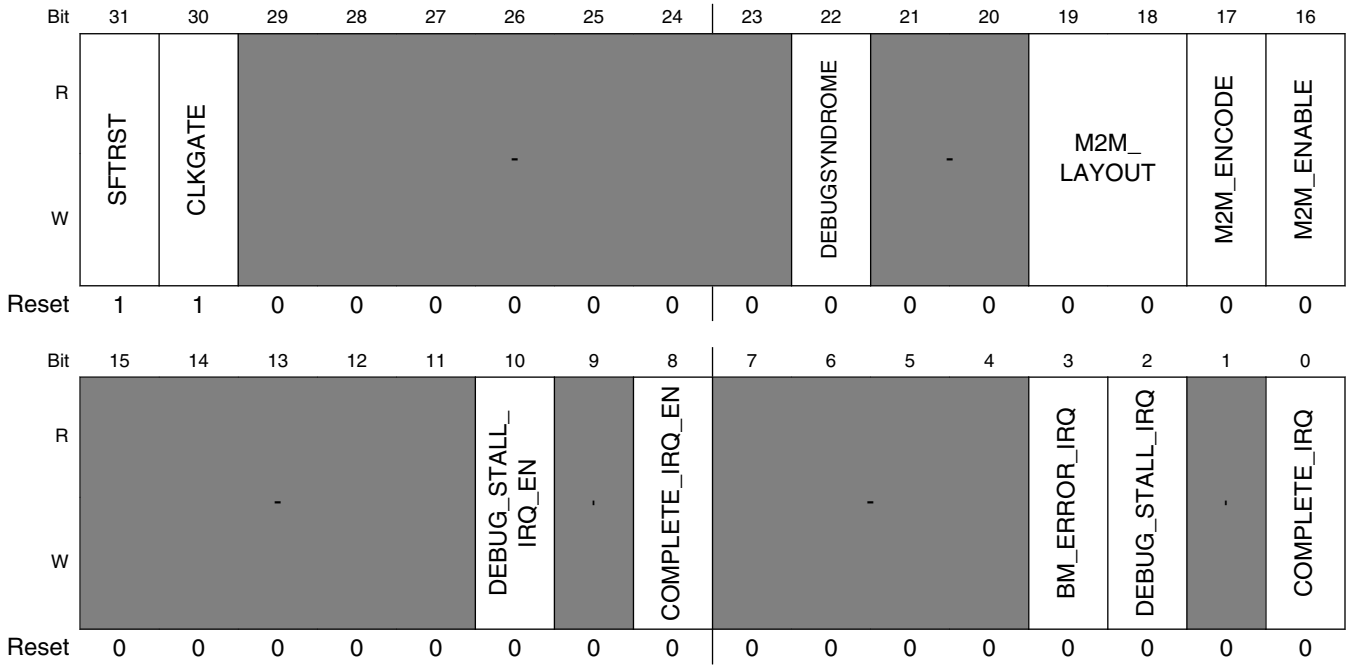
The BCH CTRL provides overall control of the hardware ECC accelerator

Addresses: BCH_CTRL is 4100_8000h base + 0h offset = 4100_8000h

BCH_CTRL_SET is 4100_8000h base + 4h offset = 4100_8004h

BCH_CTRL_CLR is 4100_8000h base + 8h offset = 4100_8008h

BCH_CTRL_TOG is 4100_8000h base + Ch offset = 4100_800Ch



BCH_CTRL*n* field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal BCH operation. Set this bit to one (default) to disable clocking with the BCH and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the BCH block to its default state. This bit resets all state machines except for the AHB master state machine 0x0 **RUN** — Allow BCH to operate normally. 0x1 **RESET** — Hold BCH in reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. 0x0 **RUN** — Allow BCH to operate normally. 0x1 **NO_CLKS** — Do not clock BCH gates in order to minimize power consumption.
29–23 -	Reserved, always set this bit to zero.
22 DEBUGSYNDROME	(For debug purposes only). Enable write of computed syndromes to memory on BCH decode operations. Computed syndromes will be written to the auxiliary buffer after the status block. Syndromes will be written as padded 16-bit values.
21–20 -	Reserved, always set these bits to zero.
19–18 M2M_LAYOUT	Selects the flash page format for memory-to-memory operations.
17 M2M_ENCODE	Selects encode (parity generation) or decode (correction) mode for memory-to-memory operations.

Table continues on the next page...

BCH_CTRL*n* field descriptions (continued)

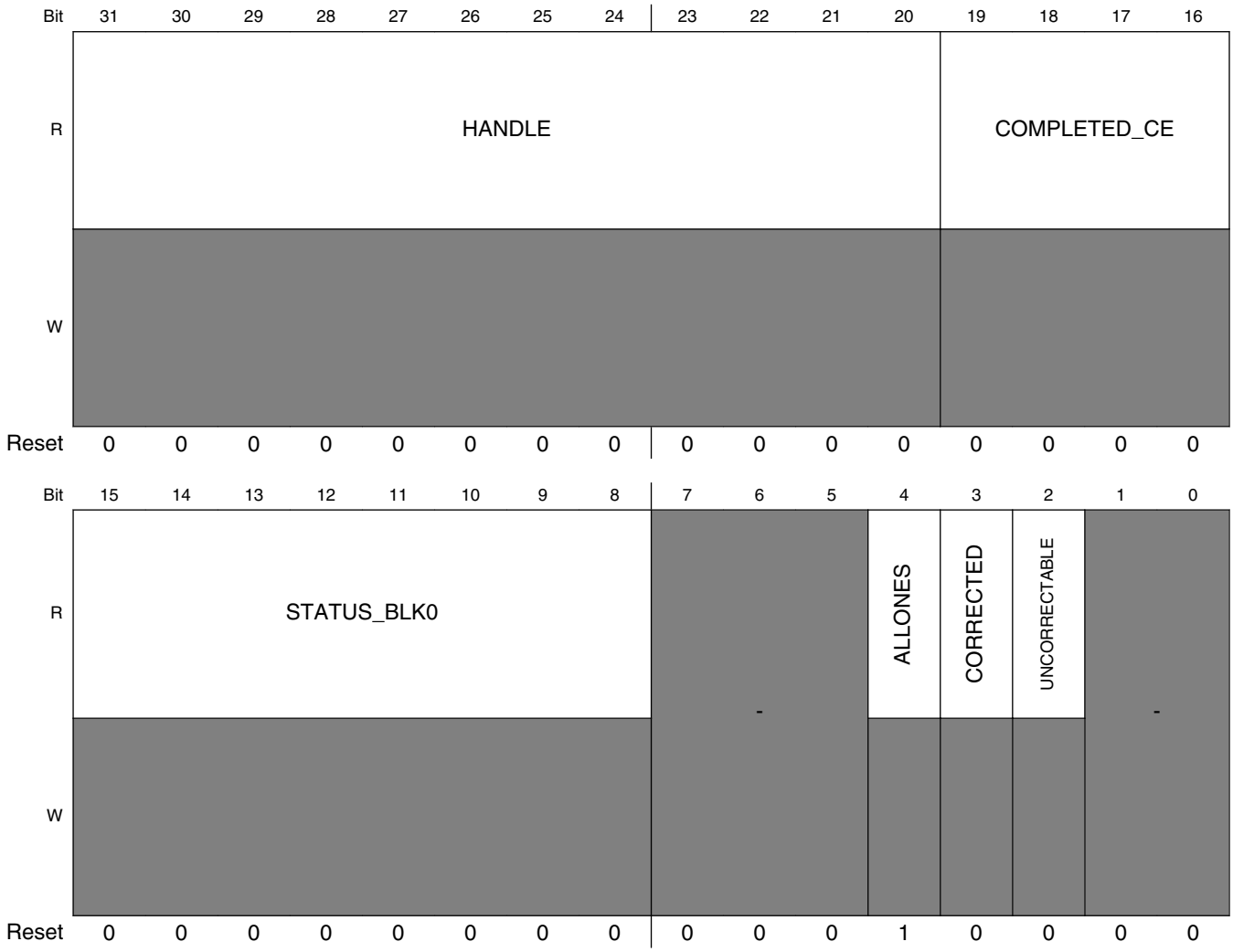
Field	Description
16 M2M_ENABLE	WARNING! WRITING THIS BIT INITIATES A MEMORY-TO-MEMORY OPERATION. The BCH module must be inactive (not processing data from the GPMI) when this bit is set. The M2M_ENCODE and M2M_LAYOUT bits as well as the ENCODEPTR, DATAPTR, and METAPTR registers are used for memory-to-memory operations and must be correctly programmed before writing this bit.
15–11 -	Reserved, always set these bits to zero.
10 DEBUG_STALL_IRQ_EN	1 = interrupt on debug stall mode is enabled. The irq is raised on every block
9 -	Reserved, always set these bits to zero.
8 COMPLETE_IRQ_EN	1 = interrupt on completion of correction is enabled.
7–4 -	Reserved, always set these bits to zero.
3 BM_ERROR_IRQ	AHB Bus interface Error Interrupt Status. Write a one to the SCT clear address to clear the interrupt status bit.
2 DEBUG_STALL_IRQ	DEBUG STALL Interrupt Status. Write a one to the SCT clear address to clear the interrupt status bit.
1 -	Reserved, always set these bits to zero.
0 COMPLETE_IRQ	This bit indicates the state of the external interrupt line. Write a one to the SCT clear address to clear the interrupt status bit. NOTE: subsequent ECC completions will be held off as long as this bit is set. Be sure to read the data from BCH_STATUS0,1 before clearing this interrupt bit.

14.7.2 Hardware ECC Accelerator Status Register 0 (BCH_STATUS0)

The BCH STAT provides overall status of the hardware ECC accelerator

The BCH STAT register provides visibility into the run-time status of the BCH and status information when processing is complete.

Address: BCH_STATUS0 is 4100_8000h base + 10h offset = 4100_8010h



BCH_STATUS0 field descriptions

Field	Description
31–20 HANDLE	Software supplies a 12 bit handle for this transfer as part of the GPMI DMA PIO operation that started the transaction. That handle passes down the pipeline and ends up here at the time the BCH interrupt is signaled.
19–16 COMPLETED_CE	This is the chip enable number corresponding to the NAND device from which this data came.
15–8 STATUS_BLK0	Count of symbols in error during processing of first block of flash (metadata block). The number of errors reported will be in the range of 0 to the ECC correction level for block 0. 0x00 **ZERO** — No errors found on block. 0x01 **ERROR1** — One error found on block. 0x02 **ERROR2** — One errors found on block. 0x03 **ERROR3** — One errors found on block. 0x04 **ERROR4** — One errors found on block. 0xFE **UNCORRECTABLE** — Block exhibited uncorrectable errors. 0xFF **ERASED** — Page is erased.

Table continues on the next page...

BCH_STATUS0 field descriptions (continued)

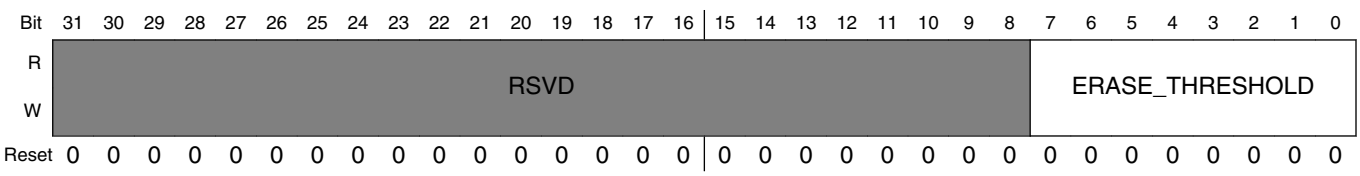
Field	Description
7–5 -	Reserved, always set these bits to zero.
4 ALLONES	1 = All data bits of this transaction are ONE.
3 CORRECTED	1 = At least one correctable error encountered during last processing cycle.
2 UNCORRECTABLE	1 = Uncorrectable error encountered during last processing cycle.
1–0 -	Reserved, always set these bits to zero.

14.7.3 Hardware ECC Accelerator Mode Register (BCH_MODE)

The BCH MODE register provides additional mode controls.

Contains additional global mode controls for the BCH engine.

Address: BCH_MODE is 4100_8000h base + 20h offset = 4100_8020h



BCH_MODE field descriptions

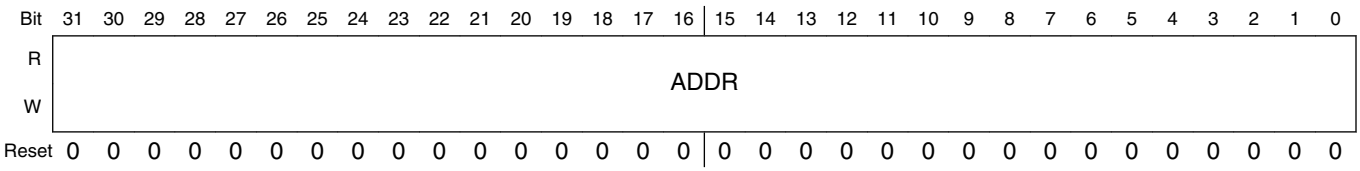
Field	Description
31–8 RSVD	Reserved, always set these bits to zero.
7–0 ERASE_ THRESHOLD	This value indicates the maximum number of zero bits on a flash page for it to be considered erased. For SLC NAND devices, this value should be programmed to 0 (meaning that the entire page should consist of bytes of 0xFF. For MLC NAND devices, bit errors may occur on reads (even on blank pages), so this threshold can be used to tune the erased page checking algorithm.

14.7.4 Hardware BCH ECC Loopback Encode Buffer Register (BCH_ENCODEPTR)

When performing memory to memory operations, indicates the address of the encode buffer. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register.

For memory to memory operations, this register is used as the pointer to the encoded data, which is an output when encoding and an input while decoding.

Address: BCH_ENCODEPTR is 4100_8000h base + 30h offset = 4100_8030h



BCH_ENCODEPTR field descriptions

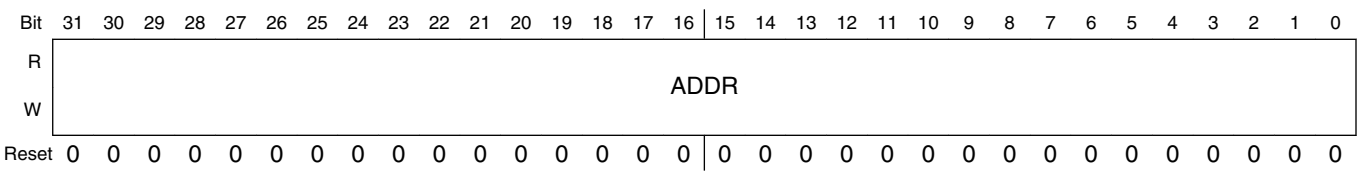
Field	Description
31–0 ADDR	Address pointer to encode buffer. This is the source for decode operations and the destination for encode operations. This value must be aligned on a 4 byte boundary.

14.7.5 Hardware BCH ECC Loopback Data Buffer Register (BCH_DATAPTR)

When performing memory to memory operations, indicates the address of the data buffer.

For memory to memory operations, this register is used as the pointer to the data to encode or the destination buffer for decode operations.

Address: BCH_DATAPTR is 4100_8000h base + 40h offset = 4100_8040h



BCH_DATAPTR field descriptions

Field	Description
31–0 ADDR	Address pointer to data buffer. This is the source for encode operations and the destination for decode operations. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register. This value must be aligned on a 4 byte boundary.

14.7.6 Hardware BCH ECC Loopback Metadata Buffer Register (BCH_METAPTR)

When performing memory to memory operations, indicates the address of the metadata buffer.

For memory to memory operations, this register is used as the pointer to the metadata to encode or the extracted metadata for decode operations.

Address: BCH_METAPTR is 4100_8000h base + 50h offset = 4100_8050h

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

BCH_METAPTR field descriptions

Field	Description
31–0 ADDR	Address pointer to metadata buffer. This is the source for encode metadata read operations and the destination for metadata decode operations. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register. This value must be aligned on a 4 byte boundary.

14.7.7 Hardware ECC Accelerator Layout Select Register (BCH_LAYOUTSELECT)

The BCH LAYOUTSELECT register provides a mapping of chip selects to layout registers.

When the BCH engine receives a request to process a data block from the GPMI interface, it will use this register to map the incoming chip select to one of the four possible flash layout registers

Address: BCH_LAYOUTSELECT is 4100_8000h base + 70h offset = 4100_8070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CS15_SELECT		CS14_SELECT		CS13_SELECT		CS12_SELECT		CS11_SELECT		CS10_SELECT		CS9_SELECT		CS8_SELECT		CS7_SELECT		CS6_SELECT		CS5_SELECT		CS4_SELECT		CS3_SELECT		CS2_SELECT		CS1_SELECT		CS0_SELECT	
W																																
Reset	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0

BCH_LAYOUTSELECT field descriptions

Field	Description
31–30 CS15_SELECT	Selects which layout is used for chip select 15.
29–28 CS14_SELECT	Selects which layout is used for chip select 14.
27–26 CS13_SELECT	Selects which layout is used for chip select 13.

Table continues on the next page...

BCH_LAYOUTSELECT field descriptions (continued)

Field	Description
25–24 CS12_SELECT	Selects which layout is used for chip select 12.
23–22 CS11_SELECT	Selects which layout is used for chip select 11.
21–20 CS10_SELECT	Selects which layout is used for chip select 10.
19–18 CS9_SELECT	Selects which layout is used for chip select 9.
17–16 CS8_SELECT	Selects which layout is used for chip select 8.
15–14 CS7_SELECT	Selects which layout is used for chip select 7.
13–12 CS6_SELECT	Selects which layout is used for chip select 6.
11–10 CS5_SELECT	Selects which layout is used for chip select 5.
9–8 CS4_SELECT	Selects which layout is used for chip select 4.
7–6 CS3_SELECT	Selects which layout is used for chip select 3.
5–4 CS2_SELECT	Selects which layout is used for chip select 2.
3–2 CS1_SELECT	Selects which layout is used for chip select 1.
1–0 CS0_SELECT	Selects which layout is used for chip select 0.

14.7.8 Hardware BCH ECC Flash 0 Layout 0 Register (BCH_FLASH0LAYOUT0)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH0LAYOUT1 register to control the format for the devices selecting layout 0 in the LAYOUTSELECT register.

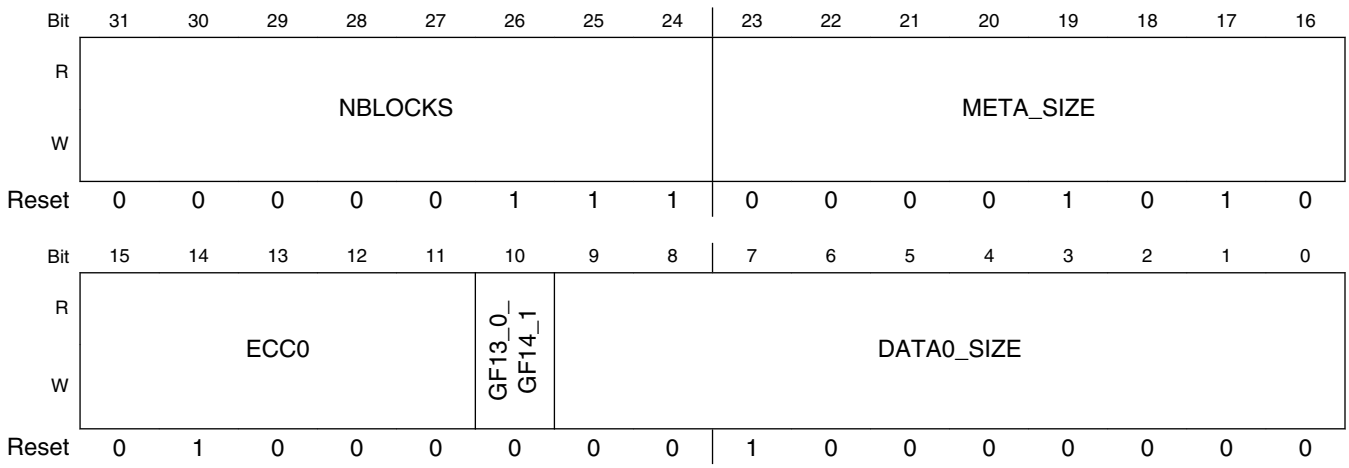
Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page

of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE

```
BCH_FLASH0LAYOUT0_WR(0x020C8000);
BCH_FLASH0LAYOUT1_WR(0x04408200);
```

Address: BCH_FLASH0LAYOUT0 is 4100_8000h base + 80h offset = 4100_8080h



BCH_FLASH0LAYOUT0 field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed

Table continues on the next page...

BCH_FLASH0LAYOUT0 field descriptions (continued)

Field	Description
	0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
9–0 DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS/four bytes) to be stored on the flash page. If set to zero, the first block will only contain metadata.

14.7.9 Hardware BCH ECC Flash 0 Layout 1 Register (BCH_FLASH0LAYOUT1)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH0LAYOUT0 register to control the format for the device selecting layout 0 in the LAYOUTSELECT register.

EXAMPLE

```
BCH_FLASH0LAYOUT0_WR(0x020C8000);
BCH_FLASH0LAYOUT1_WR(0x04408200);
```

Address: BCH_FLASH0LAYOUT1 is 4100_8000h base + 90h offset = 4100_8090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH0LAYOUT1 field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
9–0 DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS/four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

14.7.10 Hardware BCH ECC Flash 1 Layout 0 Register (BCH_FLASH1LAYOUT0)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH1LAYOUT1 register to control the format for the devices selecting layout 1 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page

of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE

```
BCH_FLASH1LAYOUT0_WR(0x020C8000);
BCH_FLASH1LAYOUT1_WR(0x04408200);
```

Address: BCH_FLASH1LAYOUT0 is 4100_8000h base + A0h offset = 4100_80A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NBLOCKS								META_SIZE							
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECC0					GF13_0_GF14_1	DATA0_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH1LAYOUT0 field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed

Table continues on the next page...

BCH_FLASH1LAYOUT0 field descriptions (continued)

Field	Description
	0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
9–0 DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS/four bytes) to be stored on the flash page. If set to zero, the first block will only contain metadata.

14.7.11 Hardware BCH ECC Flash 1 Layout 1 Register (BCH_FLASH1LAYOUT1)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH1LAYOUT0 register to control the format for the device selecting layout 1 in the LAYOUTSELECT register.

EXAMPLE

```
BCH_FLASH1LAYOUT0_WR(0x020C8000);
BCH_FLASH1LAYOUT1_WR(0x04408200);
```

Address: BCH_FLASH1LAYOUT1 is 4100_8000h base + B0h offset = 4100_80B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH1LAYOUT1 field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
9–0 DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS/four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

14.7.12 Hardware BCH ECC Flash 2 Layout 0 Register (BCH_FLASH2LAYOUT0)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH2LAYOUT1 register to control the format for the devices selecting layout 2 in the LAYOUTSELECT register.

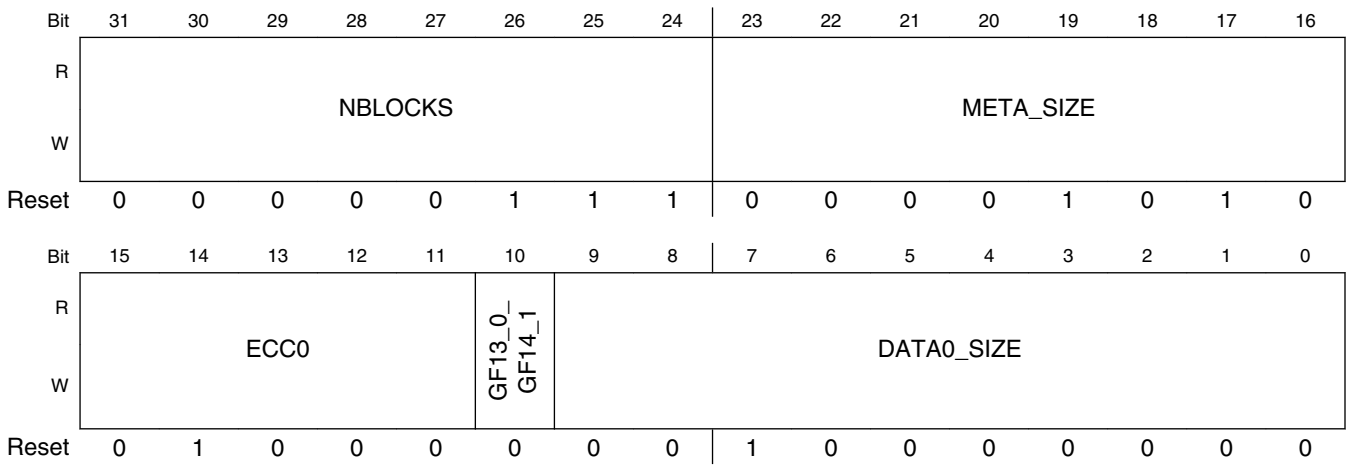
Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page

of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE

```
BCH_FLASH2LAYOUT0_WR(0x020C8000);
BCH_FLASH2LAYOUT1_WR(0x04408200);
```

Address: BCH_FLASH2LAYOUT0 is 4100_8000h base + C0h offset = 4100_80C0h



BCH_FLASH2LAYOUT0 field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed

Table continues on the next page...

BCH_FLASH2LAYOUT0 field descriptions (continued)

Field	Description
	0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
9–0 DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS/four bytes) to be stored on the flash page. If set to zero, the first block will only contain metadata.

14.7.13 Hardware BCH ECC Flash 2 Layout 1 Register (BCH_FLASH2LAYOUT1)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH2LAYOUT0 register to control the format for the device selecting layout 2 in the LAYOUTSELECT register.

EXAMPLE

```
BCH_FLASH2LAYOUT0_WR(0x020C8000);
BCH_FLASH2LAYOUT1_WR(0x04408200);
```

Address: BCH_FLASH2LAYOUT1 is 4100_8000h base + D0h offset = 4100_80D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH2LAYOUT1 field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
9–0 DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS/four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

14.7.14 Hardware BCH ECC Flash 3 Layout 0 Register (BCH_FLASH3LAYOUT0)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH3LAYOUT1 register to control the format for the devices selecting layout 3 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page

of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE

```
BCH_FLASH3LAYOUT0_WR(0x020C8000);
BCH_FLASH3LAYOUT1_WR(0x04408200);
```

Address: BCH_FLASH3LAYOUT0 is 4100_8000h base + E0h offset = 4100_80E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NBLOCKS								META_SIZE							
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECC0					GF13_0_GF14_1	DATA0_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH3LAYOUT0 field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–11 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed

Table continues on the next page...

BCH_FLASH3LAYOUT0 field descriptions (continued)

Field	Description
	0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
9–0 DATA0_SIZE	Indicates the size of the data 0 block (in DWORDS/four bytes) to be stored on the flash page. If set to zero, the first block will only contain metadata.

14.7.15 Hardware BCH ECC Flash 3 Layout 1 Register (BCH_FLASH3LAYOUT1)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH3LAYOUT0 register to control the format for the device selecting layout 3 in the LAYOUTSELECT register.

EXAMPLE

```
BCH_FLASH3LAYOUT0_WR(0x020C8000);
BCH_FLASH3LAYOUT1_WR(0x04408200);
```

Address: BCH_FLASH3LAYOUT1 is 4100_8000h base + F0h offset = 4100_80F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PAGE_SIZE															
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ECCN					GF13_0_GF14_1	DATAN_SIZE									
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

BCH_FLASH3LAYOUT1 field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–11 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed 0xB **ECC22** — ECC 22 to be performed 0xC **ECC24** — ECC 24 to be performed 0xD **ECC26** — ECC 26 to be performed 0xE **ECC28** — ECC 28 to be performed 0xF **ECC30** — ECC 30 to be performed 0x10 **ECC32** — ECC 32 to be performed
10 GF13_0_GF14_1	Select GF13 or GF14: 0-GF13; 1-GF14
9–0 DATAN_SIZE	Indicates the size of the subsequent data blocks (in DWORDS/four bytes) to be stored on the flash page. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

14.7.16 Hardware BCH ECC Debug Register0 (BCH_DEBUG0*n*)

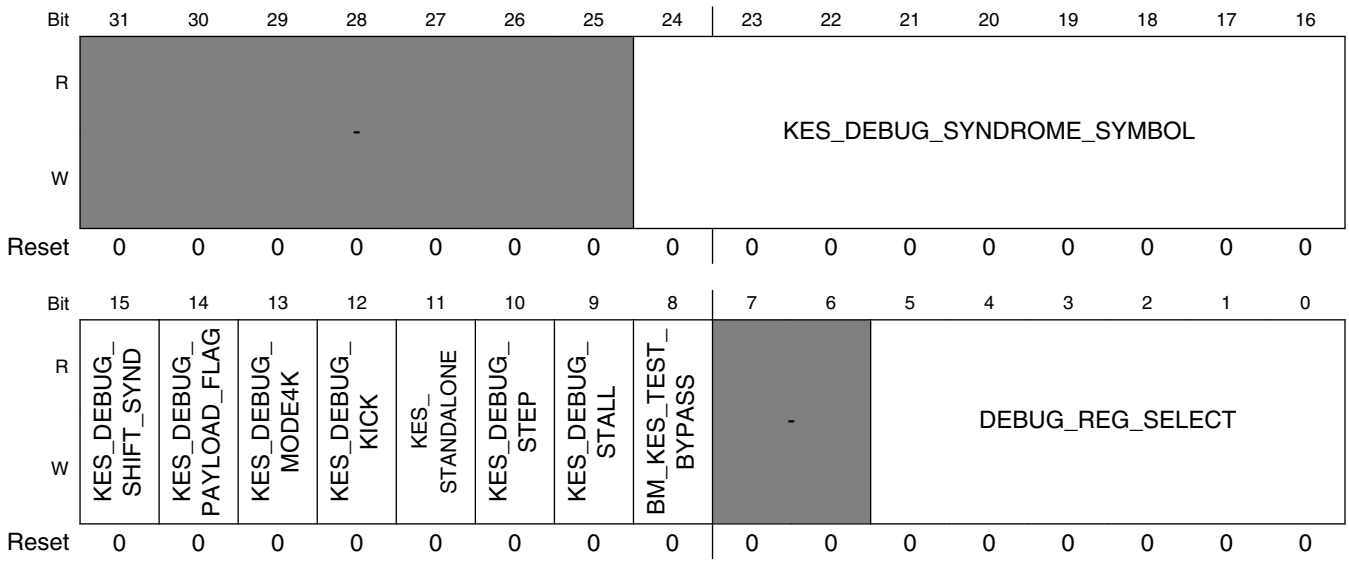
The hardware BCH accelerator internal state machines and signals can be seen in the ECC debug register.The BCH_DEBUG0 register provides access to various internal state information which might prove useful during hardware debug and validation.

Addresses: BCH_DEBUG0 is 4100_8000h base + 100h offset = 4100_8100h

BCH_DEBUG0_SET is 4100_8000h base + 104h offset = 4100_8104h

BCH_DEBUG0_CLR is 4100_8000h base + 108h offset = 4100_8108h

BCH_DEBUG0_TOG is 4100_8000h base + 10Ch offset = 4100_810Ch



BCH_DEBUG0*n* field descriptions

Field	Description
31–25 -	Reserved, always set these bits to zero.
24–16 KES_DEBUG_SYNDROME_SYMBOL	The 9 bit value in this bit field will be shifted into the syndrome register array at the input of the KES engine whenever BCH_DEBUG0_KES_DEBUG_SHIFT_SYND is toggled. 0x0 **NORMAL** — Bus master address generator for synd_gen writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxilliary block.
15 KES_DEBUG_SHIFT_SYND	Toggling this bit causes the value in BCH_DEBUG0_KES_SYNDROME_SYMBOL to be shift into the syndrome register array at the input to the KES engine. After shifting in 16 symbols, one can kick off both KES and CF cycles by toggling BCH_DEBUG0_KES_DEBUG_KICK. Be sure to set KES_BCH_DEBUG0_KES_STANDALONE mode to 1 before kicking.
14 KES_DEBUG_PAYLOAD_FLAG	When running the stand alone debug mode on the error calculator, the state of this bit is presented to the KES engine as the input payload flag. 0x1 **DATA** — Payload is set for 512 byte data block. 0x1 **AUX** — Payload is set for 65 or 19 byte auxilliary block.

Table continues on the next page...

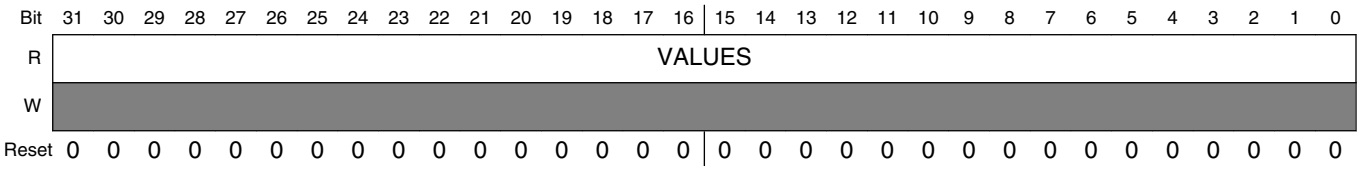
BCH_DEBUG0*n* field descriptions (continued)

Field	Description
13 KES_DEBUG_ MODE4K	When running the stand alone debug mode on the error calculator, the state of this bit is presented to the KES engine as the input mode (4K or 2K pages). 0x1 **4k** — Mode is set for 4K NAND pages. 0x1 **2k** — Mode is set for 2K NAND pages.
12 KES_DEBUG_ KICK	Toggling causes KES engine FSM to start as if kick by the Bus Master. This allows stand alone testing of the KES and Chien Search engines. Be sure to set KES_BCH_DEBUG0_KES_STANDALONE mode to 1 before kicking.
11 KES_ STANDALONE	Set to one to cause the KES engine to suppress toggling the KES_BM_DONE signal to the bus master and to suppress toggling the CF_BM_DONE signal by the CF engine. 0x0 **NORMAL** — Bus master address generator for synd_gen writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxilliary block.
10 KES_DEBUG_ STEP	Toggling this bit causes the KES FSM to skip passed the stall state if it is in DEBUG_STALL mode and it has completed processing a block.
9 KES_DEBUG_ STALL	Set to one to cause KES FSM to stall after notifying Chien search engine to start processing its block but before notifying the bus master that the KES computation is complete. This allows a diagnostic to stall the FSM after each blocks key equations are solved. This also has the effect of stalling the CSFE search engine so it's state can be examined after it finishes processing the KES stalled block. 0x0 **NORMAL** — KES FSM proceeds to next block supplied by bus master. 0x1 **WAIT** — KES FSM waits after current equations are solved and the search engine is started.
8 BM_KES_ TEST_BYPASS	1 = Point all synd_gen writes to dummy area at the end of the AUXILLIARY block so that diagnostics can preload all payload, parity bytes and computed syndrome bytes for test the KES engine. 0x0 **NORMAL** — Bus master address generator for synd_gen writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxilliary block.
7–6 -	Reserved, always set these bits to zero.
5–0 DEBUG_REG_ SELECT	The value loaded in this bit field is used to select the internal register state view of KES engine or the Chien search engine.

14.7.17 KES Debug Read Register (BCH_DBGKESREAD)

The hardware BCH ECC accelerator key equation solver internal state machines and signals can be seen in the ECC debug registers.

Address: BCH_DBGKESREAD is 4100_8000h base + 110h offset = 4100_8110h



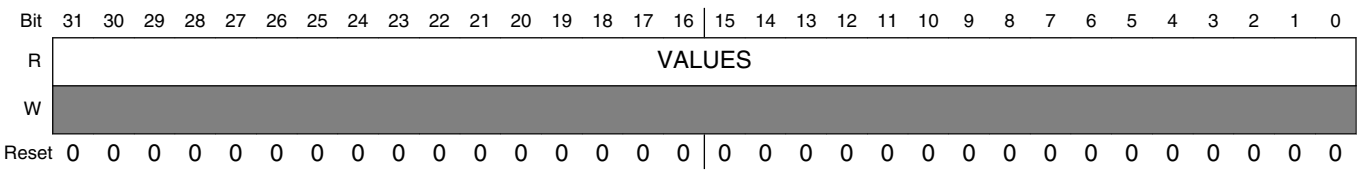
BCH_DBGKESREAD field descriptions

Field	Description
31–0 VALUES	This register will return the KES debug value.

14.7.18 Chien Search Debug Read Register (BCH_DBGCSFEREAD)

The hardware BCH ECC accelerator Chien Search internal state machines and signals can be seen in the ECC debug registers.

Address: BCH_DBGCSFEREAD is 4100_8000h base + 120h offset = 4100_8120h



BCH_DBGCSFEREAD field descriptions

Field	Description
31–0 VALUES	Reserved

14.7.19 Syndrome Generator Debug Read Register (BCH_DBGSYNDGENREAD)

The hardware BCH ECC accelerator syndrome generator internal state machines and signals can be seen in the ECC debug registers.

Address: BCH_DBGSYNDGENREAD is 4100_8000h base + 130h offset = 4100_8130h

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

BCH_DBGSYNDGENREAD field descriptions

Field	Description
31–0 VALUES	Reserved

14.7.20 Bus Master and ECC Controller Debug Read Register (BCH_DBGAHBMREAD)

The hardware BCH ECC accelerator bus master and ecc controller internal state machines and signals can be seen in the ECC debug registers.

Address: BCH_DBGAHBMREAD is 4100_8000h base + 140h offset = 4100_8140h

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

BCH_DBGAHBMREAD field descriptions

Field	Description
31–0 VALUES	Reserved

14.7.21 Block Name Register (BCH_BLOCKNAME)

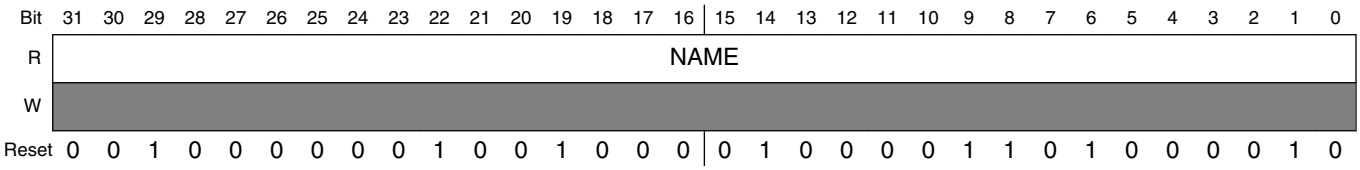
Read only view of the block name string BCH.

Fixed pattern read only value for test puposes. Can be read as an ASCII string with the zero termination coming from the first byte of the BLOCKVERSION register.

EXAMPLE

```
char *cp = ((char *)BCH_BLOCKNAME_ADDR);   reads back "BCH ".
```

Address: BCH_BLOCKNAME is 4100_8000h base + 150h offset = 4100_8150h



BCH_BLOCKNAME field descriptions

Field	Description
31–0 NAME	Should be the ascii characters "BCH " (0x20, H, C, B).

14.7.22 BCH Version Register (BCH_VERSION)

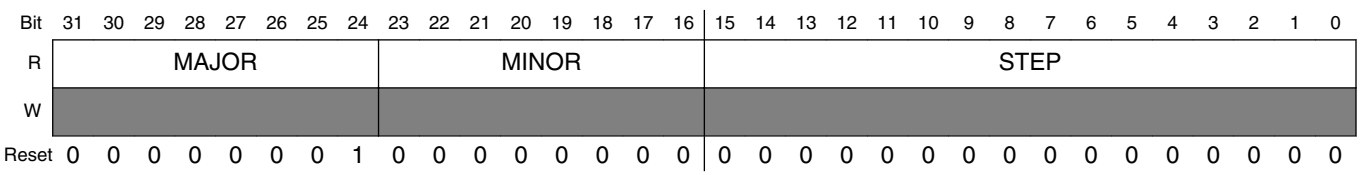
This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
if (BCH_VERSION.B.MAJOR != 1) Error();
```

Address: BCH_VERSION is 4100_8000h base + 160h offset = 4100_8160h



BCH_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 15
Clock Amplifier (CAMP)

15.1 Introduction

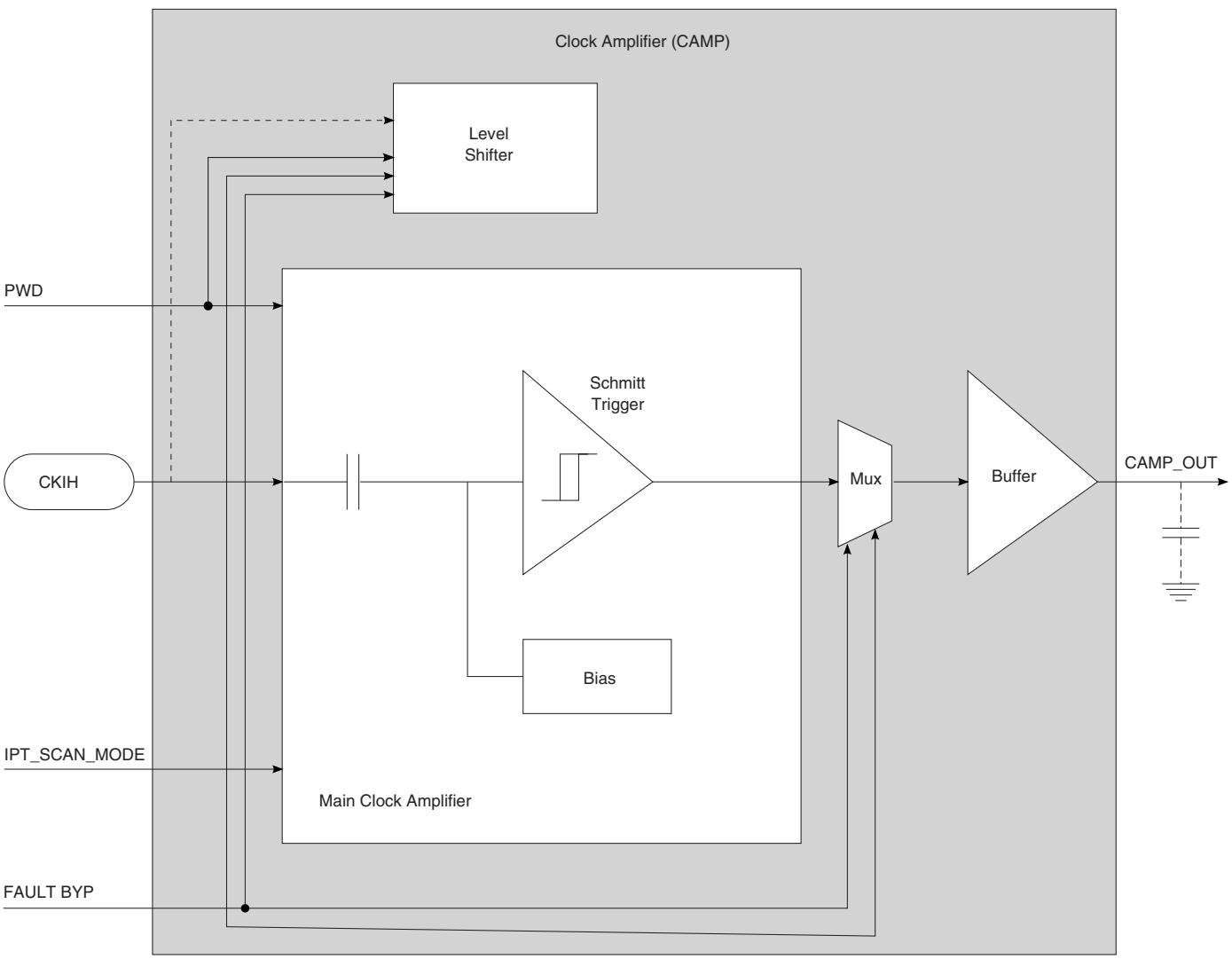


Figure 15-1. Clock Amplifier Block Diagram

15.1.1 Overview

The Clock Amplifier converts a square wave/sinusoidal input of frequency range 8-40 MHz, into a rail to rail square wave (CAMP supply voltage).

The input of the CAMP block is CKIH and the output is CAMP_OUT. The input to CAMP is internally AC coupled (in normal mode); no external coupling is required.

15.1.2 Features

The CAMP includes the following features:

- Converts sinusoidal input to square wave.
- Can accept a square wave of amplitude greater than the supply voltage of the block.

15.1.3 Modes of Operation

The CAMP supports three modes of operation.

15.1.3.1 Normal Mode

In this mode of operation, the Clock Amplifier accepts a sinusoidal/square wave as input and gives a rail to rail square wave output.

15.1.3.2 Power Down Mode

In this mode the CAMP is disabled and put in the low power state. CAMP_OUT remains at logic'0' level during power down.

15.1.3.3 Test Mode (Fault Bypass or Scan)

The Test mode is to be used for testing purpose or during scan. The MAIN CLOCK AMPLIFIER is bypassed in this mode.

15.2 External Signal Description

15.2.1 Signals Overview

Table 15-1. CAMP External Signal Properties

Name	Direction	Function
CKIH	Input	Input clock

Table 15-2. CAMP Interface Signal Properties

Name	Direction	Function
VDD	Input	Power supply
VSS	Input	Ground
PWD	Input	Power down signal
FAULT_BYP	Input	Test mode control signal
IPT_SCAN_MODE	Input	Scan mode signal
CAMP_OUT	Output	Output clock from CAMP

15.2.2 Detailed Signal Description

15.2.2.1 CKIH - External Clock Input

The signal CKIH is the clock input signal to the CAMP. It can be a sinusoidal input with a minimum swing of 400mV p-p or a square wave. It comes directly from the pad.

15.2.2.2 VDD - Power supply

The VDD is the power supply for the CAMP. It supplies the main clock amplifier, buffer and level shifter.

15.2.2.3 VSS - Ground

The VSS input is the ground for the CAMP.

15.2.2.4 IPT_SCAN_MODE - Scan Signal

This signal is active high. When this signal is asserted CAMP is forced to "Test Mode" irrespective of states of fault_byp or pwd signals.

15.2.2.5 PWD - Power Down Signal

This signal when asserted (and ipt_scan_mode = 0), puts the CAMP in the low power mode. This signal is active high.

15.2.2.6 FAULT_BYP - Control Signal for Test Mode

This signal when asserted puts CAMP in test mode. This signal is active high. When this signal is low (and ipt_scan_mode = 0) CAMP works in normal mode.

15.2.2.7 CAMP_OUT - Clock Output from CAMP

This signal is the output of the CAMP.

15.3 Memory Map/Register Definition

The CAMP gets its control signals PWD and FAULT_BYP from registers residing in the CRM. It gets its control signal IPT_SCAN_MODE from TSCM (Test and Scan Control sub-block).

15.4 Functional Description

This section provides a functional description of the Clock Amplifier. The block can be divided into three parts. First the Main Clock Amplifier, second the Output Buffer and third the Level Shifter.

15.4.1 CAMP Sub-Blocks

This section describes the Main Clock amplifier, the Output Buffer and Level Shifter of CAMP. (See Figure 15-1).

15.4.1.1 Main Clock Amplifier

The input clock is ac coupled to the input of a Schmitt trigger. A dc bias generation circuit is used to provide a fixed dc to the input of the Schmitt trigger. The Schmitt trigger converts a sinusoidal signal to a square wave.

15.4.1.2 Output Buffer

The output of the main clock amplifier is buffered by the output buffer.

15.4.1.3 Level Shifter

This is activated in the test mode. It level shifts the input signal to CAMP supply.

15.4.2 CAMP Modes of Operation

Table 15-3. Truth Table for CAMP

IPT_SCAN_MODE	PWD	FAULT_BYP	CAMP_OUT	Main Clock Amplifier Path	Level Shifter Path
0	0	0	Square Wave	Enabled	Disabled
0	0	1	Square Wave	Disabled	Enabled
0	1	X	Logic'0'	Disabled	Disabled
1	X	X	Square Wave	Disabled	Enabled

15.4.2.1 Normal Mode

In this mode of operation the IPT_SCAN_MODE, PWD and FAULT_BYP signals are de-asserted. The sinusoidal (or square wave) clock input (with peak to peak amplitude not exceeding 3 Volts) is converted to square wave with amplitude equal to the Camp supply.

15.4.2.2 Power Down Mode

In this mode of operation, the CAMP is powered down. The supply to Main Clock Amplifier and level shifter is turned off by power gating. The CAMP_OUT is a t logic '0' level.

15.4.2.3 Test Mode

In this mode the MAIN CLOCK AMPLIFIER is bypassed via a level shifter. It accepts only square wave input with voltage swing greater than or equal to CAMP supply voltage. There is no guarantee of output clock duty cycle in this mode. To obtain 45%-55% duty cycle output from CAMP in this mode, the input must be a square wave at CAMP's supply level with slews less than or equal to 2ns.

15.5 Initialization/Application Information

During initialization it has to be ensured that PWD and IPT_SCAN_MODE remain de-asserted.

Chapter 16
Configurable Serial Peripheral Interface (CSPI)

16.1 Overview

This chapter describes a block integrated into an SoC. The chapter is intended for a block driver software developer. It describes block-level operation and programming. To understand how the block is integrated at the SoC level, a system software developer should see discussions of the block in the appropriate SoC-level chapter(s).

The Configurable Serial Peripheral Interface (CSPI) block is a full-duplex, synchronous, four-wire serial communication block. The CSPI block contains an 8 x 32 receive buffer (RXFIFO) and an 8 x 32 transmit buffer (TXFIFO). With data FIFOs, the CSPI allows rapid data communication with fewer software interrupts. Figure 16-1 shows a block diagram of the CSPI.

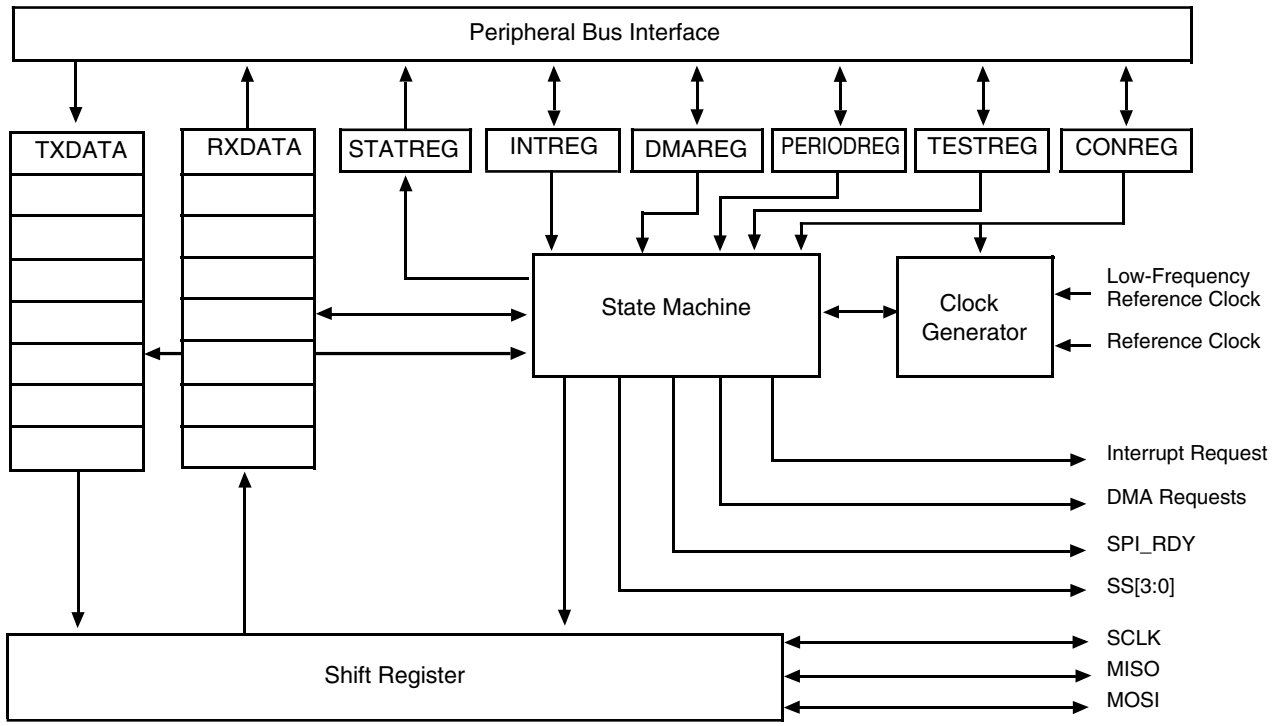


Figure 16-1. CSPI Block Diagram

16.1.1 Features

Key features of the CSPI include:

- Full-duplex synchronous serial interface
- Master/Slave configurable
- Four Chip Select (SS) signals to support multiple peripherals
- Transfer continuation function allows unlimited length data transfers
- 32-bit wide by 8 -entry FIFO for both transmit and receive data
- Polarity and phase of the Chip Select (SS) and SPI Clock (SCLK) are configurable
- Direct Memory Access (DMA) support
- Max operation frequency up to one-quarter of the reference clock frequency.

16.1.2 Modes and Operations

The CSPI supports the modes described in the indicated sections:

- Operating Modes
  - Master Mode
  - Slave Mode
- Low Power Modes

As described in Operations, the CSPI supports the operations described in the indicated sections:

- Typical Master Mode
  - Master Mode with SPI_RDY
  - Master Mode with Wait States
  - Master Mode with SSCTL Control
  - Master Mode with Phase Control
- Typical Slave Mode

16.2 External Signals

The following table lists conventions for representing signals.

Table 16-1. Block Signal Conventions

Category	Convention	Example(s)
Off-chip signal	Uppercase (all capital letters)	TXD
Internal signal[1]	Lowercase italics	*core_int*
Active low signal	_B (_b) suffix or overbar	RESET_EN_B or $\overline{\text{RESET\_EN}}$
Range of bussed or commonly named signals	Beginning and end points of the range are: • Separated by a colon. • Surrounded by square brackets.	ADDR[31:0] CSE_B[7:0] or $\overline{\text{CSE}}$[7:0]
Individual signal in a range of bussed or commonly named signals	Individual number in the range appears without a colon or square brackets	ADDR31 CSE0_B or $\overline{\text{CSE0}}$

1. Internal signals are for reference only in descriptions of internal block or SoC functionality.

The following table describes all CSPI signals that connect off-chip.

Table 16-2. Off-Chip Block Signals

Signal	I/O	Description	Reset State[1]	Pull-Up/ Down[1]
$\overline{\text{SS}}$[3:0]	I/O	Chip selects	1	-
SCLK	I/O	SPI clock	0	Active
MISO	I/O	Master data in; slave data out	0	Passive
MOSI	I/O	Master data out; slave data in	0	-
SPI_RDY	I	Master data out; slave data in	0	Active

1. The reset state values and pull-up/down requirements provided in this table are from the block-level perspective. To understand how the block is integrated at the SoC level, the system software developer must see discussions of the block in the appropriate SoC-level chapter(s). For example, a block signal that requires a pull-up could be integrated with a pull-up option built into the SoC. In this case, the system software developer must ensure the proper programming at the SoC level.

The following figure shows the CSPI in master mode connected to four external devices in a one-way communication link.

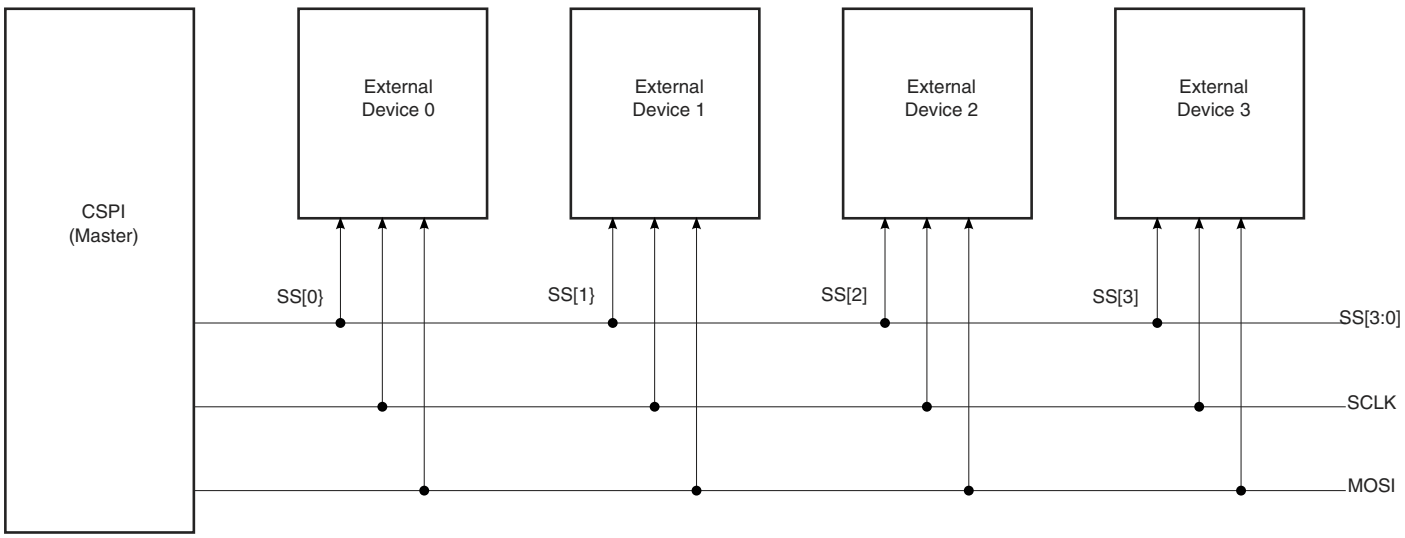


Figure 16-2. Example Connection Diagram

16.3 Functional Description

This section provides a complete functional description of the CSPI. The following figure shows the relationship of SCLK and data lines while CSPI has been configured with different POL and PHA settings.

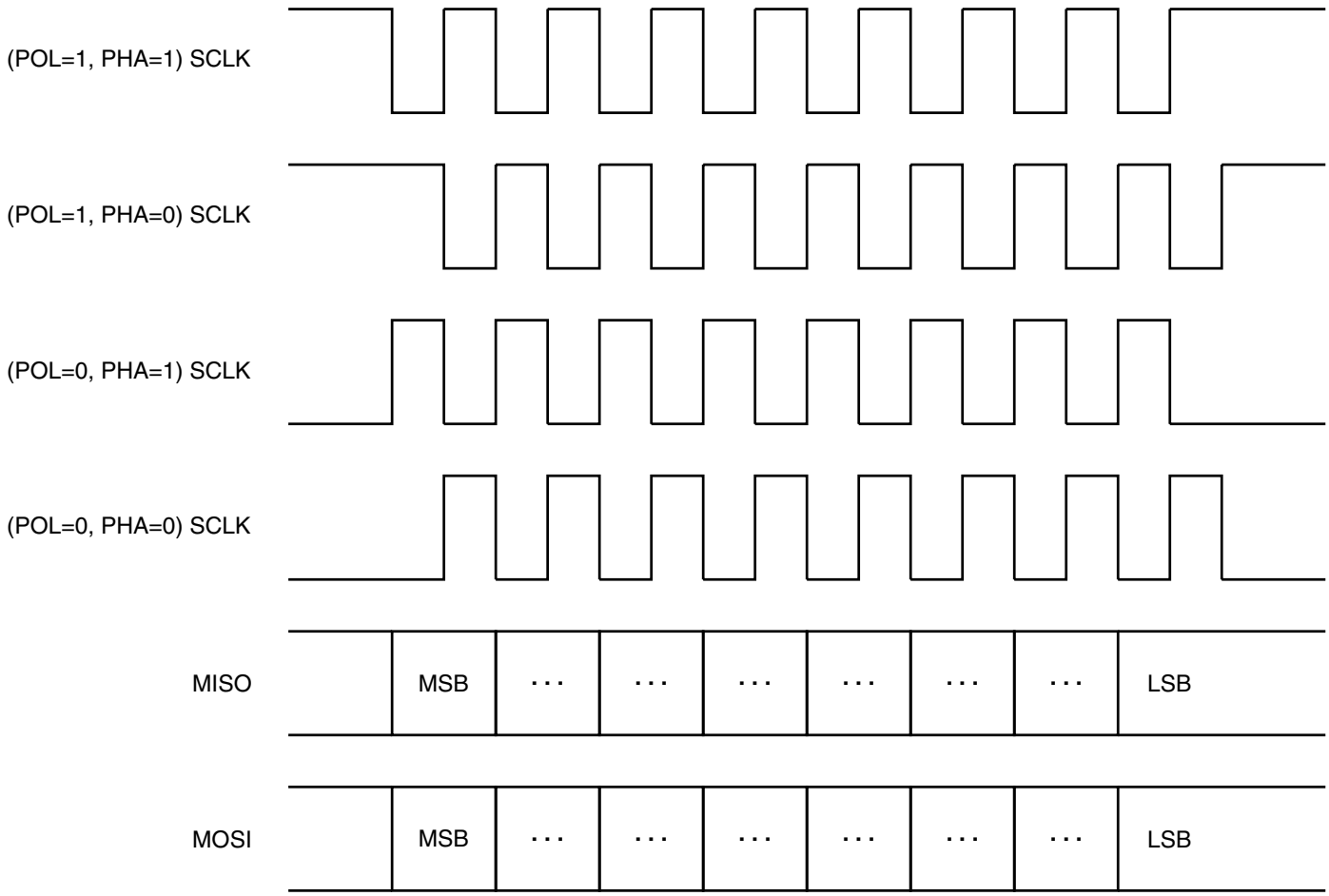


Figure 16-3. CSPI SCLK, MISO, and MOSI Relationship

16.3.1 Operating Modes

CSPI has two operating modes, master mode and slave mode. This section describes all functional operation modes of the CSPI.

16.3.1.1 Master Mode

When the CSPI is configured as a master, it uses a serial link to transfer data between the CSPI and an external slave device. One of the Chip Select (SS) signals and the clock signal (SCLK) are used to transfer data between two devices. If the external device is a transmit-only device, the CSPI master's output port can be ignored and used for other purposes. In order to use the internal TXFIFO and RXFIFO, two auxiliary output signals,

Chip Select (SS) and CSPI_CONREG[DRCTL], are used for data transfer rate control. Software can also configure the sample period control register to a fixed data transfer rate.

16.3.1.2 Slave Mode

When the CSPI is configured as a slave, software can configure the CSPI Control register to match the external SPI master's timing. In this configuration, Chip Select ($\overline{SS}$) becomes an input signal, and is used to control data transfers through the Shift register, as well as to load/store the data FIFO.

16.3.2 Low Power Modes

The CSPI does not operate under low power mode. It holds its operation when its clock is gated off in master mode. In slave mode, the CSPI does not respond when its clock is gated off.

16.3.3 Operations

This section describes the CSPI's operations.

16.3.3.1 Typical Master Mode

The CSPI master uses the Chip Select (SS) signal to enable an external SPI device, and uses the SCLK signal to transfer data in and out of the Shift register. The SPI_RDY enables fast data communication with fewer software interrupts. By programming the CSPI_PERIODREG register accordingly, the CSPI can be used for a fixed data transfer rate.

When the CSPI is in Master mode the SS, SCLK, and MOSI are output signals, and the MISO signal is an input. The following figure shows a typical SPI burst.

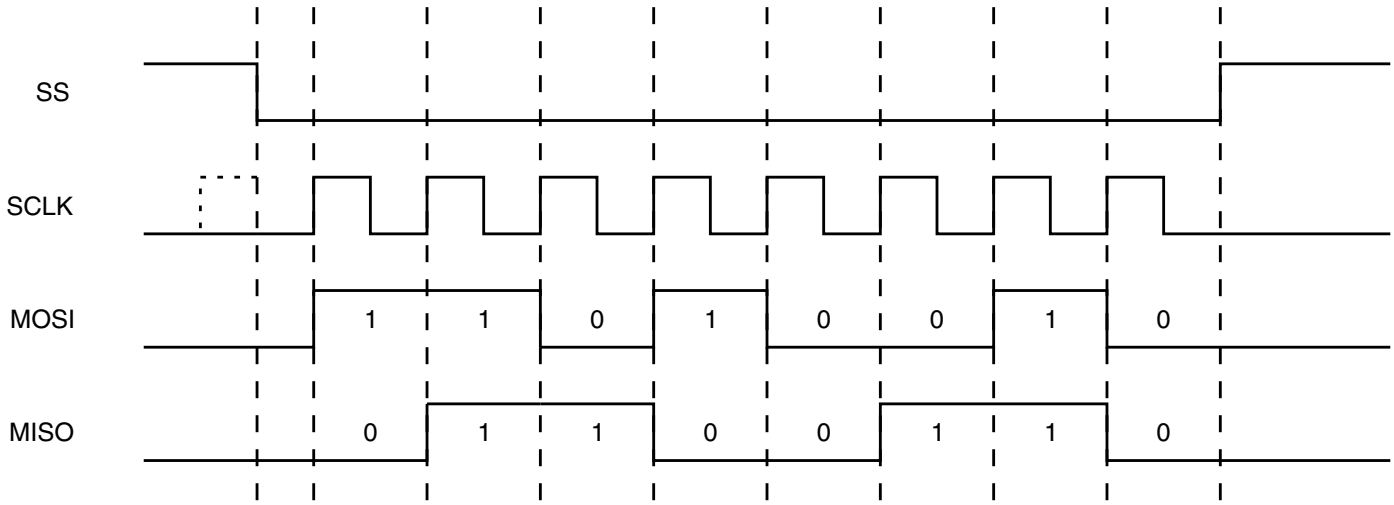


Figure 16-4. Typical SPI Burst (8-bit Transfer)

The Chip Select (SS) signal enables the selected external SPI device, and the SCLK synchronizes the data transfer. The MOSI and MISO signals change on rising edge of SCLK and the MISO signal is latched on the falling edge of the SCLK. The above figure shows a data of 0xD2 is shifted out, and a data of 0x66 is shifted in.

16.3.3.1.1 Master Mode with SPI_RDY

By default, the CSPI does not use the SPI_RDY signal in master mode (MODE =1). A SPI burst begins when the following events happen:

- The CSPI is enabled, TXFIFO has data in it, and CONREG[XCH] bit or the CONREG[SMC] bit is set.
- When the SPI Data Ready Control (CONREG[DRCTL]) bits contains either 01 or 10, the SPI_RDY signal controls when a SPI burst starts.

A SPI burst is defined as a bus transaction that starts when the slave select is asserted and ends when the slave select is negated. The Chip Select (SS) signal will remain asserted until all the bits in a SPI burst are shifted out.

If CONREG[DRCTL] is set to 01, the SPI burst can be triggered only if a falling edge of the SPI_RDY signal from CSPI_CONREG[DRCTL] register has been detected. The following figure shows the relationship between a SPI burst and the falling edge of SPI_RDY signal.

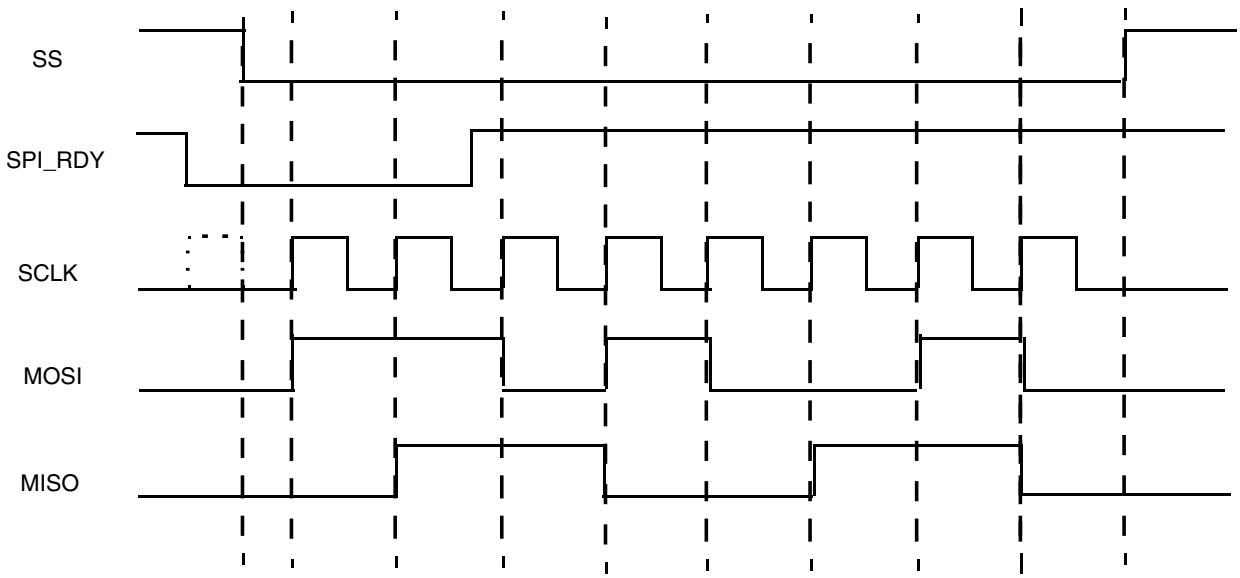


Figure 16-5. Relationship Between a SPI Burst and SPI_RDY: Falling-Edge Triggered

A SPI burst does not start until the falling edge of the SPI_RDY signal is detected. The next SPI burst starts when the next SPI_RDY falling edge is detected, after the last burst has finished.

If SPI Data Ready Control (CONREG[DRCTL]) is set to 10, the SPI burst can be triggered only if the SPI_RDY signal is low. The following figure shows the relationship between a SPI burst and the SPI_RDY signal. The SPI burst does not begin until the SPI_RDY signal goes low. The CSPI will keep transmitting SPI burst if the SPI_RDY signal remains low.

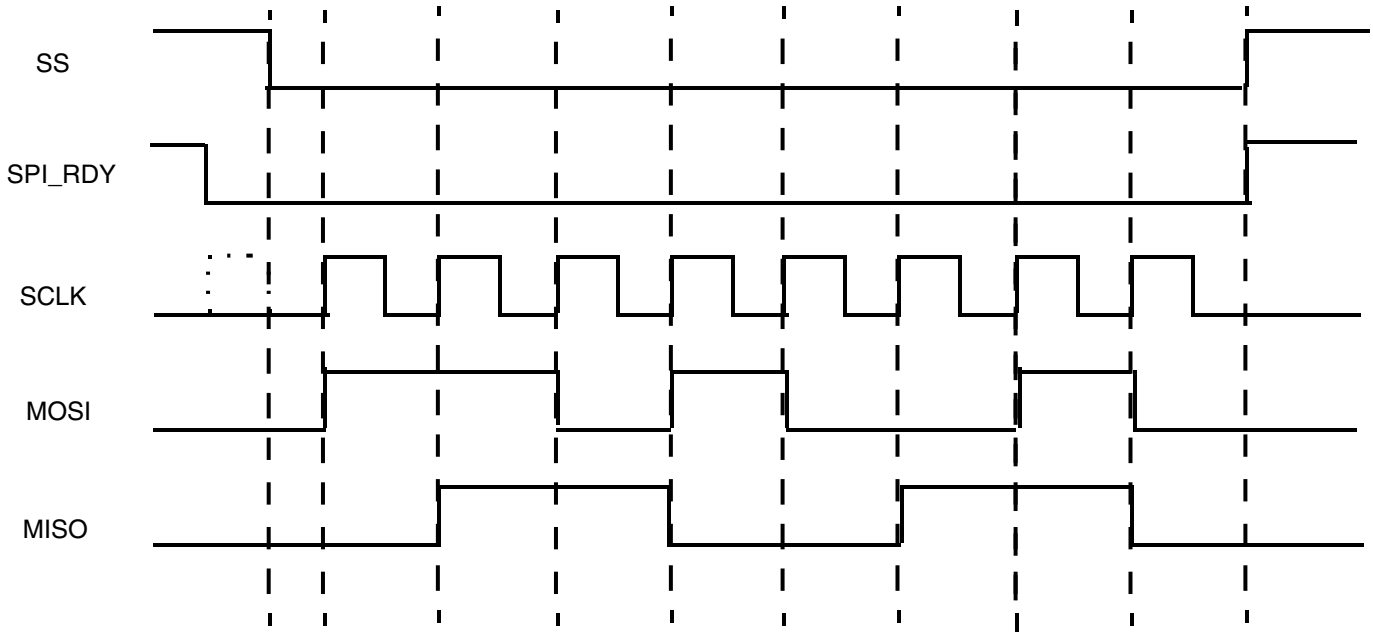


Figure 16-6. Relationship Between a SPI Burst and SPI_RDY: Low-Level Triggered

16.3.3.1.2 Master Mode with Wait States

Wait states can be inserted between SPI bursts. This provides a way for software to slow down the SPI burst to meet the timing requirements of a slower SPI device. The following figure shows wait states inserted between SPI bursts.

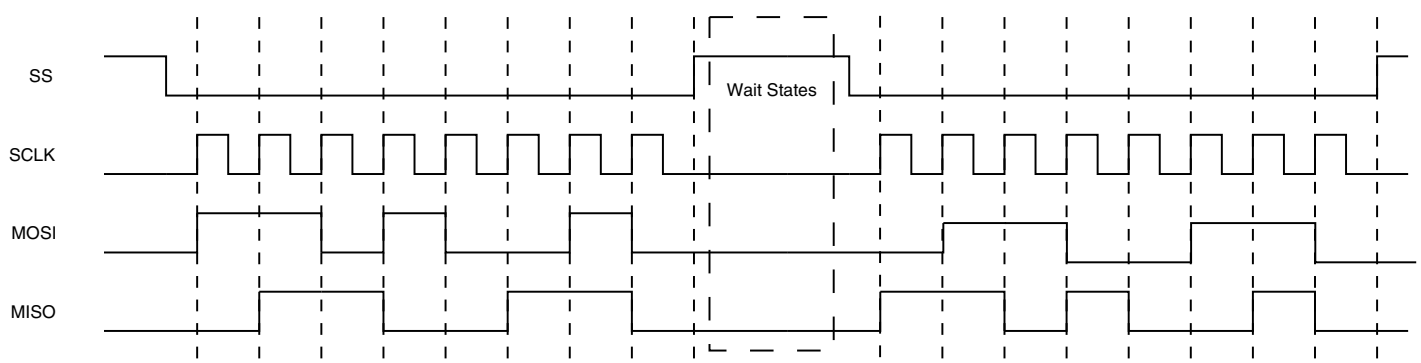


Figure 16-7. SPI Bursts with Wait States

In this case, the number of wait states is controlled by PERIODREG[SAMPLE PERIOD] and the wait states' clock source is selected by PERIODREG[CSRC].

16.3.3.1.3 Master Mode with SSCTL Control

The SPI SS Control (SSCTL) bit controls whether the current operation is single burst or multiple bursts. When the SPI SS Wave Form Select (SSCTL) bit is set, the current operation is multiple bursts transfer. When the SPI SS Wave Form Select (SSCTL) bit is cleared, the current operation is single burst transfer. A SPI burst can contains multiple words as defined in the BURST LENGTH field of the CSPI_CONREG register.

The following figure shows one SPI burst while SSCTL is clear.

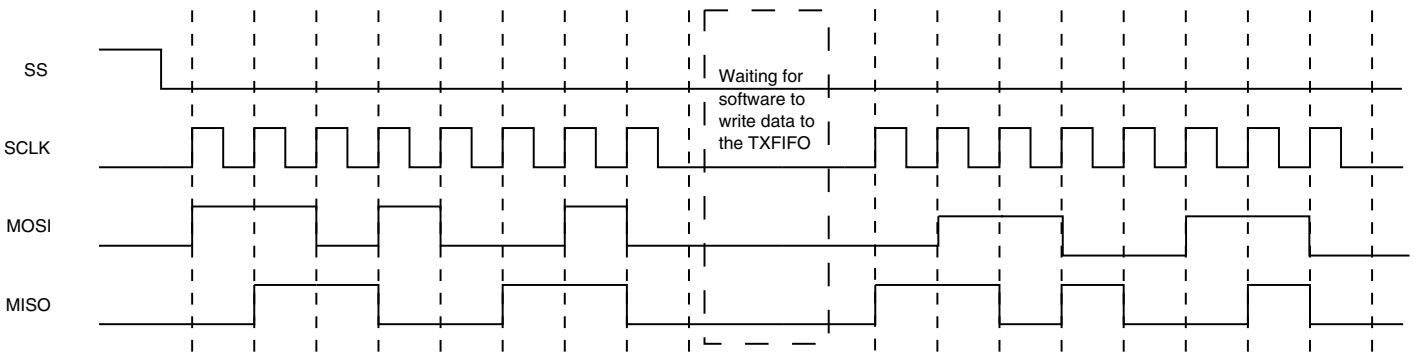


Figure 16-8. SPI Burst While SSCTL is Clear

Two 8-bit bursts in the TXFIFO have been combined and transmitted in one SPI burst. The maximum length of a single SPI burst is defined in the BURST LENGTH field of the CONREG control register. (The above figure corresponds to a BURST LENGTH of 8.) This provides a way for transferring a longer SPI burst by writing data into TXFIFO while the CSPI is transmitting.

The following figure shows two SPI bursts are transmitted while SSCTL is set.

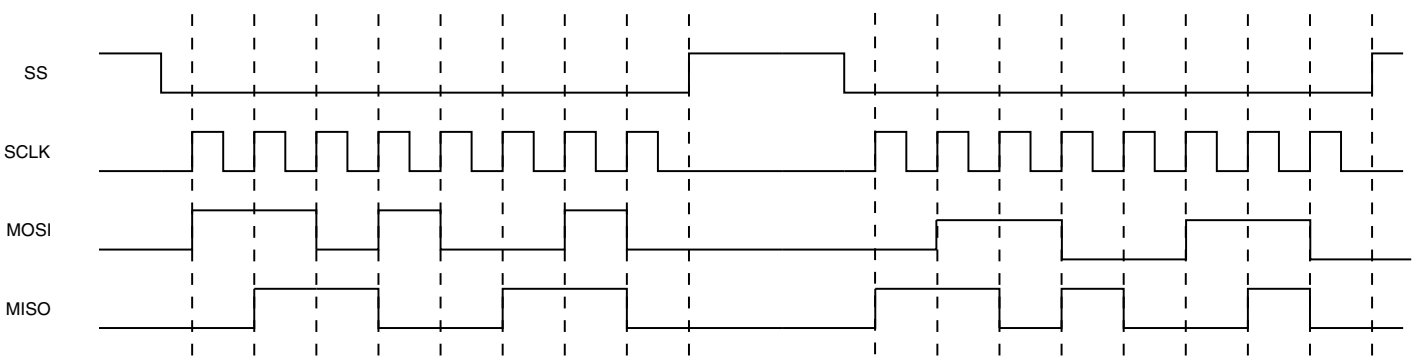


Figure 16-9. SPI Bursts While SSCTL is Set

Two FIFO entries are transmitted, one entry with each SPI burst. The CSPI will continue to transmit SPI bursts until the TXFIFO is empty. When wait states can be inserted between SPI bursts, the SS will negate between SPI bursts until the wait states finish.

16.3.3.1.4 Master Mode with Phase Control

The Phase Control (CSPI_CONREG[PHA]) bit controls how the transmit data shifts out and the receive data shifts in.

When the Phase control (CSPI_CONREG[PHA]) bit is set, the transmit data will shift out on the rising edge of SCLK, and the receive data is latched on the falling edge of SCLK. The most-significant bit is output on the first rising SCLK edge.

When CSPI_CONREG[PHA] is cleared, the transmit data is shifted out on the falling edge of SCLK and the receive data is latched on the rising edge of SCLK. The MSB is output when the host processor loads the transmitted data.

Inverting the SCLK polarity does not impact the edge-triggered operations because they are internal to the serial peripheral interface master.

The following figure shows a SPI burst using different POL and PHA configurations.

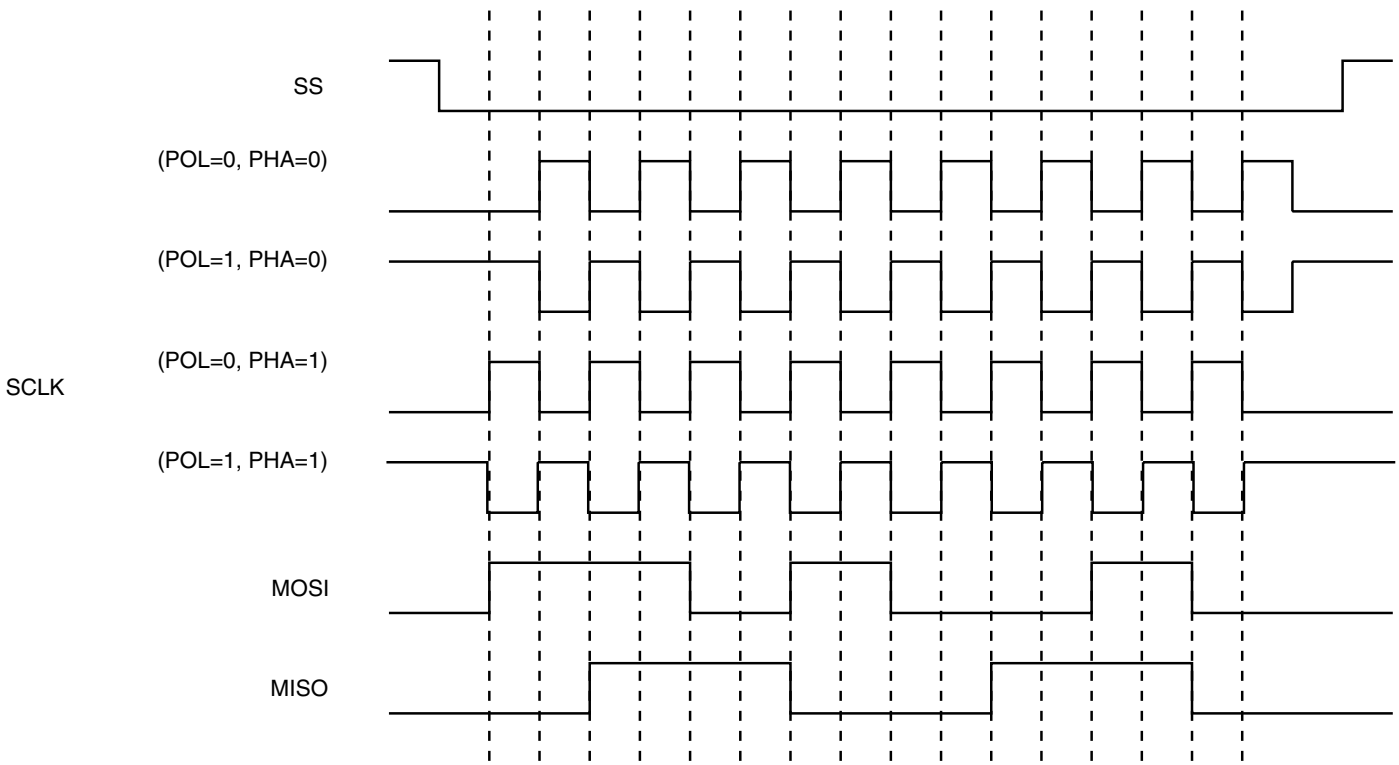


Figure 16-10. SPI Burst with Different POL and PHA Configurations

16.3.3.2 Typical Slave Mode

When the CSPI is configured as a slave (Mode = 0), software can configure the CSPI Control register to match the external SPI master's timing. In this configuration, SS becomes an input signal, and is used to latch data in and out of the internal data Shift registers, as well as to advance the data FIFO.

The SS, SCLK, and MOSI are inputs and MISO is output. Most of the timing diagrams are similar to the diagrams shown previously for the SPI in Master mode (Mode = 1), because the inputs come from a SPI master device.

However, the timing is different when SS is used to advance the data FIFO. When the SSCTL is set while the CSPI is configured in Slave mode, the data FIFO will advance on the rising edge of the SS signal. When the polarity is reversed (SSPOL = 1), the data FIFO will advance on the falling edge of the SS signal.

The following figure shows a SPI burst in which the data FIFO is advanced by the rising edge of the SS signal.

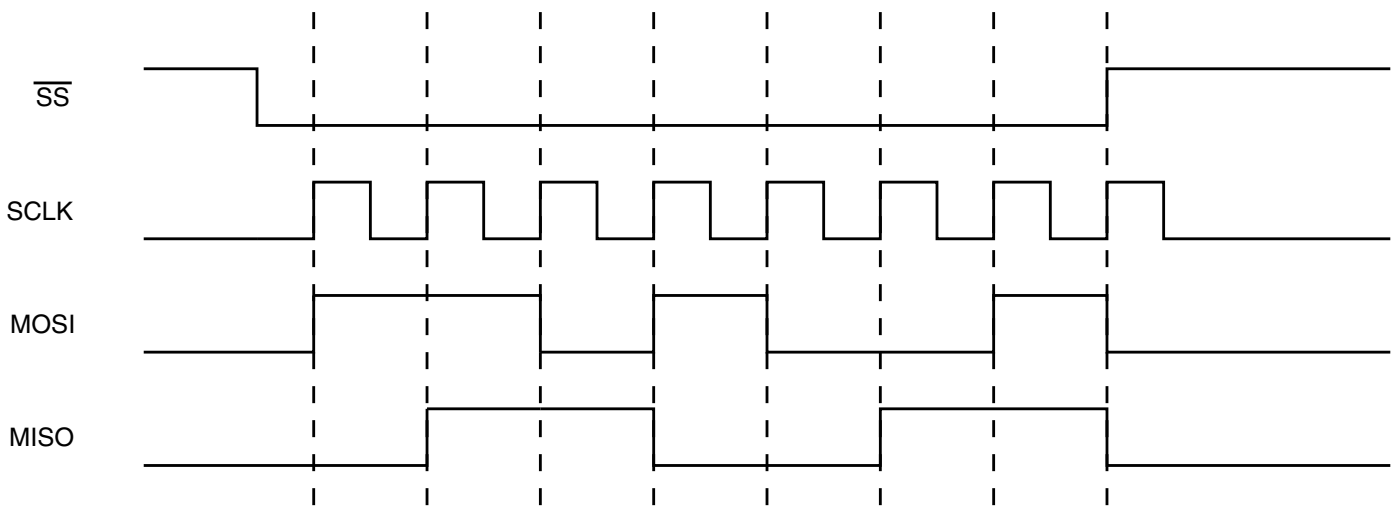


Figure 16-11. Advancing the Data FIFO on the Rising Edge of SS

In the above case, only the most significant 7 bits are loaded to the RXFIFO.

16.3.4 Clocks

This section describes clocks and special clocking requirements of the block.

CSPI has the following clock inputs:

- Reference Clock is the reference clock.
- Low-Frequency Reference Clock is a 32KHz input clock optionally used for counting wait states.

16.3.5 Reset

Whenever a device reset occurs, a reset is performed on the CSPI, resetting all registers to their default values.

Software can reset the block using the CONREG[EN] bit; see Control Register (CSPI_CONREG).

16.3.6 Interrupts

Interrupt control provides a way to manage the CSPI FIFOs:

- For transmitting data, software can enable the TXFIFO empty, TXFIFO half , and TXFIFO full interrupts to maintain the TXFIFO using an interrupt service routine.
- For receiving data, software can enable the RXFIFO ready, RXFIFO half , and RXFIFO full interrupts to retrieve data from the RXFIFO using an interrupt service routine.

Other interrupt sources can be used to control or debug the SPI bursts:

- The transfer-completed interrupt means that there is no data left in the TXFIFO and that the data in the Shift register has been shifted out.
- The RXFIFO overflow interrupt means that the RXFIFO received more than 8 words and will not accept any other words.

The following figure shows a program sequence of SPI bursts using interrupt control.

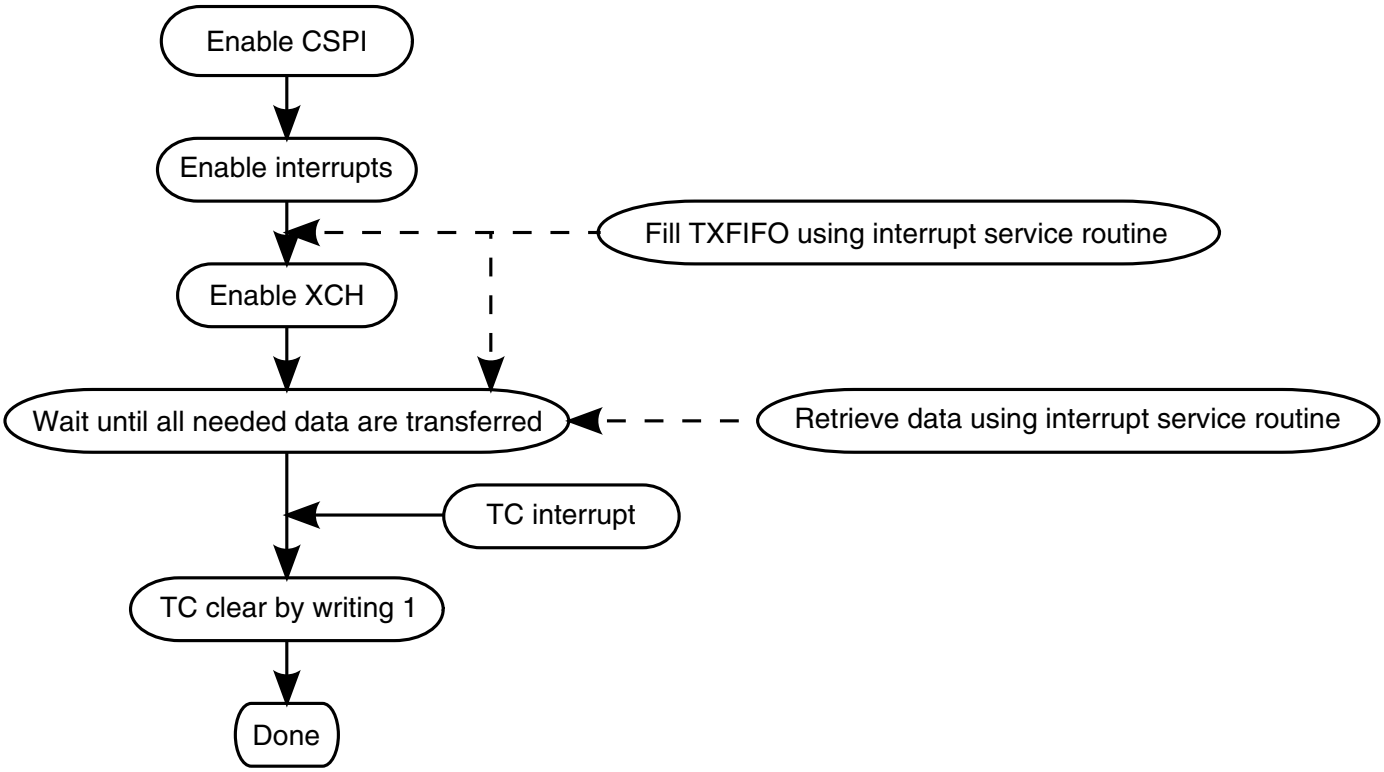


Figure 16-12. Program Sequence of SPI Burst Using Interrupt Control

16.3.7 DMA

DMA control provides another method to utilize the FIFOs in the CSPI. By using DMA request and acknowledge signals, larger amounts of data can be transferred, and will reduce interrupts and host processor loading. When the appropriate conditions are matched, the block will send out a DMA request, and the DMA can deal with the following conditions:

- TXFIFO empty
- TXFIFO half
- RXFIFO half
- RXFIFO full

The following figure shows a program sequence of SPI bursts using DMA control.

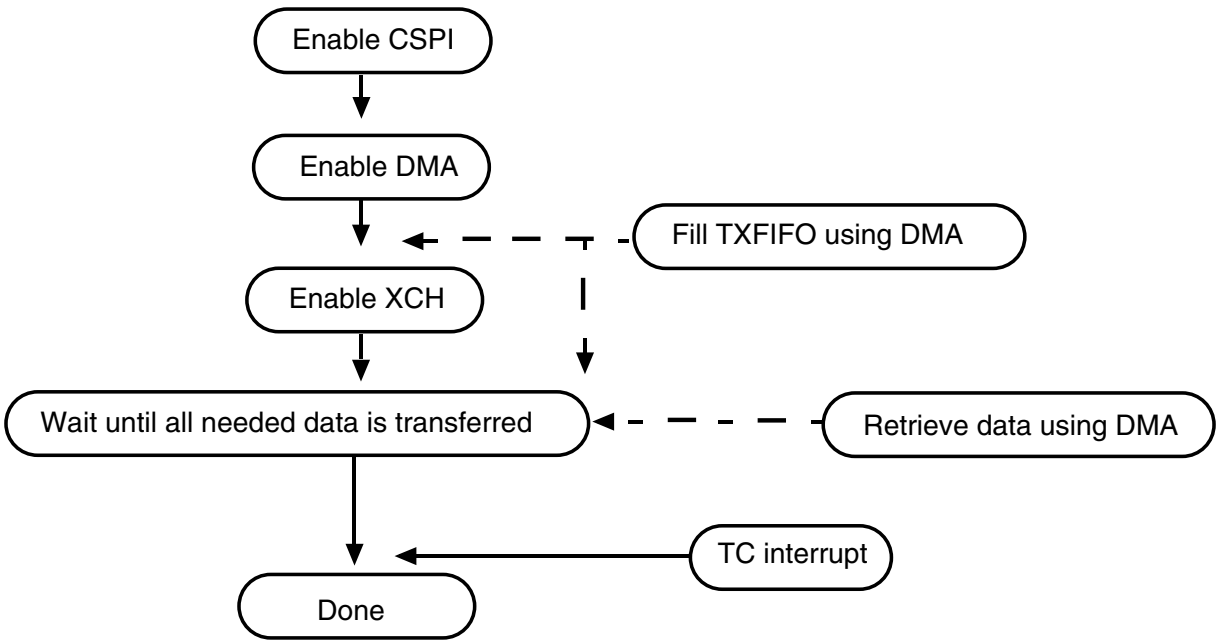


Figure 16-13. Program Sequence of SPI Burst Using DMA

16.3.8 Byte Order

Software can swap bytes for receive data using the TESTREG[SWAP] bit. See Test Control Register (CSPI_TESTREG).

16.4 Initialization

This section provides initialization information for CSPI.

To initialize the block:

1. Clear the EN bit in CSPI_CONREG to reset the block.
2. Enable the clocks for CSPI.
3. Set the EN bit in CSPI_CONREG to put CSPI out of reset.
4. Configure corresponding IOMUX for CSPI external signals.
5. Configure registers of CSPI properly according to the specifications of the external SPI device.

16.5 Applications

The following figure shows two flowcharts for the master and slave mode of operations supported by the CSPI.

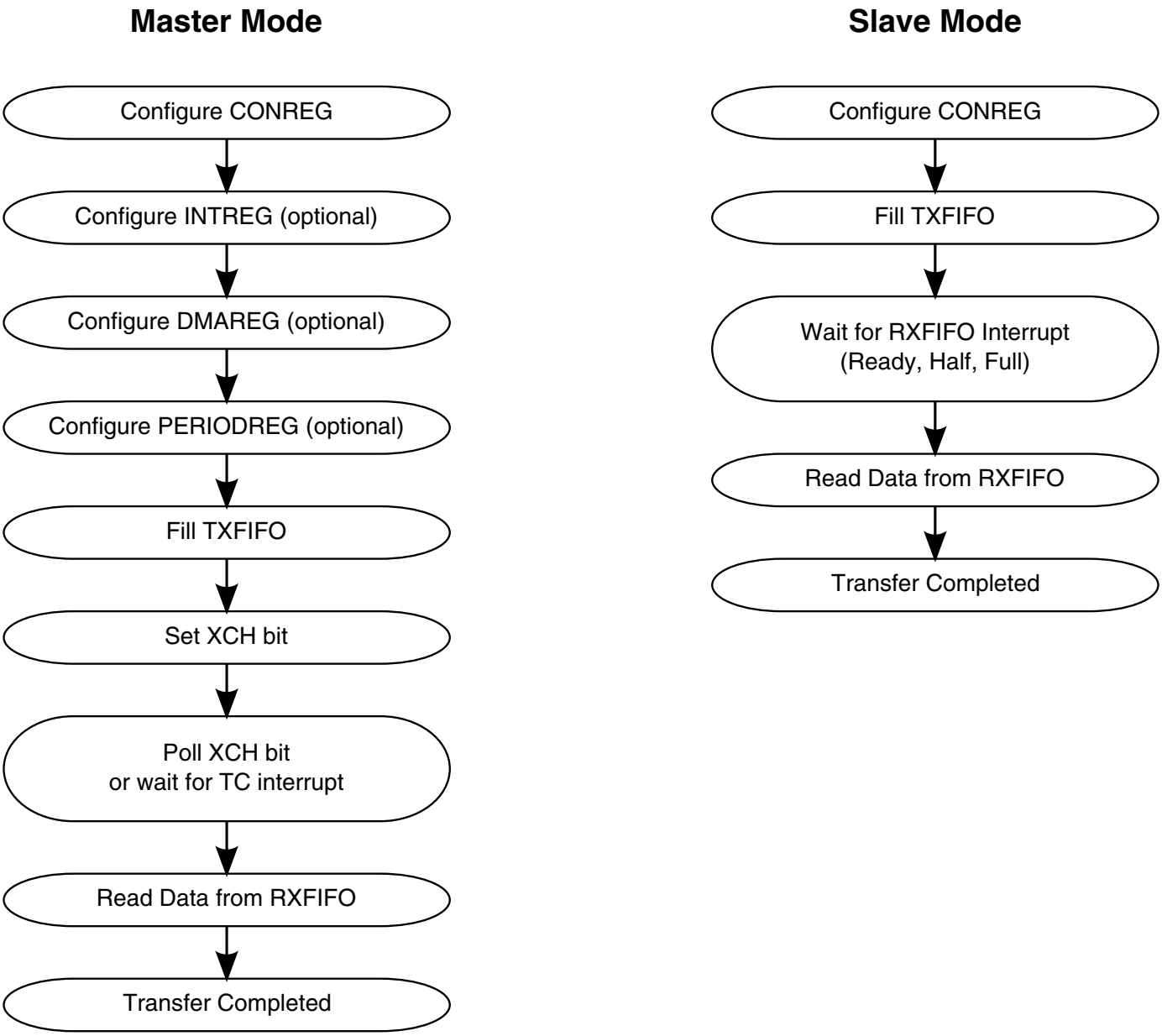


Figure 16-14. Flowchart of the CSPI Operation

The following example shows example code of CSPI operation using ARM instructions.

CSPI Operation using ARM Instructions

```
          LDR R0, =CSPI_BASE_ADDRESS                    ; Load CSPI Base Address to R0

          LDR R1, =0x01F00003                           ; Master Mode, 32-bit transaction

          STR R1, [R0, #0x08]

          LDR R1, =0x00000011                           ; Enable RXFIFO half and TXFIFO empty

          STR R1, [R0, #0x0C]                              ; interrupt (Alternatively with
DMA Mode)

          LDR R1, =0x00000011                           ; Enable RXFIFO half and TXFIFO empty

          STR R1, [R0, #0x10]                              ; DMA (Alternatively with
interrupt)

          LDR R5, =0x05                                 ; R5 as number of words to be
transferred.

          LDR R1, =0x11111111                           ; R1 as increment to generate the
data.

          LDR R2, =0x12345678                           ; R2 load the data to be transferred.

Loop_00

          STR R2, [R2,#0x04]                               ; Store data into TXFIFO.

          ADD R2, R2, R1                                ; Generating next data to be
transferred.

          SUB R5, R5, #1                                ; Decrease the R5.

          CMP R5, #0x00                                 ; Check R5 if it is zero.

          BNE Loop_00                                  ; Loop until R5 is zero.

          LDR R1, =0x01F00007                             ; set XCH bit to start transaction.

          STR R1, [R0, #0x08]

Loop_01

          LDR R1, [R0, #0x08]                              ; check XCH bit if it is cleared.

          LDR R2, =0x00000004

          AND R1, R2, R1

          CMP R1, #0x00

          BEQ PASS_00                                  ; if XCH bit is cleared then finish.

          B Loop_01                                    ; if it isn't cleared then continue loop

          LDR R1, [R0, #0x00]                              ; Read data from RXFIFO.
```

16.6 Programmable Registers

This section includes the block memory map and detailed descriptions of all registers. For the base address of a particular block instantiation, see the system memory map.

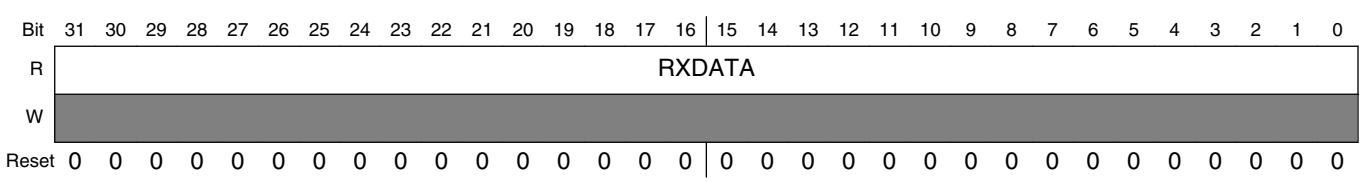
CSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FC_0000	Receive Data Register (CSPI_RXDATA)	32	R	0000_0000h	16.6.1/ 764
63FC_0004	Transmit Data Register (CSPI_TXDATA)	32	W (always reads zero)	0000_0000h	16.6.2/ 765
63FC_0008	Control Register (CSPI_CONREG)	32	R/W	0000_0000h	16.6.3/ 766
63FC_000C	Interrupt Control Register (CSPI_CSPI_INTREG)	32	R/W	0000_0000h	16.6.4/ 769
63FC_0010	DMA Control Register (CSPI_CSPI_DMAREG)	32	R/W	0000_0000h	16.6.5/ 770
63FC_0014	Status Register (CSPI_CSPI_STATREG)	32	R/W	0000_0003h	16.6.6/ 771
63FC_0018	Sample Period Control Register (CSPI_PERIODREG)	32	R/W	0000_0000h	16.6.7/ 772
63FC_001C	Test Control Register (CSPI_TESTREG)	32	R/W	0000_0000h	16.6.8/ 773

16.6.1 Receive Data Register (CSPI_RXDATA)

The Receive Data register (CSPI_RXDATA) is a read-only register that forms the top word of the 8 x 32 receive FIFO. This register holds the data received from an external SPI device during a data transaction. Only word-sized read operations are allowed.

Address: CSPI_RXDATA is 63FC_0000h base + 0h offset = 63FC_0000h



CSPI_RXDATA field descriptions

Field	Description
31–0 RXDATA	Receive Data. This register holds the top word of the receive data FIFO. The FIFO is advanced for each read of this register. The data read is undefined when the Receive Data Ready (RR) bit in the Interrupt Control/Status register is cleared. Zeros are read when CSPI is disabled.

16.6.2 Transmit Data Register (CSPI_TXDATA)

The Transmit Data (CSPI_TXDATA) register is a write-only data register that forms the bottom word of the 8 x 32 TXFIFO. The TXFIFO can be written to as long as it is not full, even when the SPI Exchange bit (XCH) in CSPI_CONREG is set. This allows software to write to the TXFIFO during a SPI data exchange process. Writes to this register are ignored when the CSPI block is disabled (CSPI_CONREG[EN] bit is cleared).

Address: CSPI_TXDATA is 63FC_0000h base + 4h offset = 63FC_0004h

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

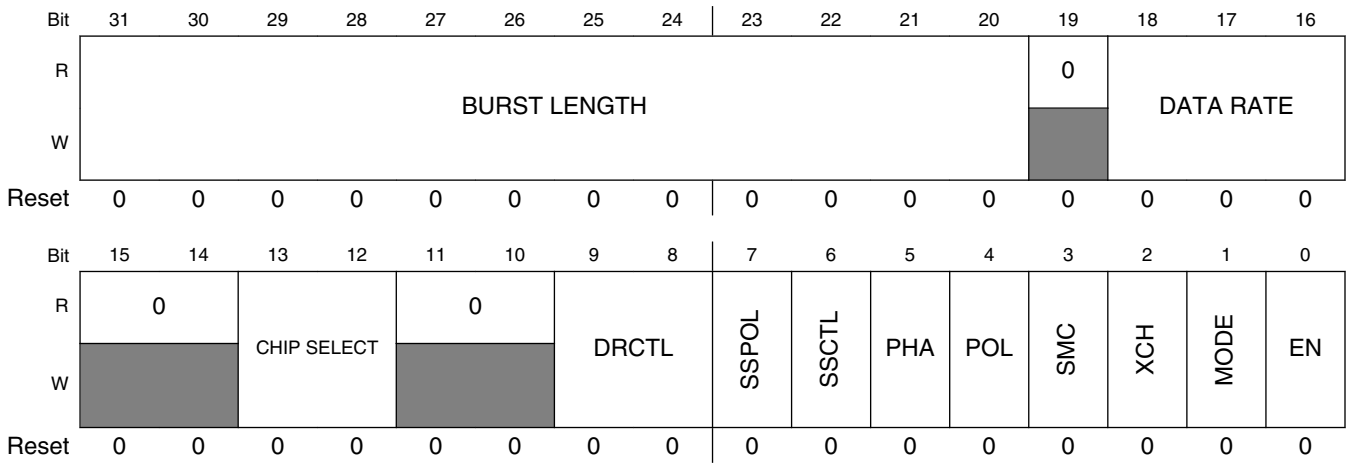
CSPI_TXDATA field descriptions

Field	Description
31–0 TXDATA	Transmit Data. This register holds the top word of data loaded into the FIFO. Data written to this register must be a word operation. The number of bits actually transmitted is determined by the BIT_COUNT field of the corresponding SPI Control register. If this field contains more bits than the number specified by BIT_COUNT, the extra bits are ignored. For example, to transfer 10 bits of data, a 32-bit word must be written to this register. Bits 9-0 are shifted out and bits 31-10 are ignored. When the CSPI is operating in Slave mode, zeros are shifted out when the FIFO is empty. Zeros are read when CSPI is disabled.

16.6.3 Control Register (CSPI_CONREG)

The Control Register (CSPI_CONREG) allows software to enable the CSPI, configure its operating modes, specify the divider value, phase, and polarity of the clock, configure the Chip Select (SS) and SPI_RDY (CSPI_CONREG[DRCTL]) control signal, and define the transfer length.

Address: CSPI_CONREG is 63FC_0000h base + 8h offset = 63FC_0008h



CSPI_CONREG field descriptions

Field	Description
31–20 BURST LENGTH	Burst Length. This field defines the length of a SPI burst to be transferred. The Chip Select (SS) will remain asserted until all bits in a SPI burst are shifted out. A maximum of 2^12 bits can be transferred in a single SPI burst. In master mode, it controls the number of bits per SPI burst. Since the shift register always loads 32-bit data from transmit FIFO, only the n least-significant (n = BURST LENGTH + 1) will be shifted out. The remaining bits will be ignored. In slave mode, only when SSCTL is cleared, this field will take effect in the transfer. Number of Valid Bits in a SPI burst. 0x000 A SPI burst contains the 1 LSB in a word. 0x001 A SPI burst contains the 2 LSB in a word. 0x002 A SPI burst contains the 3 LSB in a word. 0x01F A SPI burst contains all 32 bits in a word. 0x020 A SPI burst contains the 1 LSB in first word and all 32 bits in second word. 0x021 A SPI burst contains the 2 LSB in first word and all 32 bits in second word. 0xFFE A SPI burst contains the 31 LSB in first word and 2^7 -1 words. 0xFFF A SPI burst contains 2^7 words.
19 Reserved	This read-only field is reserved and always has the value zero. Reserved
18–16 DATA RATE	SPI Data Rate Control. This field selects the baud rate of the SCLK based on a division of the reference clock.

Table continues on the next page...

CSPI_CONREG field descriptions (continued)

Field	Description
	These bits allow the CSPI to synchronize with different external SPI devices. The max frequency is one quarter of the reference clock. The divide ratio is determined according to the following table using the equation: $2^{(n+2)}$. SPI Data Rate Control (Master Mode only) 000 Divide by 4. 001 Divide by 8. 010 Divide by 16. 011 Divide by 32. 100 Divide by 64. 101 Divide by 128. 110 Divide by 256. 111 Divide by 512.
15–14 Reserved	This read-only field is reserved and always has the value zero. Reserved
13–12 CHIP SELECT	CHIP SELECT. Select one of four external SPI Master/Slave Devices. In master mode, these two bits select the external slave devices by asserting the Chip Select (SSn) outputs. Only the selected Chip Select (SSn) signal can be active at a given time; the remaining three signals will be negated. Chip Select 00 Chip Select 0 (SS0) will be asserted. 01 Chip Select 1 (SS1) will be asserted. 10 Chip Select 2 (SS2) will be asserted. 11 Chip Select 3 (SS3) will be asserted.
11–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9–8 DRCTL	SPI Data Ready Control. This field selects the utilization of the $\overline{\text{SPI\_RDY}}$ signal in master mode. CSPI checks this field before it starts an SPI burst. 00 The $\overline{\text{SPI\_RDY}}$ signal is a don't care. 01 Burst will be triggered by the falling edge of the SPI_RDY signal (edge-triggered). 10 Burst will be triggered by a low level of the SPI_RDY signal (level-triggered). 11 Reserved.
7 SSPOL	SPI SS Polarity Select. In both Master and Slave modes, this bit selects the polarity of the Chip Select (SS) signal. 0 Active low. 1 Active high.
6 SSCTL	SPI SS Wave Form Select. In master mode, this bit controls the output wave form of the Chip Select (SS) signal when SMC is cleared. The SSCTL bit will be ignored if the SMC (Start Mode Control) bit is set. In slave mode, this bit controls when the SPI burst is completed. 0 Only one SPI burst will be transmitted. 1 Negate Chip Select (SS) signal between SPI bursts. Multiple SPI bursts will be transmitted. The SPI transfer will automatically stop when the TXFIFO is empty.

Table continues on the next page...

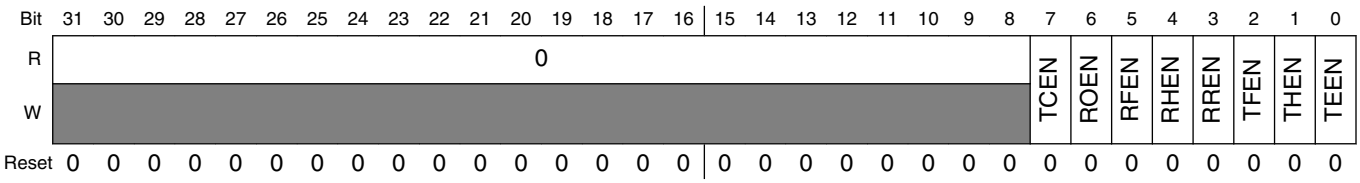
CSPI_CONREG field descriptions (continued)

Field	Description
	0 (slave mode) A SPI burst is completed when the number of bits received in the shift register is equal to BURST LENGTH + 1. Only n least-significant bits (n = BURST LENGTH[4:0] + 1) of the first received word are valid. All bits subsequent to the first received word in RXFIFO are valid. 1 (slave mode) A SPI burst is completed by the Chip Select (SS) signal edges. (SSPOL = 0: rising edge; SSPOL = 1: falling edge) The RXFIFO is advanced whenever a Chip Select (SS) signal edge is detected or the shift register contains 32-bits of valid data.
5 PHA	SPI Clock/Data Phase Control. This bit controls the clock/data phase relationship. See Typical Slave Mode for more information. 0 Phase 0 operation. 1 Phase 1 operation.
4 POL	SPI Clock Polarity Control. This bit controls the polarity of the SCLK signal. See Typical Slave Mode for more information. 0 Active high polarity (0 = Idle). 1 Active low polarity (1 = Idle).
3 SMC	Start Mode Control. This bit applies only when the block is configured in Master mode (MODE = 1). It controls how the CSPI starts a SPI burst, either through the SPI exchange bit, or immediately when the TXFIFO is written to. 0 SPI Exchange Bit (XCH) controls when a SPI burst can start. Setting the XCH bit will start a SPI burst or multiple bursts. This is controlled by the SPI SS Wave Form Select (SSCTL) bit. Refer to XCH and SSCTL bit descriptions. 1 Immediately starts a SPI burst when data is written in TXFIFO.
2 XCH	SPI Exchange Bit. This bit applies only when the block is configured in Master mode (MODE = 1). If the Start Mode Control (SMC) bit is cleared, writing a 1 to this bit starts one SPI burst or multiple SPI bursts according to the SPI SS Wave Form Select (SSCTL bit). The XCH bit remains set while either the data exchange is in progress, or when the CSPI is waiting for an active input if SPIRDY is enabled through DRCTL. This bit is cleared automatically when all data in the TXFIFO and the shift register has been shifted out. 0 Idle. 1 Initiates exchange (write) or busy (read).
1 MODE	SPI Function Mode Select. This bit selects the operating mode of the CSPI. 0 Slave mode. 1 Master mode.
0 EN	SPI Module Enable Control. This bit enables the CSPI. This bit must be set before writing to other registers or initiating an exchange. Writing zero to this bit disables the block and resets the internal logic with the exception of the CONREG. The block's internal clocks are gated off whenever the block is disabled. 0 Disable the block. 1 Enable the block.

16.6.4 Interrupt Control Register (CSPI_CSPI_INTREG)

The Interrupt Control Register (CSPI_INTREG) enables the generation of interrupts to the host processor. If the CSPI is disabled, this register reads zero.

Address: CSPI_CSPI_INTREG is 63FC_0000h base + Ch offset = 63FC_000Ch



CSPI_CSPI_INTREG field descriptions

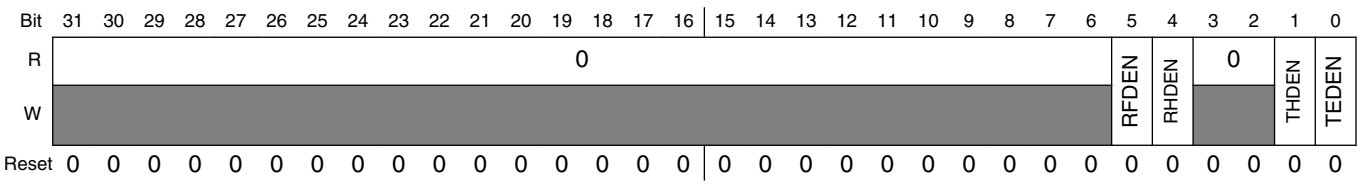
Field	Description
31–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 TCEN	Transfer Completed Interrupt enable. This bit enables the Transfer Completed Interrupt. 0 Disable 1 Enable
6 ROEN	RXFIFO Overflow Interrupt enable. This bit enables the RXFIFO Overflow Interrupt. 0 Disable 1 Enable
5 RFEN	RXFIFO Full Interrupt enable. This bit enables the RXFIFO Full Interrupt. 0 Disable 1 Enable
4 RHEN	RXFIFO Half Full Interrupt enable. This bit enables the RXFIFO Half Full Interrupt. 0 Disable 1 Enable
3 RREN	RXFIFO Ready Interrupt enable. This bit enables the RXFIFO Ready Interrupt. 0 Disable 1 Enable
2 TFEN	TXFIFO Full Interrupt enable. This bit enables the TXFIFO Full Interrupt. 0 Disable 1 Enable
1 THEN	TXFIFO Half Empty Interrupt enable. This bit enables the TXFIFO Half Empty Interrupt. 0 Disable 1 Enable
0 TEEN	TXFIFO Empty Interrupt enable. This bit enables the TXFIFO Empty Interrupt. 0 Disable 1 Enable

16.6.5 DMA Control Register (CSPI_CSPI_DMAREG)

The Direct Memory Access Control Register (CSPI_DMAREG) provides software a way to use an on-chip DMA controller for CSPI data. Internal DMA request signals enable direct data transfers between the CSPI FIFOs and system memory. The CSPI sends out DMA requests when the appropriate FIFO conditions are matched.

If the CSPI is disabled, this register is read as 0.

Address: CSPI_CSPI_DMAREG is 63FC_0000h base + 10h offset = 63FC_0010h



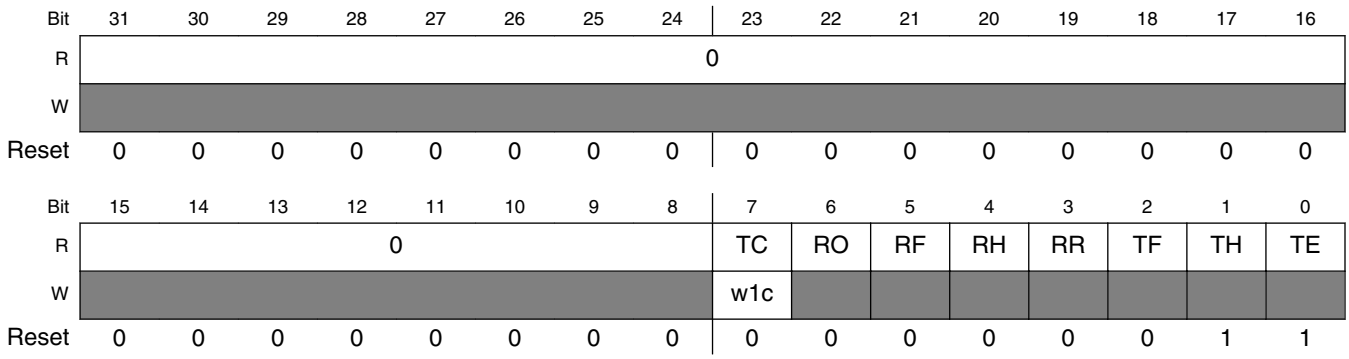
CSPI_CSPI_DMAREG field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5 RFDEN	RXFIFO Full DMA Request Enable. This bit enables/disables the RXFIFO Full DMA Request. 0 Disable 1 Enable
4 RHDEN	RXFIFO Half Full DMA Request Enable. This bit enables/disables the RXFIFO Half Full DMA Request. 0 Disable 1 Enable
3–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 THDEN	TXFIFO Half Empty DMA Request Enable. This bit enables/disables the TXFIFO Half Empty DMA Request. 0 Disable 1 Enable
0 TEDEN	TXFIFO Empty DMA Request Enable. This bit enables/disables the TXFIFO Empty DMA Request. 0 Disable 1 Enable

16.6.6 Status Register (CSPI_CSPI_STATREG)

The CSPI Status Register (CSPI_STATREG) reflects the status of the CSPI block's operating condition. If the CSPI is disabled, this register reads 0x0000_0003.

Address: CSPI_CSPI_STATREG is 63FC_0000h base + 14h offset = 63FC_0014h



CSPI_CSPI_STATREG field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 TC	Transfer Completed Status bit. Writing 1 to this bit clears it. 0 Transfer in progress. 1 Transfer completed.
6 RO	RXFIFO Overflow. When set, this bit indicates that RXFIFO has overflowed. 0 RXFIFO has no overflow. 1 RXFIFO has overflowed.
5 RF	RXFIFO Full. This bit is set when the RXFIFO is full. 0 Not Full. 1 Full.
4 RH	RXFIFO Half Full. This bit is set when the RXFIFO reaches half full. 0 Less than 4 words are stored in RXFIFO. 1 Four or more words are available in RXFIFO.
3 RR	RXFIFO Ready. This bit is set when one or more words are stored in the RXFIFO. 0 No valid data in RXFIFO. 1 More than 1 word in RXFIFO.
2 TF	TXFIFO Full. This bit is set when if the TXFIFO is full. 0 TXFIFO is not Full. 1 TXFIFO is Full.
1 TH	TXFIFO Half empty. This bit is set when the TXFIFO reaches half empty.

Table continues on the next page...

CSPI_CSPI_STATREG field descriptions (continued)

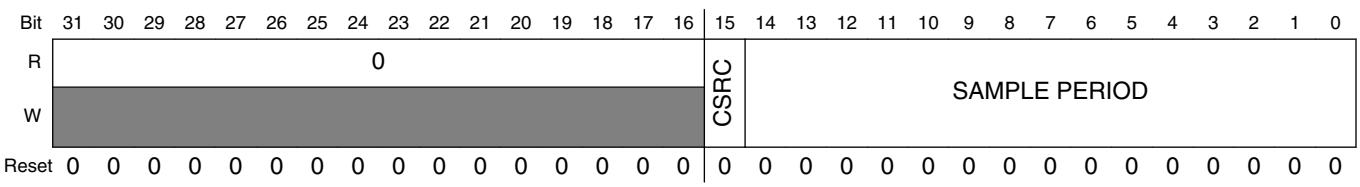
Field	Description
	0 TXFIFO holds more than 4 words. 1 TXFIFO holds 4 or fewer words.
0 TE	TXFIFO Empty. This bit is set if the TXFIFO is empty. 0 TXFIFO contains one or more words. 1 TXFIFO is empty.

16.6.7 Sample Period Control Register (CSPI_PERIODREG)

The Sample Period Control Register (CSPI_PERIODREG) provides software a way to insert delays (wait states) between consecutive SPI transfers. Control bits in this register select the clock source for the sample period counter and the delay count indicating the number of wait states to be inserted between data transfers.

The delay counts apply only when the block is configured in Master mode (CONREG[MODE] = 1).

Address: CSPI_PERIODREG is 63FC_0000h base + 18h offset = 63FC_0018h



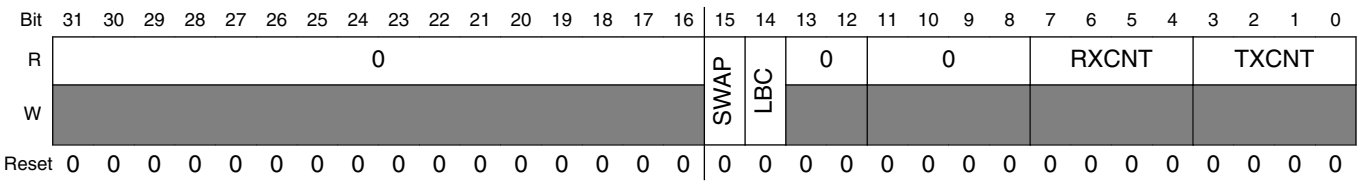
CSPI_PERIODREG field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 CSRC	Clock Source Control. This bit selects the clock source for the sample period counter. 0 SPI Clock (SCLK) 1 Low-Frequency Reference Clock (32.768 KHz)
14–0 SAMPLE PERIOD	Sample Period Control. These bits control the number of wait states to be inserted in data transfers. During the idle clocks, the state of the SS output will operate according to the SSCTL control field in the CSPI_CONREG register. 0x00000 wait states inserted 0x00011 wait state inserted 0x7FFE32766 wait states inserted 0x7FFF32767 wait states inserted

16.6.8 Test Control Register (CSPI_TESTREG)

The Test Control Register (CSPI_TESTREG) provides software a mechanism to internally connect the receive and transmit devices of the CSPI, swap the byte order for receive data, and monitor the contents of the receive and transmit FIFO.

Address: CSPI_TESTREG is 63FC_0000h base + 1Ch offset = 63FC_001Ch



CSPI_TESTREG field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 SWAP	Data Swap. This bit is used to swap data as it is read from the RXFIFO. When this bit is set, data read from RXFIFO is swapped. RXDATA[31:0] is swapped as follows: {RXDATA[7:0],RXDATA[15:8], RXDATA[23:16],RXDATA[31:24]} 0 Data read from RXFIFO is unchanged. 1 Data read from RXFIFO is swapped.
14 LBC	Loop Back Control. This bit is used in Master mode only. When this bit is set, the CSPI connects the transmitter and receiver sections internally, and the data shifted out from the most-significant bit of the shift register is looped back into the least-significant bit of the Shift register. In this way, a self-test of the complete transmit/receive path can be made. The output pins are not affected, and the input pins are ignored. 0 Not connected. 1 Transmitter and receiver sections internally connected for Loopback.
13–12 Reserved	This read-only field is reserved and always has the value zero. Reserved
11–8 Reserved	This read-only field is reserved and always has the value zero. Reserved, all bits should be ignored.
7–4 RXCNT	RXFIFO Counter. These bits indicate the number of words in RXFIFO. RXFIFO Counter 0000 0 word in RXFIFO 0001 1 word in RXFIFO 0111 7 words in RXFIFO 1000 8 words in RXFIFO
3–0 TXCNT	TXFIFO Counter. These bits indicate the number of words in TXFIFO. TXFIFO Counter

Table continues on the next page...

CSPI_TESTREG field descriptions (continued)

Field	Description
	0000 0 word in TXFIFO 0001 1 word in TXFIFO 0111 7 words in TXFIFO 1000 8 words in TXFIFO

Chapter 17
Data Co-Processor (DCP)

17.1 Overview

This chapter describes the data co-processor (DCP) block included on the i.MX50 and how to use it. It includes sections on memory copy functionality, Advanced Encryption Standard (AES), hashing, and arbitration. Sections on programming channel operations and example code are also provided. Programmable registers are described in DCP.

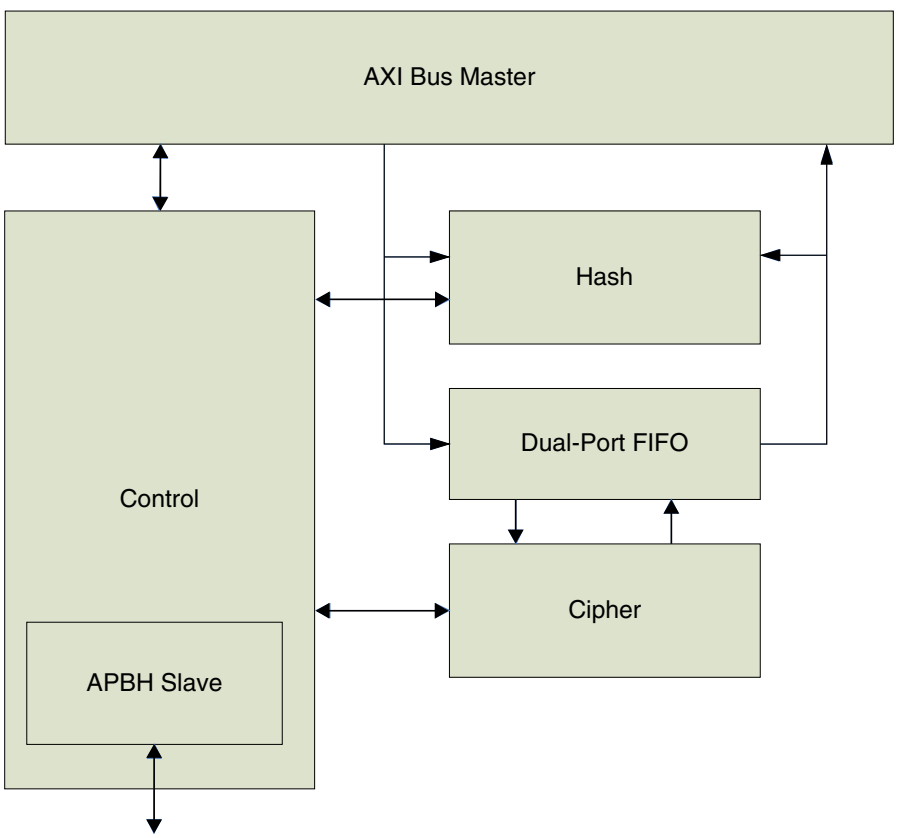


Figure 17-1. Data Co-Processor (DCP) Block Diagram

The DCP module provides support for general encryption and hashing functions typically used for security functions. Because its basic job is moving data from memory to memory, it also incorporates memory-copy (memcopy) function for both debugging and as a more efficient method of copying data between memory blocks than the DMA-based approach. The memcopy function also has a "blit" mode of operation where it can transfer a rectangular block of data to a video frame buffer.

The DCP has been designed to support a wide variety of encryption and hashing algorithms, and the control structures have been designed to allow flexibility in adding additional algorithms and modes in the future. It supports up to 16 encryption algorithms (for example DES, TDEA, AES-128, and so on) with 16 different modes of operation (ECB, CBC, and so on) as well as 16 hashing algorithms (for example MD5, SHA-1, SHA-2, and so on). While the DCP module has been designed to support numerous algorithms, only a subset may be implemented in any given implementation (see the Capability Register).

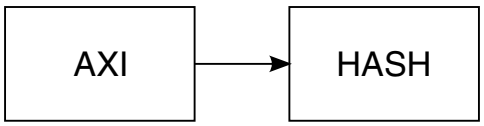
The DCP module processes data based on chained command structures written to system memory by software (in a manner similar to the DMA engine). The control block maintains registers to support four independent and concurrent chains, allowing software to virtualize access to the DCP block. Each command in a chain represents a work unit that the module will process to completion. At the end of each work unit, the control logic arbitrates among chains with outstanding commands and processes a command from that chain. Arbitration among the channels is round-robin, allowing all active channels equal access to the data engine. Each channel also supports a "high-priority" mode that allows it to have priority over the remaining channels. If multiple channels are selected as high-priority, the channel arbiter selects among the high-priority channels in round-robin fashion.

The data flow through the DCP module can be configured in one of five fashions, depending on the functionality activated by the control packet:

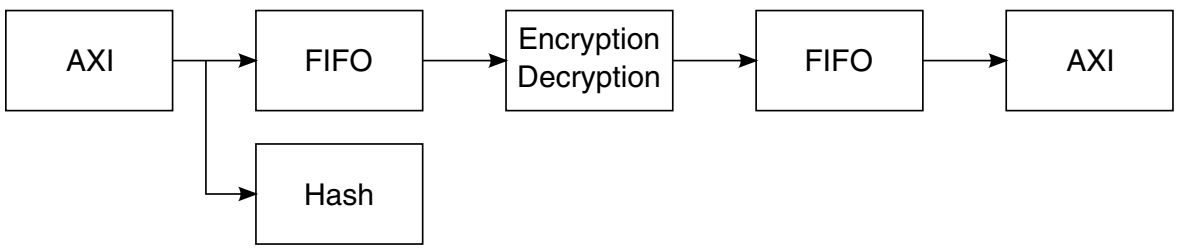
- **Memcopy/Blit Mode-**Data is moved unchanged from one memory buffer to another.

- **Encryption Only-**Data from source buffer is encrypted/decrypted into the destination buffer

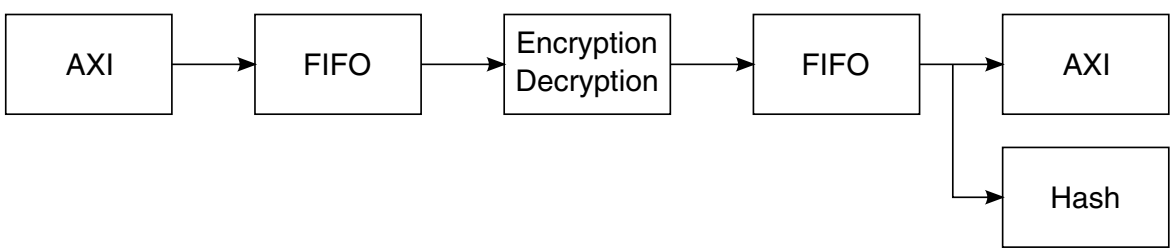
- **Hashing Only-**Data from source buffer is read, and a hash is generated.



- **Encryption and Input Hashing-**Data from source buffer is encrypted/decrypted into destination, and source buffer is hashed.



- **Encryption and Output Hashing-**Data from source buffer is encrypted/decrypted into destination, and output data is hashed.



17.1.1 DCP Limitations for Software

While the DCP module has been designed to be as flexible as possible, there are a few limitations to which software must adhere:

- Buffer sizes for all operations MUST be aligned to the natural size of the transfer algorithm used. Memcopy operations can transfer any number of bytes (one-byte granularity) and AES operations must be multiples of 16 bytes (four-word granularity). For all operations, if the byte count is not a word granularity, the hardware rounds up to the next word. Hashing is supported at a byte granularity.

- The DCP module supports buffer operations to any byte alignment, but performance will be improved if buffers are aligned to a four-byte boundary, since fetch/store operations can be performed without having to do byte operations to accommodate the misaligned addresses.
- Hash operations are limited to a 512 Mbyte buffer size. The hardware only implements a 32-bit hash length counter instead of the 64-bit counter supported by the SHA-1/SHA-256 algorithm (counter counts bits, not bytes, therefore a total of 512 Mbytes).
- For chained hashing operations (operations involving multiple descriptors), every descriptor except the last must have a byte count that is a 16-word multiple (granularity of the hash algorithm).
- Key values cannot be written while the AES block is active. This limitation exists because the key RAM is in use while AES is operational. Any writes from the APB cannot be held in wait states; therefore, the RAM must be accessible during key writes.
- The byte-swap controls can only be used with modulo-4 length buffers. For non-modulo-4 lengths, the final partial word will contain incorrect data. Any address alignment can be used with byte swapping, however.
- The word-swap controls are only useful with cipher operations, because the logic forms the 128-bit cipher data from four words from system memory. The word-swap controls are ignored for memcopy or hashing operations.
- DCP only supports writes to word boundaries to OCRAM. This is not required for DRAM MC address.

17.2 Operation

The top-level DCP module contains the AXI master, APB slave bus interface units, the main control block and FIFO, and any encryption or hashing blocks included with the design.

The controller manages the fetching of work blocks, the fetching/storing of context information when switching between chain pointers, and the managing of the data flow through the FIFO, SHA, and AES blocks. Data entering the block from the AXI master is placed in the FIFO for consumption by the cipher block. After the cipher module has finished its operation, data is placed back into the FIFO and stored back to memory via the AXI master. When hashing is enabled, the SHA block takes its inputs from the bus side of the FIFO to allow it to operate without waiting for the cipher block to complete. The APB slave provides all register controls and interfaces mainly with the control block.

17.2.1 Memory Copy, Blit, and Fill Functionality

In its most basic operation, the DCP supports moving unmodified data from one place in system memory to another. This functionality is referred to as "memcopy", because it operates only on memory and it copies data from one place to another. Typical uses for memcopy might be for fast virtual memory page moves or repositioning data blocks in memory. Memcopy buffers can be aligned to any memory address and can be of any length (byte granularity). For best performance, buffers should be word-aligned, although the DCP includes enhancements to improve performance for unaligned cases.

The DCP also has the ability to do basic "blit" operations that are typical in graphics operations. To specify a blit, the control packet must have the ENABLE_BLIT bit set in the packet control register. Blit source buffers must be contiguous. The output destination buffer for a blit operation is defined as a "M runs of N bytes" that define a rectangular region in a frame buffer. For blit operations, each line of the blit may consist of any number of bytes. After performing a "run", the DCP updates the destination pointer such that the next destination address falls on the pixel below the start of the previous run operation. This is done by incrementing the starting pointer by the frame buffer width, which is specified in the Control field.

In addition to being able to copy data within memory, the DCP also provides a "fill" operation, where source data comes not from another memory location, but from an internal register (the source buffer address in the control packet). This is done whenever the DCP_Control0[CONSTANT_FILL] flag is set in the packet control register (see Control1 Field). This feature may be used with memcopy to prefill memory with a specified value or during a blit operation to fill a rectangular region with a constant color.

17.2.2 Advanced Encryption Standard (AES)

The AES block implements a 128-bit key/data encryption/decryption block as defined by the National Institute of Standards and Technology (NIST) as US FIPS PUB 197, dated November 2001 (see references for specifications and toolkits).

The AES core used in the design was derived from the AES design available from OpenCores.org under a modified BSD license as described here: *http://www.opencores.org/projects.cgi/web/aes_core/overview. The license for this code is documented on page 60 for compliance.*

There are three variations of AES, each corresponding to the key size used: AES-128, AES-192 or AES-256. AES always operates on 128 bits of data at a time. Only the AES-128 algorithm is implemented at this time.

17.2.2.1 Key Storage

The DCP implements four SRAM-based keys that may be used by software to securely store keys on a semi-permanent basis. The keys may be written via the PIO interface by specifying a key index to specify which key to load and a subword pointer that indicates which word to write within the key. After a subword is written, the logic automatically increments the subword pointer so that software can program the higher-order words of the key without rewriting the key index. Keys written into the key storage are not readable.

To use a key in the key storage, the cipher descriptor packet should select the key by setting the DCP_Control1[KEY_SELECT] field in the Control1 descriptor field without setting the OTP_KEY or PAYLOAD_KEY fields in the Control0 register. See Control0 Field and Control1 Field.

17.2.2.2 AES OTP Key

After a system reset, the OTP controller reads the e-fuse devices and provides OTP key information to the DCP over a static 128-bit data bus input called otp_crypto_key_data. The DCP internally generates the control logic signals to capture this key into the key RAM. The system reset controller coordinates the deassertion of reset such that the OTP key is stable before the DCP reads it.

To use the OTP key, the descriptor packet should set the OTP_KEY field in the Control1 register (see Control1 Field).

17.2.2.3 Encryption Modes

The most basic form of encryption is the Electronic Code Book (ECB) mode. In this mode, the encryption output is a function only of the key and the plaintext, thus it can be visualized as a giant lookup table. While this provides a great deal of security, there are a few limitations. For instance, if the same plaintext appears more than once in a block of data, the same ciphertext will also appear. This can be very evident if the plaintext contains large blocks of constant data (0s for example) and can be used to formulate attacks against a key.

In order to make ciphers stronger, several modes of operation can be implemented around the basic ECB cipher to provide additional security. One such mode is CBC mode (Cipher Block Chaining), which takes the previous encrypted data and logically XORs it

with the next incoming plaintext before performing the encryption. During decryption, the process is reversed and the previous encrypted data is XORed with the decrypted ECB data to provide the plaintext again.

The AES engine supports handling the various modes of operation. The core AES block supports ECB mode, and other algorithms are handled in the wrapper around the encryption blocks. The DCP module supports Cipher Block Chaining (CBC), which chains the data blocks by XORing the previously encrypted data with the plaintext before encryption. Cipher block chaining encryption is illustrated in Figure 17-2.

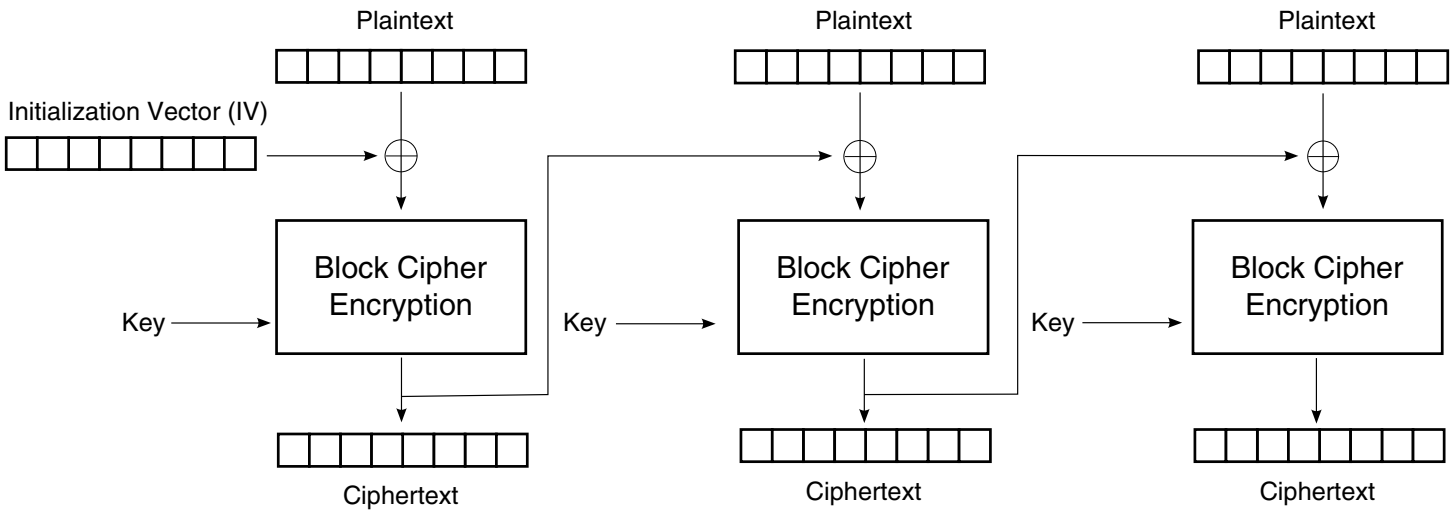


Figure 17-2. Cipher Block Chaining (CBC) Mode Encryption

Decryption (shown in Figure 17-3) works in a similar manner, where the cipher text is first decrypted and then XORed with the previous ciphertext. For the first encryption/ decryption operation, an initialization vector (IV) is used to seed the operation. The IV must be the same for both the encryption and decryption steps; otherwise, decrypted data will not match the original plaintext.

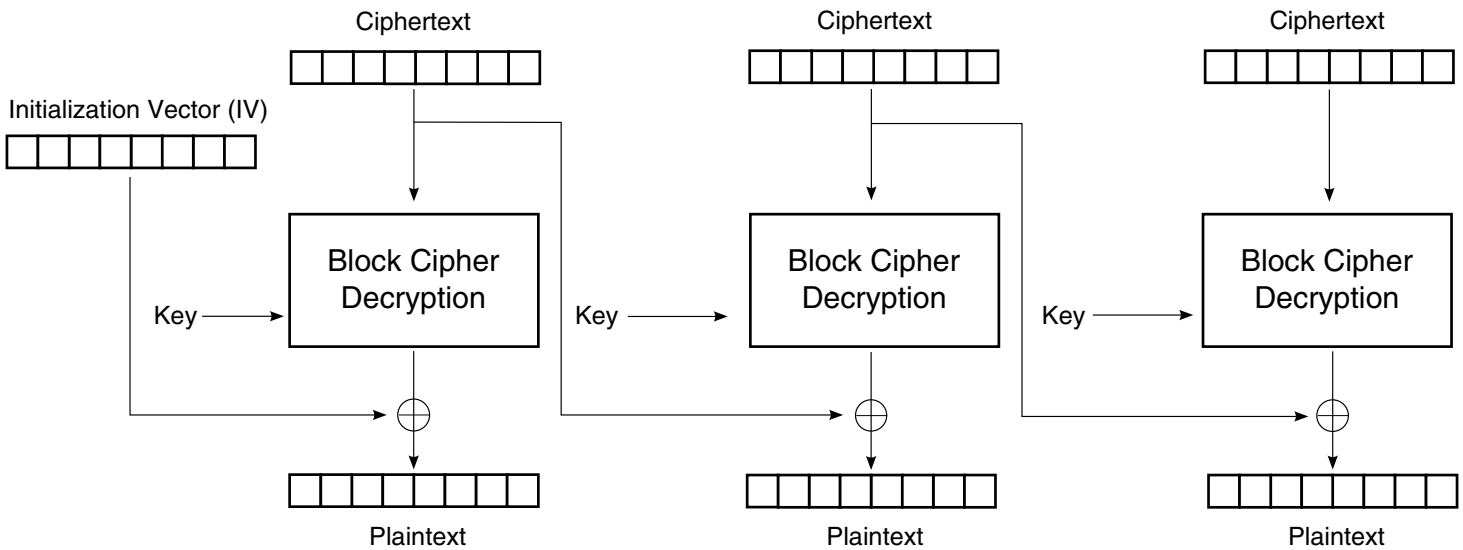


Figure 17-3. Cipher Block Chaining (CBC) Mode Decryption

17.2.3 Hashing

The hashing module implements the SHA-1 and SHA-256 hashing algorithms and a modified CRC-32 checksum algorithm. These algorithms produce a signature for a block of data that can be used to determine whether the data is intact.

The CRC-32 algorithm implements a 32-bit CRC algorithm similar to the one used by Ethernet and many other protocols. The CRC differs from the Unix cksum() function in three ways:

- The CRC is initialized as 0xFFFFFFFF instead of 0x00000000.
- Logic pads zeros to a 32-bit boundary for trailing bytes.
- Logic does not post-pend the file length.

The SHA-1 block implements a 160-bit hashing algorithm that operates on 512-bit (64-byte) blocks as defined by US FIPS PUB 180-1 in 1995. The SHA-256 mode implements a 256-bit hashing algorithm that operates on 512-bit (64-byte) blocks as defined by US FIPS PUB 180-2 in 2002. The purpose of the hashing module is to generate a unique signature for a block of data that can be used to validate the integrity of the data by comparing the resulting "digest" with the original digest.

Results from hash operations are written to the beginning of the payload for the descriptor. The DCP also has the ability to check the resulting hash against a value in the payload and issue an interrupt if a mismatch occurs.

17.2.4 One Time Programmable (OTP) Key

After a system reset, the One Time Programmable (OTP) controller will read the e-fuse devices and provide OTP key information to the DCP through the 128-bit wide otp_crypto_key_data input data bus and the 64 bit wide otp_unique_key_data unique key input that come directly from the OTP controller. These bus input values are held constant and originate from the eFuse shadow registers after reset. (Note that the shadow registers for these bits can be changed in the OTP block if they are unlocked.) They will only be loaded by the DCP once after the deassertion of system reset. After reset, the DCP automatically loads these wide keys 32 bits at the time into the internal key RAM.

While OTP key is normally not available to read operations, the module will allow the key to be read if the otp_crypto_key_ren (read-enable) input is active (high). This allows verification of the key after it has been programmed into the eFuse device. After programming has been validated, another fuse will be set to disable the read capability.

The OTP key (crypto key) may be selected using the DCP_Control1[OTP_KEY] bit in the control field of the packet descriptor or by using key select 0xFF in the CTRL1 field of the descriptor. DCP also supports a second hardware key called the UNIQUE_KEY which is generated from the OTP KEY and key modifier bits from other OTP fuse fields. This key is unique to the device and may be used for encrypting private data stored on the NAND. This key may be selected by writing 0xFE to the KEY_SELECT field in the CTRL1 packet data.

17.2.5 Managing DCP Channel Arbitration and Performance

The DCP can have four channels all competition for DCP resources to complete their operations. Depending on the situation, critical operations may need to be prioritized above less important operations to ensure smooth system operation. To help software achieve this goal, the DCP implements a programmable arbiter for internal DCP operations and provides "recovery" timers on each channel to throttle channel activity.

17.2.5.1 DCP Arbitration

The DCP implements a multi-tiered arbitration policy that allows software a lot of flexibility in scheduling DCP operations. Figure 17-4 illustrates the arbitration levels and where each channel fit into the arbitration scheme.

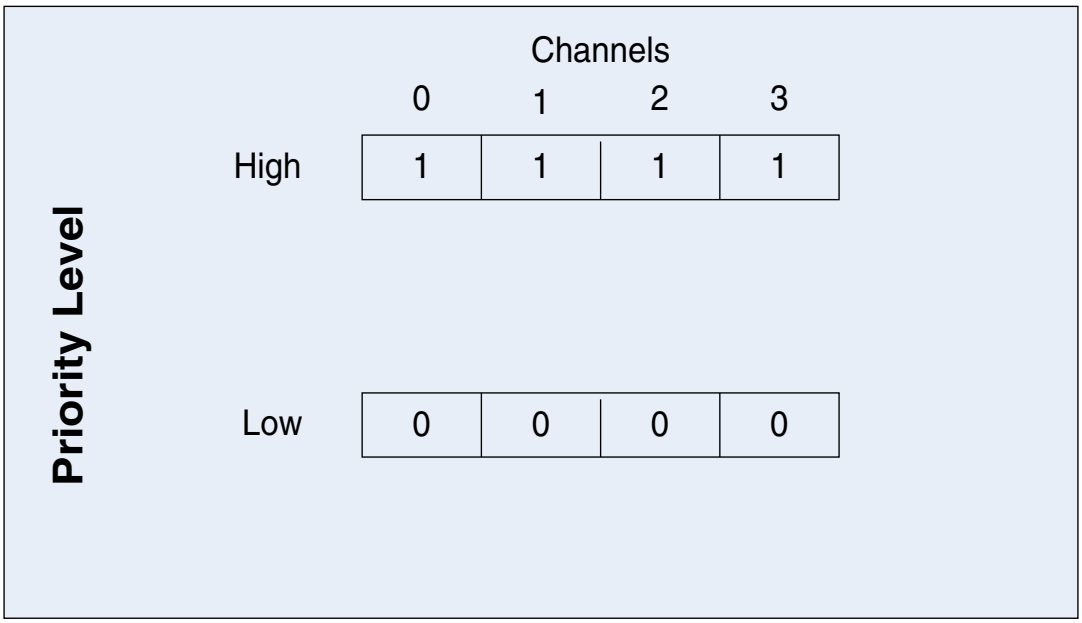


Figure 17-4. DCP Arbitration

Each channel can be programmed as being in either the high-priority or low-priority arbitration pool, depending on the setting in the HIGH_PRIORITY_CHANNEL field of the DCP_CHANNELCTRL register. When the corresponding bit is programmed as a 1, the channel arbitrates in the high-priority pool; otherwise it arbitrates in the low-priority pool. Once a channel has been selected, it completes one packet and then the arbiter re-arbitrates. The channel that just completed participates in the new arbitration round.

17.2.5.2 Channel Recovery Timers

Each channel also contains a channel recovery timer in its DCP_CHnOPTS register. The purpose of the recovery timer is to keep the channel inactive for a period of time after it completes an operation. This capability could be used for a high-priority channel to ensure that at least some lower-priority requests get serviced between packets or to simply allow more timeslices for other channels to perform operations. The value programmed into the recovery timers register delays the channel from operations until 16 times the programmed value. Any non-zero value should prevent the channel from participating in the next arbitration cycle.

17.2.6 Programming Channel Operations

The control logic block maintains the channel pointers and manages arbitration and context switching between the different channels. It also manages the fetching of work packets and data fetch/store operations from the AXI master interface and coordinates the actions of the hashing and encryption blocks.

The control logic maintains four channels that allow software to effectively create four independent work sets for the DCP module. Software can construct chained control packets in memory that describe encryption/hashing/memcopy operations to the hardware unit. The address for this first control packet can be written to one of the four virtual channels and enabled. When one or more of the channels is enabled, the controller fetches the control packet pointed to by that channel and initiate data fetches from the source buffer. Data is then processed as described in the control packet and stored back to system memory.

Once the processing for a control packet is complete, the controller writes completion status information back to the control packet, and optionally stores relevant state information to the context buffer. If the control packet specifies a subsequent control packet, the channel's pointer is updated to the address for the next packet and an optional interrupt can be issued to the processor.

At this point, the DCP module arbitrates among the virtual channels for the next operation and processes its control packet. If a subsequent operation is continued from a previous operation, the controller automatically loads the context from the previous session into the working registers before resuming operation for that channel.

17.2.6.1 Virtual Channels

The DCP module processes data via work packets stored in memory. Each of the channels contains a pointer to the current work packet and enough control logic to determine whether the channel is active and to provide status to the processor. Each channel also provides a recovery timer to help throttle processing by a particular channel. After processing a packet, the channel enables its 16-bit recovery timer (if the recovery time is set to a non-zero value). The channel will not become active again until its recovery timer has expired. The recovery timer timebase is 16 HCLK cycles, so the timer acts as a 20-bit timer with the bottom four bits implicitly tied to 0. This provides an effective range of zero to 2^{20-1} clocks or 0 ns to 7.8 ms at 133 MHz.

A channel is activated any time its semaphore is non-zero and its recovery timer is cleared. The semaphore can be incremented by software to indicate that the chain pointer has been loaded with a valid pointer. As the hardware completes the work packets, it decrements the semaphore if the Decrement Semaphore flag in the Control 0 field set.

Work packets may be chained together using the CHAIN or CHAIN_CONTIGUOUS bits in the Control0 field, which causes the channel to automatically update the work packet pointer to the value in the NEXT_COMMAND_ADDRESS field at the end of the current work packet.

17.2.6.2 Context Switching

The control logic maintains four virtual channels that allow the DCP block to time-multiplex encryption, hashing, and memcopy operations, it must also retain state information when changing channels so that when a channel is resumed, it can resume the operation from where it left off. This process is called context-switching.

To minimize the number of registers used in the design, the controller saves context information from each channel into a private context area in system memory. When initializing the DCP module, software must allocate memory for the context buffer and write the address into the Context Buffer Pointer register. As the DCP module processes packets, it saves the context information for each channel to the buffer after completing each control packet. When the channel is subsequently activated, the DCP module's internal registers are then reloaded with the proper context before resuming the operation.

Each channel reserves one-fourth of the context buffer area for its context storage. The context buffer consumes 208 bytes of system memory and is formatted as shown in Table 17-1.

Table 17-1. DCP Context Buffer Layout

Range	Channel	Data	Size
0x00-0x0C	3	Cipher Context	16 bytes
0x10-0x30	-	Hash Context	36 bytes
0x34-0x40	2	Cipher Context	16 bytes
0x44-0x64	-	Hash Context	36 bytes
0x68-0x74	1	Cipher Context	16 bytes
0x78-0x98	-	Hash Context	36 bytes
0x9C-0xA8	0	Cipher Context	16 bytes
0xAC-0xCC	-	Hash Context	36 bytes

The control logic writes to the context buffer only if the function is being used. This effectively means that the cipher context is stored for CBC encryption/decryption operations only, and the hash context is written only if SHA-1/SHA-256 is utilized. If neither of these modes are used for a given channel, the memory for the context buffer need not be allocated by software.

Since channel 0 is likely to be used for VMI in an SDRAM-based system-to-page data from the SDRAM to on-chip SRAM, the buffer allocation table has been organized so that the *highest* numbered channel uses the *lowest* area in the context buffer. For this reason, software should allocate the higher numbered channels for encryption/hashing operations and the lower numbered channels for memcopy operations to reduce the size of the context buffer.

If only a single channel is used for CBC mode operations or hashing operations, the controller provides a bit in the control register to disable context switching. In this scenario, context switching is not required, because other channels will not corrupt the state of the hashing or cipher modes. Thus, when the channel resumes, a context load is not required.

Additionally, the control logic monitors the use of CBC/hashing, so that a context reload is not done if the previous channel resumes an operation without an intermediate operation from another channel.

17.2.6.3 Working with Semaphores

Each channel has a semaphore register to indicate that control packets are ready to be processed. Several techniques can be used when programming the semaphores to control the execution of packets within a channel. The channel will continue to execute packets as long as the semaphore is non-zero, a chain bit is set in the descriptor, and no error has occurred.

- Software can write the number of pending packets into the semaphore register with the Decrement Semaphore bit in each control packet set. In this scenario, the channel simply decrements the semaphore for each packet set and terminates at the end of the packet chain. The benefit of this method is that software can easily determine how many packets have executed by reading the semaphore status register.
- Software can create a packet chain with the Decrement Semaphore bit set only in the last packet. In this case, software would write a 1 to the semaphore register to start the chains and the DCP will terminate after executing the last packet.
- Software can create a packet chain with the Decrement Semaphore bit set for each packet and write either a 1 or a number less than the number of packets to the semaphore register. This can be useful when debugging, because it allows the channel to execute only a portion of the packets and software can inspect intermediate values before restarting the channel again.

If an error occurs, the channel issues an interrupt and clears the semaphore register. The channel does not perform any further operations until the error bits in the status register have been cleared. Software can manually clear a non-zero semaphore by writing 0xFF to the CLR (clear) address of the semaphore register.

17.2.6.4 Work Packet Structure

Work packets for the DCP module are created in memory by the processor. Each work packet includes all information required for the hardware to process the data. The general structure of the packet is shown in Figure 17-5.

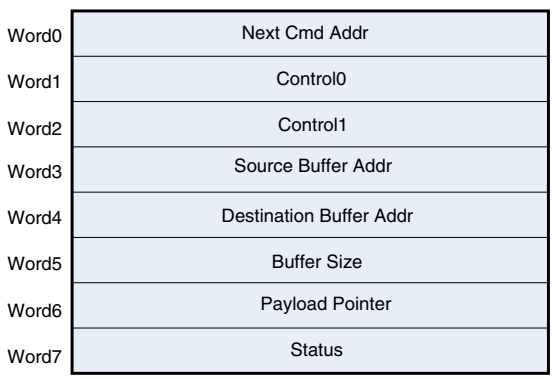


Figure 17-5. DCP Work Packet Structure

For some operations, additional information is required to process the data in the packet. This additional information is provided in the variable-sized payload buffer, which can be found at the address specified in the payload pointer field. When encryption is specified, the payload may include the encryption key (if the key selected resides in the packet), an initialization vector (for modes of operation such as CBC), and expected hash values when data hashing is indicated. The hardware can automatically compare the expected hash with the actual hash and interrupt the processor only on a mismatch. The payload area is used by software to store the calculated hash value at the end of hashing operations (when the HASH_TERM control bit in DCP_Control0 is set).

17.2.6.4.1 Next Command Address Field

The NEXT_COMMAND_ADDRESS field (as shown in Table 17-2) is used to point to the next work packet in the chain. This field is loaded into the channel's command pointer after the current packet has completed processing if the CHAIN bit in the CONTROL0

field (see Table 17-3) is set. The Next Command Address field should be programmed to a non-zero value when the CHAIN bit is set; otherwise, the channel will flag an invalid setup error.

Table 17-2. DCP Next Command Address Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															

17.2.6.4.2 Control0 Field

The main functions of the DCP module are enabled with the ENABLE_MEMCOPY, ENABLE_BLIT, ENABLE_CIPHER, and ENABLE_HASH bits from the Control0 field in the work packet. The combinations of these bits determine the function performed by the DCP. Table 17-4 summarizes the function performed for each combination.

Table 17-3. DCP Control0 Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TAG								OUTPUT_WORDSWAP	OUTPUT_BYTESWAP	INPUT_WORDSWAP	INPUT_BYTESWAP	KEY_WORDSWAP	KEY_BYTESWA	TEST_SEMA_IRQ	CONSTANT_FILL	HASH_OUTPUT	HASH_CHECK	HASH_TERM	HASH_INIT	PAYLOAD_KEY	OTP_KEY	CIPHER_INIT	CIPHER_ENCRYP	ENABLE_BLIT	ENABLE_HASH	ENABLE_CIPHE	ENABLE_MEMCOPY	CHAIN_CONTINUOUS	CHAIN	DECR_SEMAPHOR	INTERRUPT_ENABLE

Table 17-4. DCP Function Enable Bits

Hash	Cipher	Blit	Memcopy	Description
0	0	0	X	Simple memcopy operation.
0	0	1	0	Blit operation.
0	1	0	0	Simple encrypt/decrypt operation.
1	0	0	0	Simple hash. Only reads performed.
1	0	0	1	Memcopy and hash operation.
1	1	0	0	Hash with encryption/decryption.
All Others				Invalid setup.

The CHAIN bit should be set if the NEXT_COMMAND_ADDRESS field has a valid pointer to the next work packet. When set, this bit causes the channel to update its pointer to the next packet. The CHAIN_CONTINUOUS bit is similar, but it indicates that the

next packet follows immediately after the current packet. This allows software to generate templates of operations without regard to the actual physical addresses used, which makes programming easier, especially in a virtual memory environment.

If the INTERRUPT_ENABLE bit is set, the channel generates an interrupt to the processor after completing the work for this packet. When the interrupt is issued, the packet will have been completely processed, including the update of the status/payload fields in the work packet.

Each channel contains an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore in non-zero, the channel is ready to run and process commands. Whenever a command finishes its operation, it checks the DECR_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the idle state and remains there until the semaphore is incremented by software. When the semaphore goes to non-zero and the channel is in its idle state, it then uses the value in the NEXT_COMMAND_ADDRESS field to begin processing.

If the ENABLE_CIPHER bit is set, the data is encrypted or decrypted based on the CIPHER_ENCRYPT bit using the key/encryption settings from the Control1 field. The OTP_KEY and PAYLOAD_KEY indicate the source of the keys for the operation. If the OTP_KEY value is set, the KEY_SELECT field from the Control1 register indicates which OTP key is to be used. If the PAYLOAD_KEY bit is set, the first entry in the payload is the key to be used for the operation. If neither bit is set, the KEY_SELECT field indicates an index in the key RAM that contains the key to be used. (For cases when both the OTP_KEY and PAYLOAD_KEY bits are set, the PAYLOAD_KEY takes precedence.)

The HASH_ENABLE enables hashing of the data with the HASH_OUTPUT bit controlling whether the input data (HASH_OUTPUT=0) or output data (HASH_OUTPUT=1) is hashed. In addition, the hardware can be programmed to automatically check the hashed data if the HASH_CHECK bit is set. If the hash does not match, an interrupt is generated and the channel terminates operation on the current chain. The hashing algorithms also require two additional fields in order to operate properly. The HASH_INIT bit should be set for the first block in a hashing pass to properly initialize the hashing function. The HASH_TERM bit must be set on the final block of a hash to notify the hardware that the hashing operation is complete so that it can properly pad the tail of the buffer according to the hashing algorithm. This flag also triggers the write-back of the hash output to the work packet's payload area.

The CONSTANT_FILL flag may be used for both memcopy and blit operations. When this is set, the source address field is used instead as a constant that is written to all locations in the output buffer.

The WORDSWAP and BYTESWAP bits enable muxes around the FIFO to swap data to handle little-endian and big-endian storage of data in system memory. When these bits are cleared, data is assumed to be in little-endian format. When all bits are set, data is assumed to be in big-endian format.

The TAG field is used to identify the work packet and the associated completion status. As each packet is completed, the channel provides the status and tag information for the last packet processed.

17.2.6.4.3 Control1 Field

The Control1 field (shown in Table 17-5) provides additional values used when hashing or encrypting/decrypting data. For blit operations, this field indicates the number of bytes in a frame buffer line, which is used to calculate the address for successive lines in each blit operation.

Table 17-5. DCP Control1 Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CIPHER_CONFIG												HASH_ SELECT				KEY_SELECT								CIPHER_ MODE				CIPHER_ SELECT			
																FRAMEBUFFER_LENGTH (in bytes, blit mode)															

The CIPHER_SELECT field selects from one of sixteen possible encryption algorithms (0=AES128). The CIPHER_MODE selects the mode of operation for that cipher (0=ECB, 1=CBC).

The KEY_SELECT field allows key selection for one of several sources. Keys can be included in the packet payload or may come from OTP or write-only registers.

The HASH_SELECT field selects a hashing algorithm for the operation (0=SHA-1, 1=CRC32, 2=SHA-256).

The CIPHER_CONFIG field provides optional configuration information for the selected cipher. An example would be a key length for the RC4 algorithm.

17.2.6.4.4 Source Buffer

The SOURCE_BUFFER pointer (shown in Table 17-6) specifies the location of the source buffer in memory. The buffer may reside at any byte alignment and should refer to an on-chip SRAM of off-chip SDRAM location. For optimal performance, buffers should be word aligned. When the CONSTANT_FILL flag is set in the Control0 field, the value in this field is used as the data written to all destination buffer locations.

Table 17-6. DCP Source Buffer Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SOURCE_BUFFER																															
CONSTANT (CONSTANT_FILL mode)																															

17.2.6.4.5 Destination Buffer

The DESTINATION_BUFFER (shown in Table 17-7) specifies the location of the destination buffer in on-chip SRAM or off-chip SDRAM memory and may be set to any byte alignment. For in-place encryption, the destination buffer and source buffer should be the same value. For optimal performance, the buffer location should be word-aligned.

Table 17-7. DCP Destination Buffer Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DESTINATION_BUFFER																															

17.2.6.4.6 Buffer Size Field

The BUFFER_SIZE field (shown in Table 17-8) indicates the size of the buffers for memcopy, encryption, and hashing modes. For memcopy and hashing operations, the value may be any number of bytes (byte granularity), and for encryption modes, the length must be a multiple of the selected encryption algorithm's natural data size (16 bytes for AES).

Table 17-8. DCP Buffer Size Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NUMBER_LINES (blit mode)																BLIT_WIDTH (in bytes, blit mode)															
BUFFER_SIZE (in bytes, memcopy, encryption, hashing modes)																															

For blit operations, the buffer size field is split into two portions, a BLIT_WIDTH value, which specifies the number of bytes in each "line" of the blit operation, and a NUMBER_LINES value, which specifies how many vertical rows of pixels are in the image buffer.

17.2.6.4.7 Payload Pointer

Some operations require additional control values that are stored in a Payload Buffer (shown in Table 17-9), which is pointed to by this field. After the DCP reads the control packet, it examines the Control0 register and determines whether any payload information is required. If so, the DCP loads the payload from the address specified in this field. (See Payload for more details.)

Table 17-9. DCP Payload Buffer Pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PAYLOAD_POINTER																															

The payload area is also written to by the DCP at the completion of a hash operation (when the HASH_TERM) bit is set. Software must allocate the appropriate amount of storage (20 bytes for SHA-1 and 4 bytes for CRC32) in the payload or risk having the DCP write to an unallocated address.

17.2.6.4.8 Status

After the DCP engine has completed processing of a descriptor, it writes the packet status (shown in Table 17-10) back to the descriptor In the Status field. The packet status is the value of the channel status register at the time the packet completed processing.

Table 17-10. DCP Status Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TAG								ERROR_CODE																		ERROR_DS	ERROR_SR	ERROR_PACKET	ERROR_SETU	HASH_MISMATCH	COMPLETE

For various error conditions, the DCP encodes additional error information in the ERROR_CODE field. See DCP for more details on assignments of error codes. For any completion codes besides the COMPLETE flag, the channel suspends processing of the command chain until the error status values are cleared in the channel status register.

The format for this field is the same as for the channel status register, with the exception that the COMPLETE bit is written to the payload status but is not present in the channel status register.

17.2.6.4.9 Payload

The payload is a variable-length field that is used to provide key, initialization values, and expected results from hashing operations. The payload may consist of the data fields listed in Table 17-11.

Table 17-11. DCP Payload Field

Field Name	Size	Description	Condition
Cipher Key	16 bytes	AES key	Cipher enable with PAYLOAD_KEY
Cipher IV	16 bytes	CBC initialization vector	Cipher enable with CBC mode.
Hash Check	20 bytes	Hash completion value	Hash enabled with HASH_TERM fields set.

If fields are not used, they do not appear in the payload and the other payload values move upwards in the payload area. For instance, if only hashing is used, then the HASH_CHECK value would appear at offset 0 in the payload area. Table 17-12 should be used by software to determine the amount of payload to allocate and initialize.

Table 17-12. DCP Payload Allocation by Software

Control Bits Present			Payload Size
Hash_Check	Cipher_Init	Payload_Key	
0	0	0	0 words / 0 bytes
0	0	1	4 words / 16 bytes
0	1	0	4 words / 16 bytes
0	1	1	8 words / 32 bytes
1	0	0	SHA-1: 5 words / 20 bytes SHA-2: 8 words / 32 bytes
1	0	1	SHA-1: 9 words / 36 bytes SHA-2:12 words / 48 bytes
1	1	0	SHA-1: 9 words / 36 bytes SHA-2: 12 words / 48 bytes
1	1	1	SHA-1: 13 words / 52 bytes SHA-2: 16 words / 64 bytes

For hashing operations, the DCP module writes the final hash value back to the start of the payload area for descriptors with the HASH_TERM bit set in the control packet. It is important that software allocate the required payload space, even though it is not required to set up the payload for control purposes.

17.2.7 Programming DCP Functions

17.2.7.1 Basic Memory Copy Programming Example

To perform a basic memcopy operation, only a single descriptor is required. The DCP simply copies data from the source to the destination buffer. This process is illustrated in Figure 17-6.

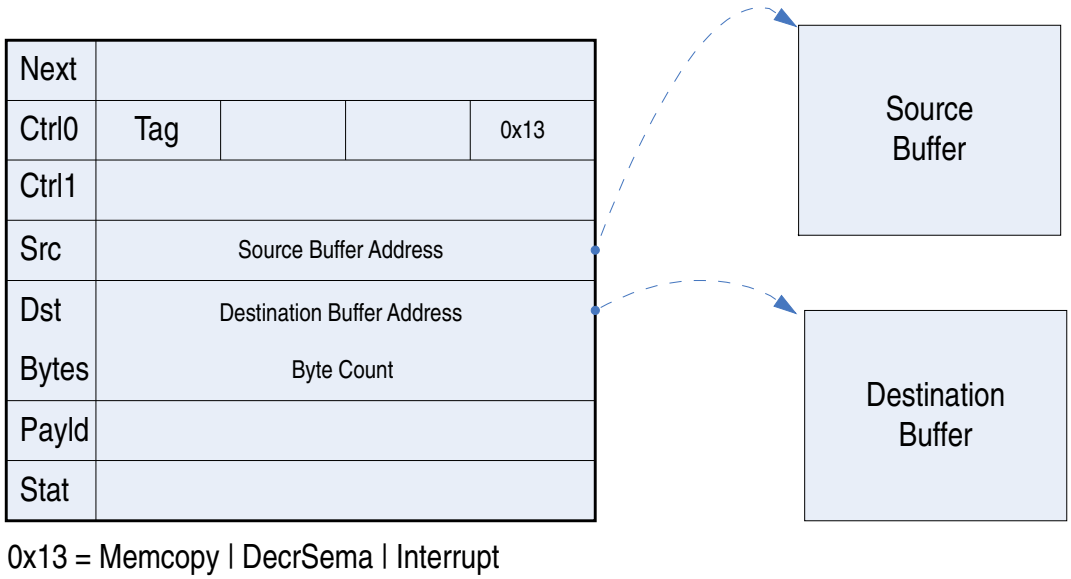


Figure 17-6. Basic Memory Copy Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32              *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32              *src,
                     *dst,
                      buf_size,
                     *payload,
                      stat;
}  DCP_DESCRIPTOR;
  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer, *dstbuffer;
  // set up control packet
  dcp1.next = 0;                    // single packet in chain
  dcp1.ctrl0.U = 0;                 // clear ctrl0 field
  dcp1.ctrl0.B.ENABLE_MEMCOPY = 1; // enable memcopy
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;       // interrupt
  dcp1.ctrl1.U = 0;                 // clear ctrl1
  dcp1.src = srcbuffer;             // source buffer
  dcp1.dst = dstbuffer;             // destination buffer
  dcp1.buf_size = 512;              // 512 bytes
  dcp1.payload = NULL;              // not required
  dcp1.status = 0;                  // clear status
  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);     // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);          // increment semaphore by 1
  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
```

```
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);
</code>
```

17.2.7.2 Basic Hash Operation Programming Example

To perform a basic hash operation, only a single descriptor is required. The DCP simply reads data from the source buffer and computes the hash value on the contents. This process is illustrated in Figure 17-7.

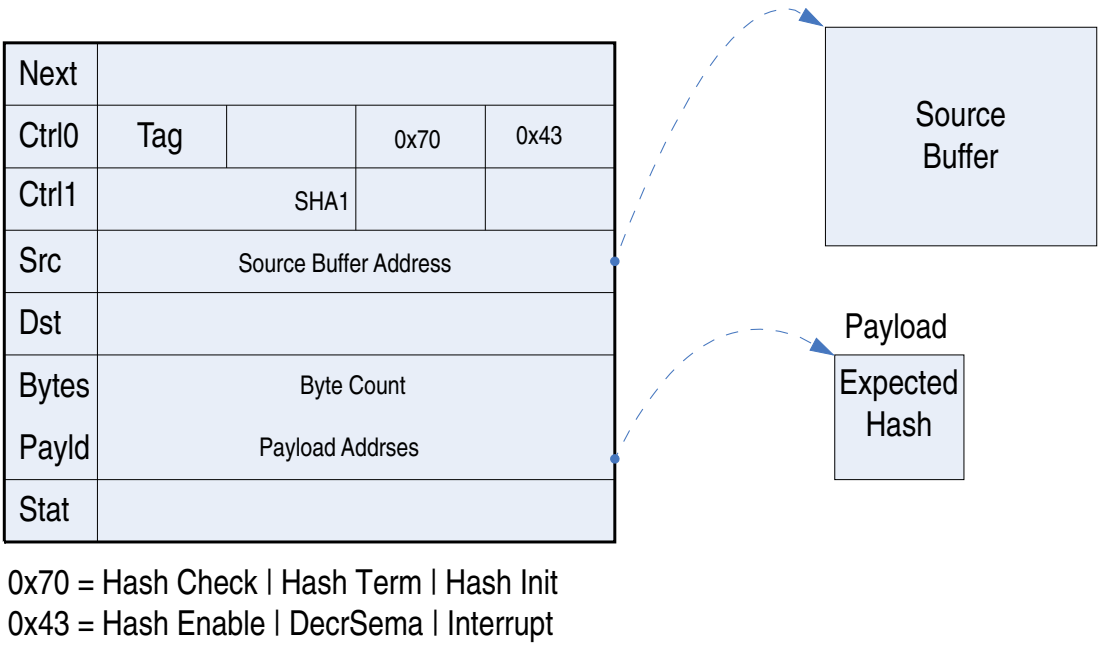


Figure 17-7. Basic Hash Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32              *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32              *src,
                     *dst,
                      buf_size,
                     *payload,
                      stat;
}  DCP_DESCRIPTOR;
  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer;
  u32 payload[5];
  // set up expected hash check value
  payload[0]=0x01234567;
  payload[1]=0x89ABCDEF;
  payload[2]=0x00112233;
  payload[3]=0x44556677;
  payload[4]=0x8899aabb;
```

```
  // set up control packet
  dcp1.next = 0;                    // single packet in chain
  dcp1.ctrl0.U = 0;                 // clear ctrl0 field
  dcp1.ctrl0.B.HASH_CHECK = 1;      // check hash when complete
  dcp1.ctrl0.B.HASH_INIT = 1;       // initialize hash with this block
  dcp1.ctrl0.B.HASH_TERM = 1;       // terminate hash with this block
  dcp1.ctrl0.B.ENABLE_HASH = 1;     // enable hash
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;       // interrupt
  dcp1.ctrl1.U = 0;                 // clear ctrl1
  dcp1.src = srcbuffer;             // source buffer
  dcp1.dst = 0;                     // no destination buffer
  dcp1.buf_size = 512;              // 512 bytes
  dcp1.payload = payload;           // holds expected hash value
  dcp1.status = 0;                  // clear status
  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);     // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);          // increment semaphore by 1
  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);
</code>
```

17.2.7.3 Basic Cipher Operation Programming Example

To perform a basic cipher operation, only a single descriptor is required. The DCP reads data from the source buffer, encrypts it, and writes it to the destination buffer. For this example, the key is provided in the payload and the algorithm uses the AES CBC mode of operation. This requires a payload with both the key and CBC initialization value. This process is illustrated in Figure 17-8.

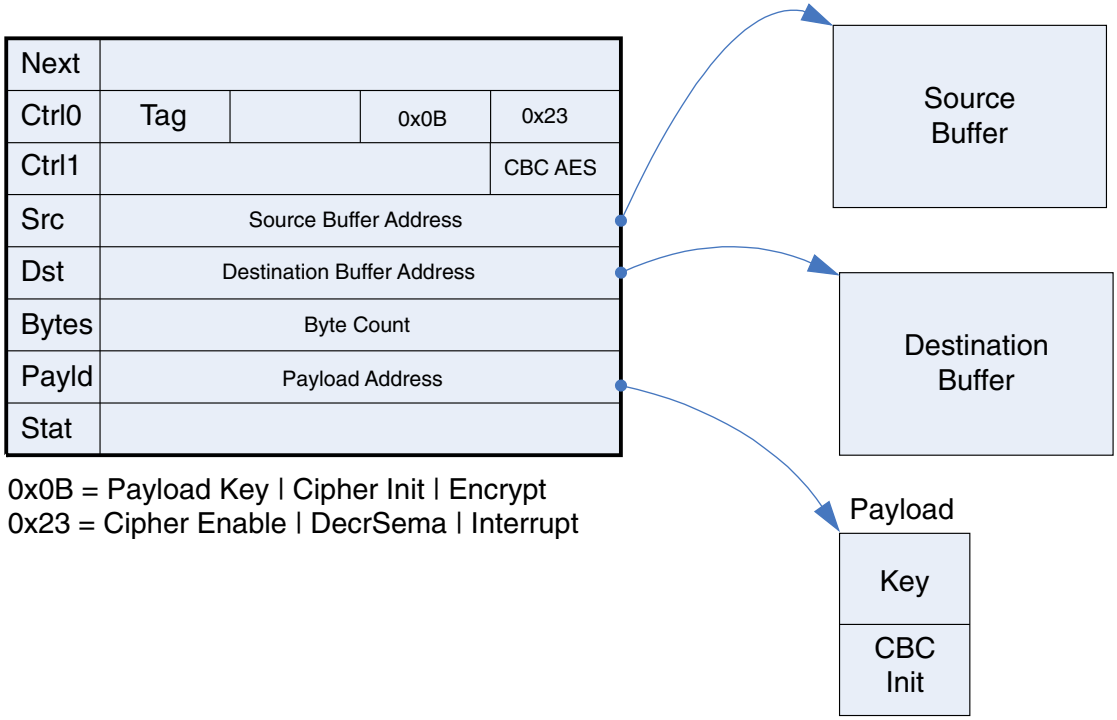


Figure 17-8. Basic Cipher Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32              *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32              *src,
                     *dst,
                      buf_size,
                     *payload,
                      stat;
}  DCP_DESCRIPTOR;
  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer;
  u32 *dstbuffer;
  u32 payload[8];
  // set up key/CBC init in the payload
  payload[0]=0x01234567;           // key
  payload[1]=0x89ABCDEF;
  payload[2]=0xfedcba98;
  payload[3]=0x76543210;
  payload[4]=0x00112233;           // CBC initialization
  payload[5]=0x44556677;
  payload[6]=0x8899aabb;
  payload[7]=0xccddeeff;
  // set up control packet
  dcp1.next = 0;                   // single packet in chain
  dcp1.ctrl0.U = 0;                // clear ctrl0 field
  dcp1.ctrl0.B.PAYLOAD_KEY = 1;    // key is located in payload
  dcp1.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp1.ctrl0.B.CIPHER_INIT = 1;    // init CBC for this block (from payload)
  dcp1.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;      // interrupt
  dcp1.ctrl1.U = 0;                // clear ctrl1
  dcp1.ctrl1.B.CIPHER_MODE = 1;    // select CBC mode of operation
  dcp1.src = srcbuffer;            // source buffer
  dcp1.dst = dstbuffer;            // destination buffer
```

```
  dcp1.buf_size = 512;              // 512 bytes
  dcp1.payload = payload;           // holds key/CBC init
  dcp1.status = 0;                  // clear status
  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);     // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);          // increment semaphore by 1
  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);
</code>
```

17.2.7.4 Multi-Buffer Scatter/Gather Cipher and Hash Operation Programming Example

For this example, three separate buffers are encrypted and hashed with the results being directed to a unified buffer (gather operation). Three descriptors are used for the operation because there are three separate source buffer pointers. The DCP reads data from the source buffer and computes a hash on the unencrypted data. It then encrypts the data and writes it to the destination buffer. For this example, the key is located in the key RAM, and the algorithm uses the AES CBC mode of operation with a SHA-1 hash. The payload for the first operation contains the CBC initialization value, and the payload for the last packet contains the expected hash value. The middle packet requires no payload. This process is illustrated in Figure 17-9.

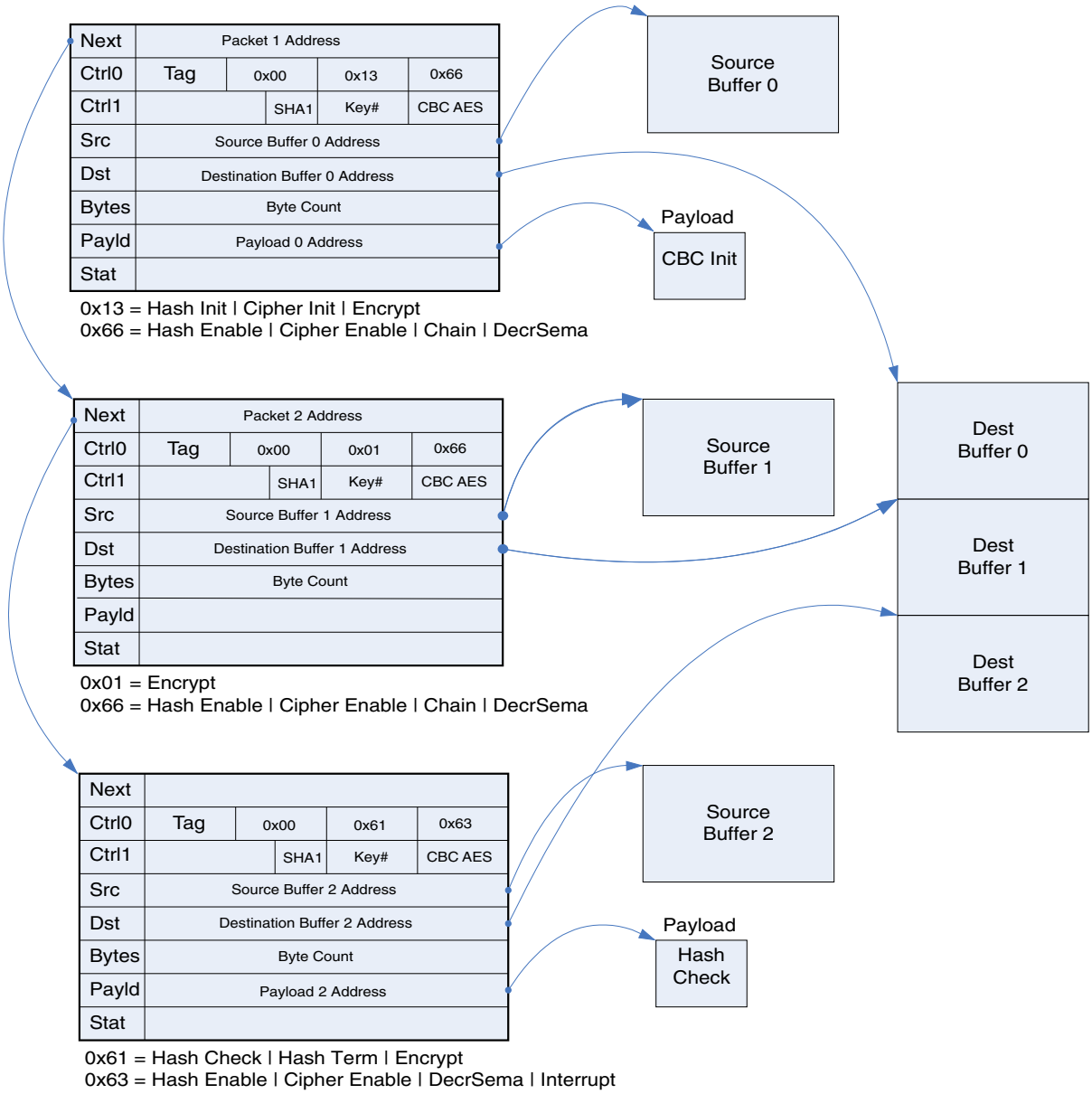


Figure 17-9. Multi-Buffer Scatter/Gather Cipher and Hash Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32              *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32              *src,
                     *dst,
                      buf_size,
                     *payload,
                      stat;
}  DCP_DESCRIPTOR;
  DCP_DESCRIPTOR dcp1, dcp2, dcp3;
  u32 *srcbuffer0, *srcbuffer2, *srcbuffer3;
  u32 *dstbuffer;
  u32 payload0[4], payload2[5];
  // set up CBC init in the payload
  payload0[0]=0x01234567;           // key
  payload0[1]=0x89ABCDEF;
```

```
  payload0[2]=0xfedcba98;
  payload0[3]=0x76543210;
  payload2[0]=0x00112233;           // CBC initialization
  payload2[1]=0x44556677;
  payload2[2]=0x8899aabb;
  payload2[3]=0xccddeeff;
  payload2[3]=0xaabbccdd;
  // set up control packet
  dcp1.next = dcp2;                 // point to packet 2
  dcp1.ctrl0.U = 0;                 // clear ctrl0 field
  dcp1.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp1.ctrl0.B.CIPHER_INIT = 1;     // init CBC for this block (from payload)
  dcp1.ctrl0.B.HASH_INIT = 1;       // init hash this block
  dcp1.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp1.ctrl0.B.ENABLE_HASH = 1;     // enable cipher
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.CHAIN = 1;           // chain to next packet
  dcp1.ctrl1.U = 0;                 // clear ctrl1
  dcp1.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE__CBC; //select CBC mode of
operation
  dcp1.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT__SHA1;    // select SHA1 hash
  dcp1.ctrl1.B.KEY_SELECT  = 2;     // select key #2
  dcp1.src = srcbuffer0;             // source buffer
  dcp1.dst = dstbuffer;             // destination buffer
  dcp1.buf_size = 512;              // 512 bytes
  dcp1.payload = payload0;           // holds key/CBC init
  dcp1.status = 0;                  // clear status
  // set up control packet
  dcp2.next = dcp3;                 // point to packet 2
  dcp2.ctrl0.U = 0;                 // clear ctrl0 field
  dcp2.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp2.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp2.ctrl0.B.ENABLE_HASH = 1;     // enable cipher
  dcp2.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp2.ctrl0.B.CHAIN = 1;           // chain to next packet
  dcp2.ctrl1.U = 0;                 // clear ctrl1
  dcp2.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE__CBC; //select CBC mode of
operation
  dcp2.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT__SHA1;    // select SHA1 hash
  dcp2.ctrl1.B.KEY_SELECT  = 2;     // select key #2
  dcp2.src = srcbuffer1;            // source buffer
  dcp2.dst = dstbuffer+512;         // destination buffer
  dcp2.buf_size = 512;              // 512 bytes
  dcp2.payload = NULL;              // no payload required
  dcp2.status = 0;                  // clear status
  // set up control packet
  dcp3.next = dcp3;                 // point to packet 2
  dcp3.ctrl0.U = 0;                 // clear ctrl0 field
  dcp3.ctrl0.B.HASH_TERM      = 1; // terminate hash block
  dcp3.ctrl0.B.HASH_CHECK     = 1; // check hash against payload value
  dcp3.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp3.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp3.ctrl0.B.ENABLE_HASH = 1;     // enable cipher
  dcp3.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp3.ctrl0.B.INTERRUPT = 1;       // interrupt on completion
  dcp3.ctrl1.U = 0;                 // clear ctrl1
  dcp3.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE__CBC; //select CBC mode of
operation
  dcp3.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT__SHA1;    // select SHA1 hash
  dcp3.ctrl1.B.KEY_SELECT  = 2;     // select key #2
  dcp3.src = srcbuffer1;            // source buffer
  dcp3.dst = dstbuffer+1024;        // destination buffer
  dcp3.buf_size = 512;              // 512 bytes
  dcp3.payload = payload2;          // payload is hash check value
  dcp3.status = 0;                  // clear status
  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);     // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 3);          // increment semaphore by 3 (for 3 packets)
  // now wait for interrupt or poll
  // polling code
```

```
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);
</code>
```

17.3 Programmable Registers

DCP Hardware Register Format Summary

DCP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0000_0000	DCP Control Register 0 (DCP_CTRL)	32	R/W	F080_0000h	17.3.1/ 804
0000_0010	DCP Status Register (DCP_STAT)	32	R/W	1000_0000h	17.3.2/ 806
0000_0020	DCP Channel Control Register (DCP_CHANNELCTRL)	32	R/W	0000_0000h	17.3.3/ 808
0000_0030	DCP Capability 0 Register (DCP_CAPABILITY0)	32	R/W	0000_0404h	17.3.4/ 809
0000_0040	DCP Capability 1 Register (DCP_CAPABILITY1)	32	R	0007_0001h	17.3.5/ 810
0000_0050	DCP Context Buffer Pointer (DCP_CONTEXT)	32	R/W	0000_0000h	17.3.6/ 811
0000_0060	DCP Key Index (DCP_KEY)	32	R/W	0000_0000h	17.3.7/ 811
0000_0070	DCP Key Data (DCP_KEYDATA)	32	R/W	0000_0000h	17.3.8/ 812
0000_0080	DCP Work Packet 0 Status Register (DCP_PACKET0)	32	R	0000_0000h	17.3.9/ 813
0000_0090	DCP Work Packet 1 Status Register (DCP_PACKET1)	32	R	0000_0000h	17.3.10/ 813
0000_00A0	DCP Work Packet 2 Status Register (DCP_PACKET2)	32	R	0000_0000h	17.3.11/ 816
0000_00B0	DCP Work Packet 3 Status Register (DCP_PACKET3)	32	R	0000_0000h	17.3.12/ 817
0000_00C0	DCP Work Packet 4 Status Register (DCP_PACKET4)	32	R	0000_0000h	17.3.13/ 817

Table continues on the next page...

DCP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0000_00D0	DCP Work Packet 5 Status Register (DCP_PACKET5)	32	R	0000_0000h	17.3.14/ 818
0000_00E0	DCP Work Packet 6 Status Register (DCP_PACKET6)	32	R	0000_0000h	17.3.15/ 818
0000_0100	DCP Channel 0 Command Pointer Address Register (DCP_CH0CMDPTR)	32	R/W	0000_0000h	17.3.16/ 819
0000_0110	DCP Channel 0 Semaphore Register (DCP_CH0SEMA)	32	R/W	0000_0000h	17.3.17/ 820
0000_0120	DCP Channel 0 Status Register (DCP_CH0STAT)	32	R/W	0000_0000h	17.3.18/ 821
0000_0130	DCP Channel 0 Options Register (DCP_CH0OPTS)	32	R/W	0000_0000h	17.3.19/ 823
0000_0140	DCP Channel 1 Command Pointer Address Register (DCP_CH1CMDPTR)	32	R/W	0000_0000h	17.3.20/ 824
0000_0150	DCP Channel 1 Semaphore Register (DCP_CH1SEMA)	32	R/W	0000_0000h	17.3.21/ 825
0000_0160	DCP Channel 1 Status Register (DCP_CH1STAT)	32	R/W	0000_0000h	17.3.22/ 826
0000_0170	DCP Channel 1 Options Register (DCP_CH1OPTS)	32	R/W	0000_0000h	17.3.23/ 827
0000_0180	DCP Channel 2 Command Pointer Address Register (DCP_CH2CMDPTR)	32	R/W	0000_0000h	17.3.24/ 828
0000_0190	DCP Channel 2 Semaphore Register (DCP_CH2SEMA)	32	R/W	0000_0000h	17.3.25/ 829
0000_01A0	DCP Channel 2 Status Register (DCP_CH2STAT)	32	R/W	0000_0000h	17.3.26/ 830
0000_01B0	DCP Channel 2 Options Register (DCP_CH2OPTS)	32	R/W	0000_0000h	17.3.27/ 832
0000_01C0	DCP Channel 3 Command Pointer Address Register (DCP_CH3CMDPTR)	32	R/W	0000_0000h	17.3.28/ 833
0000_01D0	DCP Channel 3 Semaphore Register (DCP_CH3SEMA)	32	R/W	0000_0000h	17.3.29/ 834
0000_01E0	DCP Channel 3 Status Register (DCP_CH3STAT)	32	R/W	0000_0000h	17.3.30/ 835
0000_01F0	DCP Channel 3 Options Register (DCP_CH3OPTS)	32	R/W	0000_0000h	17.3.31/ 836
0000_0400	DCP Debug Select Register (DCP_DBGSELECT)	32	R/W	0000_0000h	17.3.32/ 837
0000_0410	DCP Debug Data Register (DCP_DBGDATA)	32	R	0000_0000h	17.3.33/ 838

Table continues on the next page...

DCP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0000_0420	DCP Page Table Register (DCP_PAGETABLE)	32	R/W	0000_0000h	17.3.34/ 838
0000_0430	DCP Version Register (DCP_VERSION)	32	R	0201_0000h	17.3.35/ 839

17.3.1 DCP Control Register 0 (DCP_CTRL)

The CTRL register contains controls for the DCP module.

CTRL: 0x000

CTRL_SET: 0x004

CTRL_CLR: 0x008

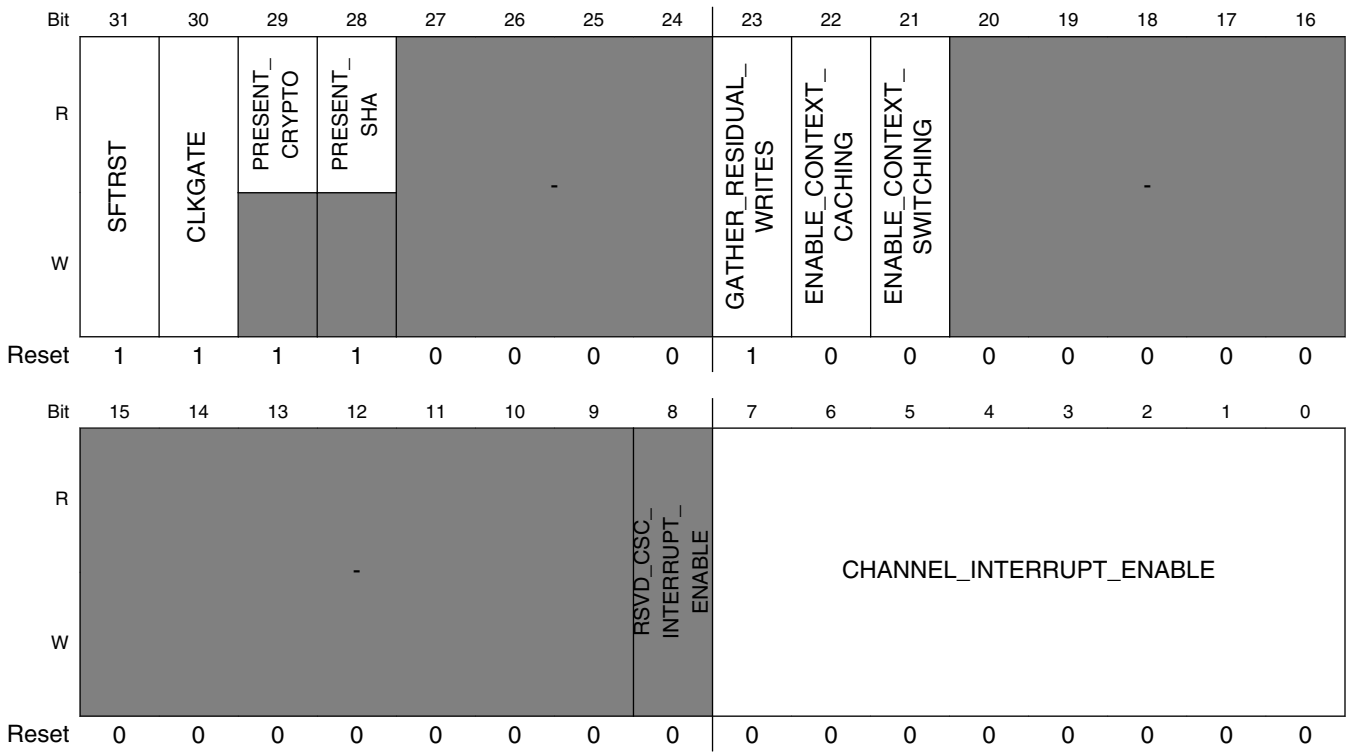
CTRL_TOG: 0x00C

The Control register contains the primary controls for the DCP block. The present bits indicate which of the sub-features of the block are present in the hardware. The context control bits control how the DCP utilizes it's context buffer and the gather residual writes bit controls how the master handles writing misaligned data to the bus. Each channel and the color-space converter contains an independent interrupt enable.

EXAMPLE

```
DCP_CTRL_SET(BM_DCP_CTRL_SFTRST);
DCP_CTRL_CLR(BM_DCP_CTRL_SFTRST | BM_DCP_CTRL_CLKGATE);
```

Address: DCP_CTRL is 0h base + 0h offset = 0000_0000h



DCP_CTRL field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal DCP operation. Set this bit to one (default) to disable clocking with the DCP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the DCP block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 PRESENT_ CRYPTO	Indicates whether the crypto (Cipher/Hash) functions are present. 0x1 **Present** — 0x0 **Absent** —
28 PRESENT_SHA	Indicates whether the SHA1/SHA2 functions are present. 0x1 **Present** — 0x0 **Absent** —
27–24 -	Reserved, always set to zero.
23 GATHER_ RESIDUAL_ WRITES	Software should set this bit to enable ragged writes to unaligned buffers to be gathered between multiple write operations. This improves performance by removing several byte operations between write bursts. Trailing byte writes are held in a residual write data buffer and combined with a subsequent write to the buffer to form a word write.
22 ENABLE_ CONTEXT_ CACHING	Software should set this bit to enable caching of contexts between operations. If only a single channel is used for encryption/hashing, enabling caching causes the context to not be reloaded if the channel was the last to be used.

Table continues on the next page...

DCP_CTRL field descriptions (continued)

Field	Description
21 ENABLE_ CONTEXT_ SWITCHING	Enable automatic context switching for the channels. Software should set this bit if more than one channel is doing hashing or cipher operations that require context to be saved (for instance, when CBC mode is enabled). By disabling context switching, software can save the 208 bytes used for the context buffer.
20–9 -	Reserved, always set to zero.
8 RSVD_CSC_ INTERRUPT_ ENABLE	Reserved, always set to zero.
7–0 CHANNEL_ INTERRUPT_ ENABLE	Per-channel interrupt enable bit. When set, the channel's interrupt will get routed to the interrupt controller. Channel 0 is routed to the dcp_vmi_irq signal and the other channels are combined (along with the CRC interrupt) into the dcp_irq signal. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —

17.3.2 DCP Status Register (DCP_STAT)

The DCP Interrupt Status register provides channel interrupt status information.

STAT: 0x010

STAT_SET: 0x014

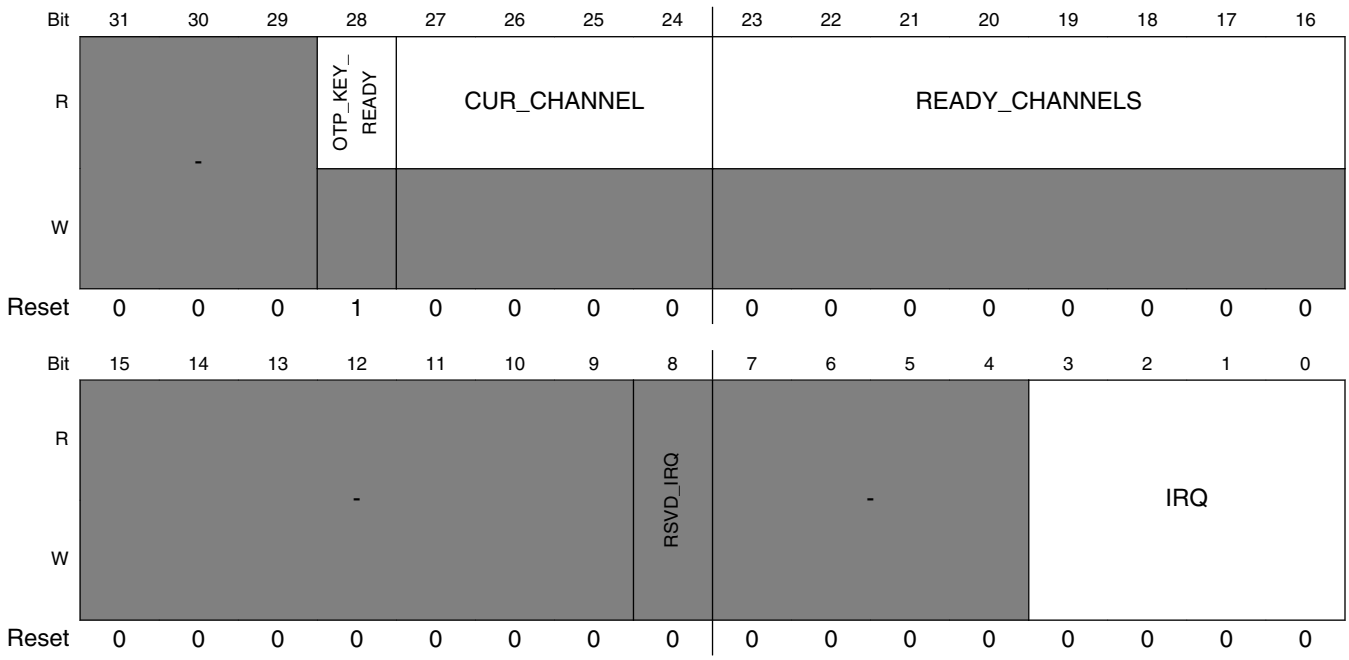
STAT_CLR: 0x018

STAT_TOG: 0x01C

This register provides status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE

Address: DCP_STAT is 0h base + 10h offset = 0000_0010h



DCP_STAT field descriptions

Field	Description
31–29 -	Reserved, always set to zero.
28 OTP_KEY_READY	When set, indicates that the OTP key has been shifted from the fuse block and is ready for use.
27–24 CUR_CHANNEL	Current (active) channel (encoded). 0x0 **None** — 0x1 **CH0** — 0x2 **CH1** — 0x3 **CH2** — 0x4 **CH3** —
23–16 READY_CHANNELS	Indicates which channels are ready to proceed with a transfer (active channel also included). Each bit is a one-hot indicating the request status for the associated channel. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —
15–9 -	Reserved, always set to zero.
8 RSVD_IRQ	Reserved, always set to zero.
7–4 -	Reserved, always set to zero.

Table continues on the next page...

DCP_STAT field descriptions (continued)

Field	Description
3–0 IRQ	Indicates which channels have pending interrupt requests. Channel 0's interrupt is routed through the dcp_vmi_irq and the other interrupt bits are routed through the dcp_irq.

17.3.3 DCP Channel Control Register (DCP_CHANNELCTRL)

The DCP Channel Control register provides controls for channel arbitration and channel enables.

CHANNELCTRL: 0x020

CHANNELCTRL_SET: 0x024

CHANNELCTRL_CLR: 0x028

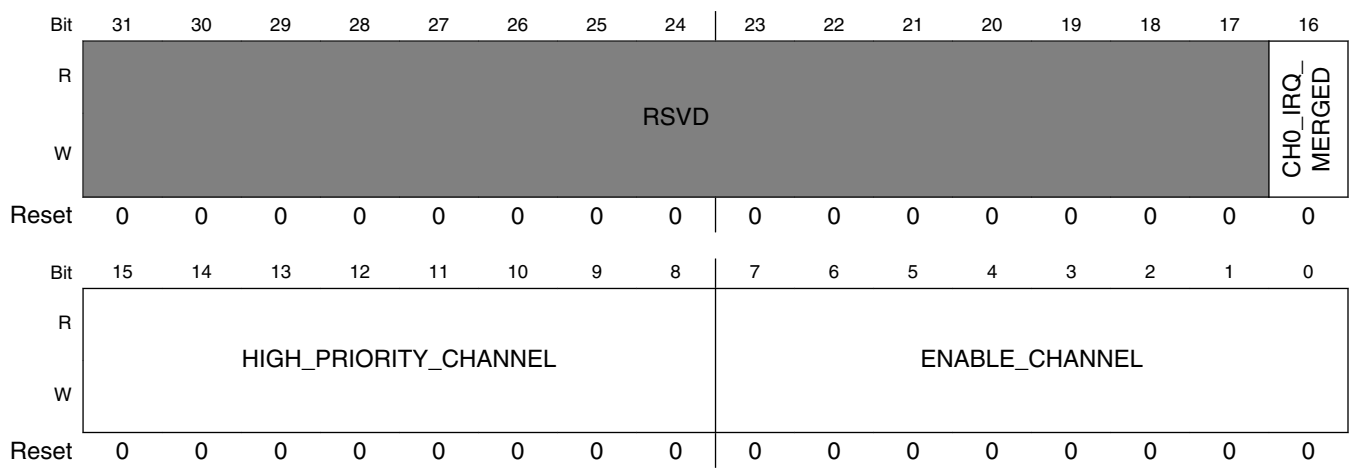
CHANNELCTRL_TOG: 0x02C

This register provides status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE

```
        BW_DCP_CHANNELCTRL_ENABLE_CHANNEL(BV_DCP_CHANNELCTRL_ENABLE_CHANNEL__CH0);  //
enable channel 0
```

Address: DCP_CHANNELCTRL is 0h base + 20h offset = 0000_0020h



DCP_CHANNELCTRL field descriptions

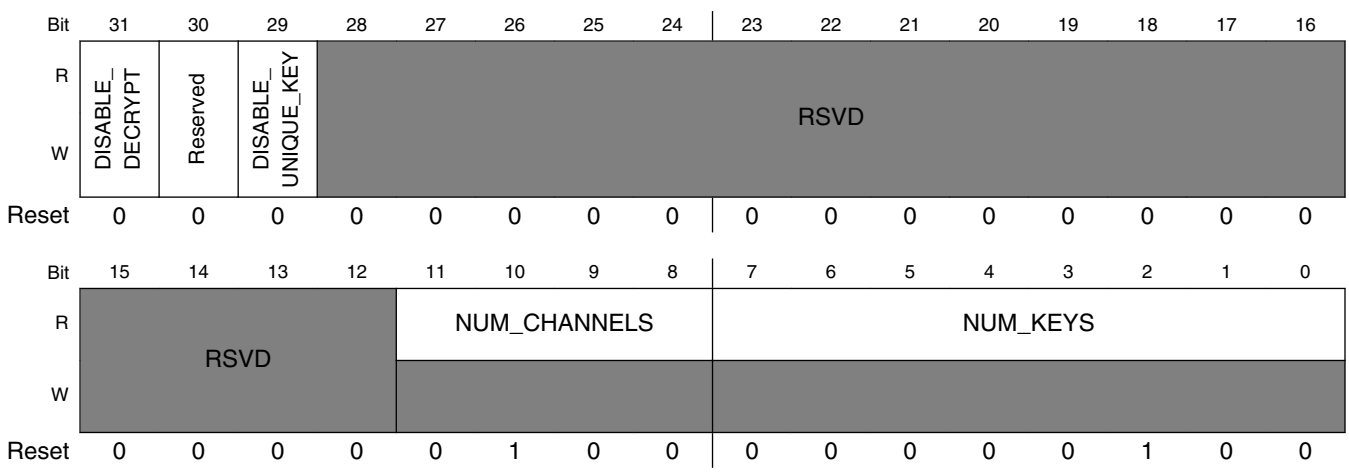
Field	Description
31–17 RSVD	Reserved, always set to zero.
16 CH0_IRQ_ MERGED	Indicates that the interrupt for channel 0 should be merged with the other interrupts on the shared dcp_irq interrupt. When set to 0, channel 0's interrupt will be routed to the separate dcp_vmi_irq. When set to 1, the interrupt will be routed to the shared DCP interrupt.
15–8 HIGH_ PRIORITY_ CHANNEL	Setting a bit in this field causes the corresponding channel to have high-priority arbitration. High priority channels will be arbitrated round-robin and will take precedence over other channels that are not marked as high priority. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —
7–0 ENABLE_ CHANNEL	Setting a bit in this field will enabled the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When not enabled, the channel is denied access to the central DMA resources. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —

17.3.4 DCP Capability 0 Register (DCP_CAPABILITY0)

This register contains additional information about the DCP module implementation parameters.

This register provides capability information for the DCP block. It indicates the number of channels implemented as well as the number of key storage locations available for software use.

Address: DCP_CAPABILITY0 is 0h base + 30h offset = 0000_0030h



DCP_CAPABILITY0 field descriptions

Field	Description
31 DISABLE_ DECRYPT	Write to a 1 to disable decryption. This bit can only be written by secure software and the value can only be cleared by a reset.
30 Reserved	This field is reserved. -
29 DISABLE_ UNIQUE_KEY	Write to a 1 disable the per-device unique key. The device-specific hardware key may be selected by using a value of 0xFE in the key select field.
28–12 RSVD	Reserved, always set to zero.
11–8 NUM_ CHANNELS	Encoded value indicating the number of channels implemented in the design.
7–0 NUM_KEYS	Encoded value indicating the number of key storage locations implemented in the design.

17.3.5 DCP Capability 1 Register (DCP_CAPABILITY1)

This register contains information about the algorithms available on the implementation.

This register provides capability information for the DCP block. It contains two fields indicating which encryption and hashing algorithms are present in the design. Each bit set indicates that support for the associated function is present.

Address: DCP_CAPABILITY1 is 0h base + 40h offset = 0000_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HASH_ALGORITHMS																CIPHER_ALGORITHMS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

DCP_CAPABILITY1 field descriptions

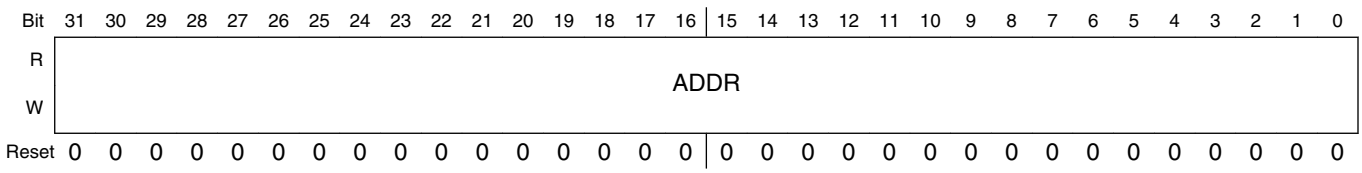
Field	Description
31–16 HASH_ ALGORITHMS	One-hot field indicating which hashing features are implemented in HW. 0x0001 **SHA1** — 0x0002 **CRC32** — 0x0004 **SHA256** —
15–0 CIPHER_ ALGORITHMS	One-hot field indicating which cipher algorithms are available. 0x0001 **AES128** —

17.3.6 DCP Context Buffer Pointer (DCP_CONTEXT)

This register contains a pointer to the memory region to be used for DCP context swap operations.

This register contains a pointer to the start of the context pointer memory in on-chip SRAM or off-chip SDRAM. This buffer will be used to store state information when the DCP module changes from one channel to another.

Address: DCP_CONTEXT is 0h base + 50h offset = 0000_0050h



DCP_CONTEXT field descriptions

Field	Description
31–0 ADDR	Context pointer address. Address should be located in system RAM and should be word-aligned for optimal performance.

17.3.7 DCP Key Index (DCP_KEY)

This register contains a pointer to the key location to be written.

The DCP module maintains a set of write-only keys that may be used by software. To write a key, software must first write the desired key index/subword to this register and then write the key values to the key registers (below). After each write to the key data register, the SUBWORD field will increment to allow software to write the subsequent key to be written without having to rewrite the key index.

EXAMPLE

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeff
DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0));  // set key index to key 0, subword
0
DCP_KEYDATA_WR(0xccddeeff);  // write key values (subword 0)
DCP_KEYDATA_WR(0x8899aabb);  // write key values (subword 1)
DCP_KEYDATA_WR(0x44556677);  // write key values (subword 2)
DCP_KEYDATA_WR(0x00112233);  // write key values (subword 3)
```

Address: DCP_KEY is 0h base + 60h offset = 0000_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSVD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD								RSVD_INDEX		INDEX		RSVD_ SUBWORD		SUBWORD	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_KEY field descriptions

Field	Description
31–8 RSVD	Reserved, always set to zero.
7–6 RSVD_INDEX	Reserved, always set to zero.
5–4 INDEX	Key index pointer. Valid indices are 0-[number_keys].
3–2 RSVD_ SUBWORD	Reserved, always set to zero.
1–0 SUBWORD	Key subword pointer. Valid indices are 0-3. After each write to the key data register, this field will increment.

17.3.8 DCP Key Data (DCP_KEYDATA)

This register provides write access to the key/key subword specified by the Key Index Register.

Writing this location updates the selected subword for the key located at the index specified by the Key Index Register. A write also triggers the SUBWORD field of the KEY register to increment to the next higher word in the key.

EXAMPLE

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeff
DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0));  // set key index to key 0, subword
0
DCP_KEYDATA_WR(0xccddeeff);  // write key values (subword 0)
DCP_KEYDATA_WR(0x8899aabb);  // write key values (subword 1)
DCP_KEYDATA_WR(0x44556677);  // write key values (subword 2)
DCP_KEYDATA_WR(0x00112233);  // write key values (subword 3)
```

Address: DCP_KEYDATA is 0h base + 70h offset = 0000_0070h

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

DCP_KEYDATA field descriptions

Field	Description
31–0 DATA	Word 0 data for key. This is the least-significant word.

17.3.9 DCP Work Packet 0 Status Register (DCP_PACKET0)

This register displays the values for the current work packet offset 0x00 (Next Command) field.

The Work Packet Status Registers show the contents of the currently executing packet. When the channels are inactive, the packet status register return 0. The register bits are fully documented here to document the packet structure in memory.

Address: DCP_PACKET0 is 0h base + 80h offset = 0000_0080h

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

DCP_PACKET0 field descriptions

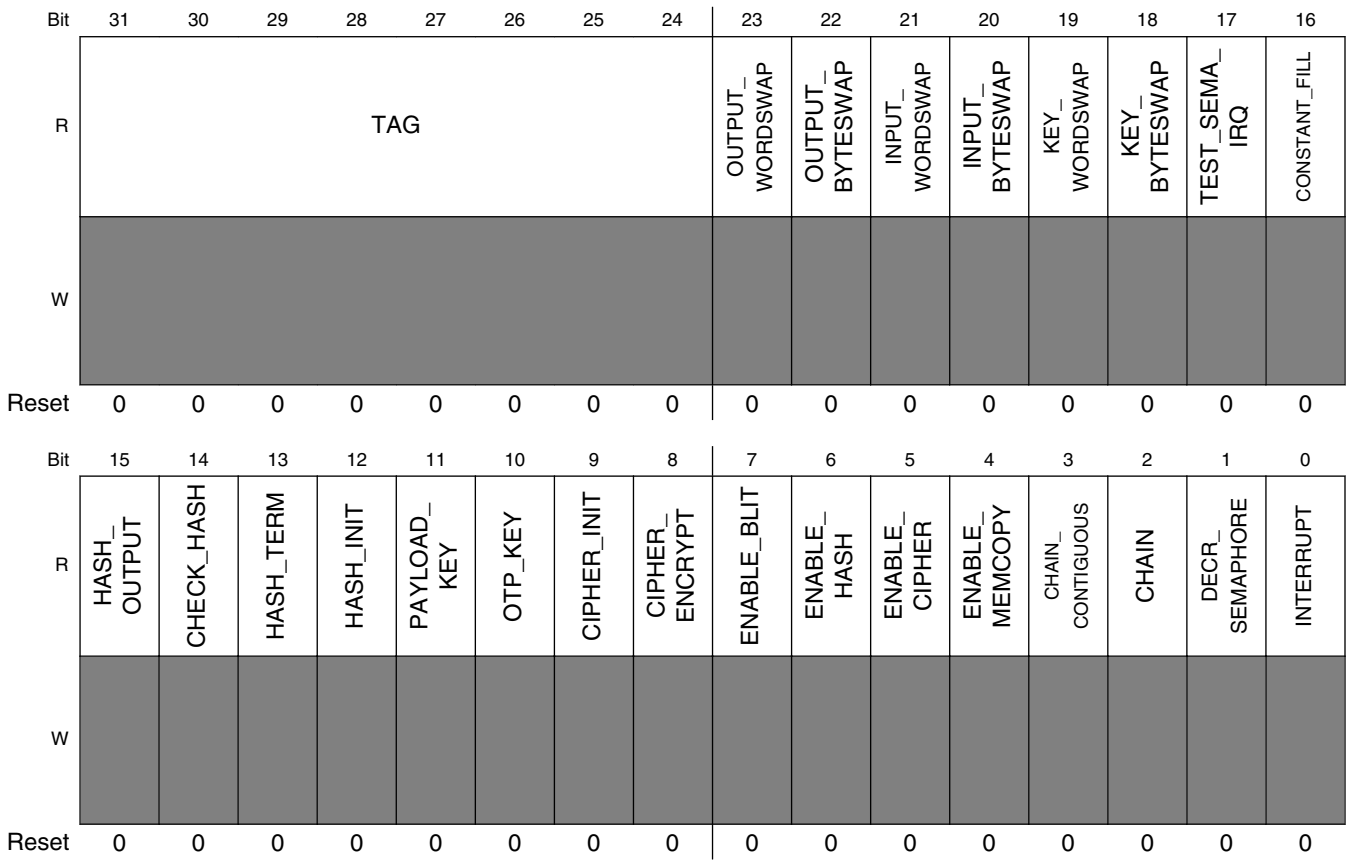
Field	Description
31–0 ADDR	Next Pointer Register,

17.3.10 DCP Work Packet 1 Status Register (DCP_PACKET1)

This register displays the values for the current work packet offset 0x04 (control) field.

This register shows the contents of the Control0 register from the packet being processed.

Address: DCP_PACKET1 is 0h base + 90h offset = 0000_0090h



DCP_PACKET1 field descriptions

Field	Description
31–24 TAG	Packet Tag
23 OUTPUT_ WORDSWAP	Reflects whether the DCP engine will wordswap output data (big-endian data).
22 OUTPUT_ BYTESWAP	Reflects whether the DCP engine will byteswap output data (big-endian data).
21 INPUT_ WORDSWAP	Reflects whether the DCP engine will wordswap input data (big-endian data).
20 INPUT_ BYTESWAP	Reflects whether the DCP engine will byteswap input data (big-endian data).
19 KEY_ WORDSWAP	Reflects whether the DCP engine will swap key words (big-endian key).
18 KEY_ BYTESWAP	Reflects whether the DCP engine will swap key bytes (big-endian key).

Table continues on the next page...

DCP_PACKET1 field descriptions (continued)

Field	Description
17 TEST_SEMA_ IRQ	This bit is used to test the channel semaphore transition to 0. FOR TEST USE ONLY!
16 CONSTANT_ FILL	When this bit is set (MEMCOPY and BLIT modes only), the DCP will simply fill the destination buffer with the value found in the Source Address field.
15 HASH_OUTPUT	When hashing is enabled, this bit controls whether the input or output data is hashed. 0 **INPUT** — 1 **OUTPUT** —
14 CHECK_HASH	Reflects whether the calculated hash value should be compared against the hash provided in the payload.
13 HASH_TERM	Reflects whether the current hashing block is the final block in the hashing operation, so the hash padding should be applied by hardware.
12 HASH_INIT	Reflects whether the current hashing block is the initial block in the hashing operation, so the hash registers should be initialized before the operation.
11 PAYLOAD_KEY	When set, indicates the payload contains the key. This bit takes precedence over the OTP_KEY control
10 OTP_KEY	Reflects whether a hardware-based key should be used. The KEY_SELECT field from the Control1 field is used to select from multiple hardware keys. The PAYLOAD_KEY bit takes precedence over the OTP_KEY bit.
9 CIPHER_INIT	Reflects whether the cipher block should load the initialization vector from the payload for this operation.
8 CIPHER_ ENCRYPT	When the cipher block is enabled, this bit indicates whether the operation is encryption or decryption. 1 **ENCRYPT** — 0 **DECRYPT** —
7 ENABLE_BLIT	Reflects whether the DCP should perform a blit operation. Source data is always continuous and the destination buffer is written in run/stride format. When set, the BUFFER_SIZE field is treated as two 16-bit values for the X-Y extents of the blit operation.
6 ENABLE_HASH	Reflects whether the selected hashing function should be enabled for this operation.
5 ENABLE_ CIPHER	Reflects whether the selected cipher function should be enabled for this operation.
4 ENABLE_ MEMCOPY	Reflects whether the selected hashing function should be enabled for this operation.
3 CHAIN_ CONTIGUOUS	Reflects whether the next packet's address is located following this packet's payload.
2 CHAIN	Reflects whether the next command pointer register should be loaded into the channel's current descriptor pointer.

Table continues on the next page...

DCP_PACKET1 field descriptions (continued)

Field	Description
1 DECR_ SEMAPHORE	Reflects whether the channel's semaphore should be decremented at the end of the current operation. When the semaphore reaches a value of zero, no more operations will be issued from the channel.
0 INTERRUPT	Reflects whether the channel should issue an interrupt upon completion of the packet.

17.3.11 DCP Work Packet 2 Status Register (DCP_PACKET2)

This register displays the values for the current work packet offset 0x08 (Control1) field.

This register shows the contents of the Control0 register from the packet being processed.

Address: DCP_PACKET2 is 0h base + A0h offset = 0000_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CIPHER_CFG								RSVD				HASH_SELECT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	KEY_SELECT								CIPHER_MODE				CIPHER_SELECT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_PACKET2 field descriptions

Field	Description
31–24 CIPHER_CFG	Cipher configuration bits. Optional configuration bits required for ciphers
23–20 RSVD	Reserved, always set to zero.
19–16 HASH_SELECT	Hash Selection Field 0x00 **SHA1** — 0x01 **CRC32** — 0x02 **SHA256** —
15–8 KEY_SELECT	Key Selection Field. The value here reflects the key index for the cipher operation. Values 0-3 refer to the software keys that can be written to the key RAM. The OTP key or the unique device-specific key may also be selected with a value of 0xFF (OTP key) or 0xFE (unique key). 0x00 **KEY0** — 0x01 **KEY1** — 0x02 **KEY2** — 0x03 **KEY3** —

Table continues on the next page...

DCP_PACKET2 field descriptions (continued)

Field	Description
	0xFE **UNIQUE_KEY** — 0xFF **OTP_KEY** —
7–4 CIPHER_MODE	Cipher Mode Selection Field. Reflects the mode of operation for cipher operations. 0x00 **ECB** — 0x01 **CBC** —
3–0 CIPHER_ SELECT	Cipher Selection Field 0x00 **AES128** —

17.3.12 DCP Work Packet 3 Status Register (DCP_PACKET3)

This register displays the values for the current work packet offset 0x0C (Source Address) field.

This register shows the contents of the Source Address register from the packet being processed. When the CONSTANT_FILL bit in the Control 0 field is set, this field contains the data written to the destination buffer.

Address: DCP_PACKET3 is 0h base + B0h offset = 0000_00B0h

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

DCP_PACKET3 field descriptions

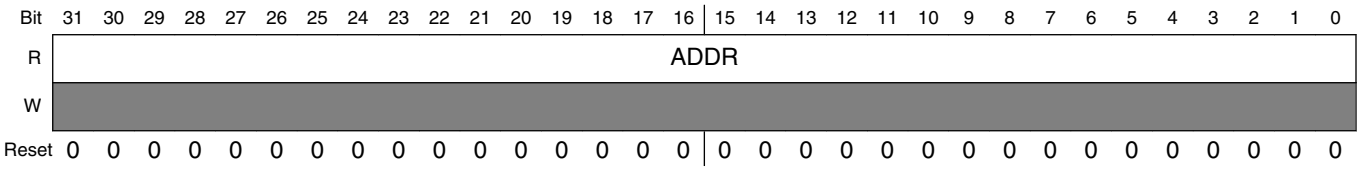
Field	Description
31–0 ADDR	Source Buffer Address Pointer. This value is the working value and will update as the operation proceeds.

17.3.13 DCP Work Packet 4 Status Register (DCP_PACKET4)

This register displays the values for the current work packet offset 0x10 (Destination Address) field.

This register shows the contents of the Destination Address register from the packet being processed.

Address: DCP_PACKET4 is 0h base + C0h offset = 0000_00C0h



DCP_PACKET4 field descriptions

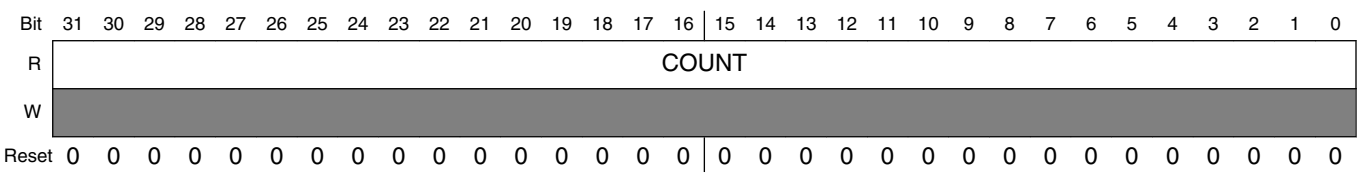
Field	Description
31–0 ADDR	Destination Buffer Address Pointer. This value is the working value and will update as the operation proceeds.

17.3.14 DCP Work Packet 5 Status Register (DCP_PACKET5)

This register displays the values for the current work packet offset 0x14 (Buffer Size) field.

This register shows the contents of the bytecount register from the packet being processed. The field can be considered either a byte count or a buffer size. The logic treats this as a decrmenting count of bytes from the buffer size programmed into the field. As the transaction proceeds, the logic will decrement the bytecount as data is written to the destination buffer. For blit operations, the top 16-bits of this field represents the number of lines (y size) in the blit and the lower 16-bits represent the number of bytes in a line (x size).

Address: DCP_PACKET5 is 0h base + D0h offset = 0000_00D0h



DCP_PACKET5 field descriptions

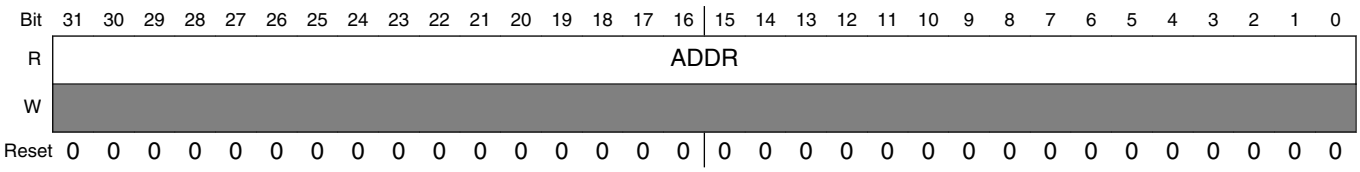
Field	Description
31–0 COUNT	Byte Count register. This value is the working value and will update as the operation proceeds.

17.3.15 DCP Work Packet 6 Status Register (DCP_PACKET6)

This register displays the values for the current work packet offset 0x1C (Payload Pointer) field.

This register shows the contents of the payload pointer fieldr from the packet being processed.

Address: DCP_PACKET6 is 0h base + E0h offset = 0000_00E0h



DCP_PACKET6 field descriptions

Field	Description
31–0 ADDR	This regiser reflects the payload pointer for the current control packet.

17.3.16 DCP Channel 0 Command Pointer Address Register (DCP_CH0CMDPTR)

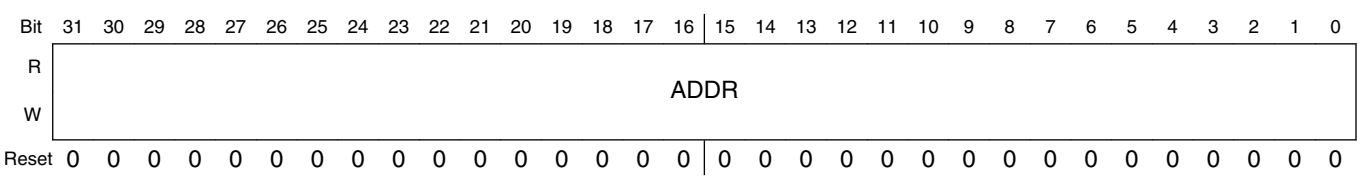
The DCP channel 0 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 0 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
            DCP_CHnCMDPTR_WR(0, v);   // Write channel 0 command pointer
            pCurptr = (DCP_chncmdptr_t *) DCP_CHnCMDPTR_RD(0);  // Read current command
pointer
```

Address: DCP_CH0CMDPTR is 0h base + 100h offset = 0000_0100h



DCP_CH0CMDPTR field descriptions

Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 0.

17.3.17 DCP Channel 0 Semaphore Register (DCP_CH0SEMA)

The DCP Channel 0 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjuction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. After processing each control packet, the DCP decrements the semaphore if it is non-zero. The channel will continue processing packets as long as the semaphore contains a non-zero value and the CHAIN or CHAIN_CONTIGOUS control bits in the Control0 field are set.

Address: DCP_CH0SEMA is 0h base + 110h offset = 0000_0110h

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

DCP_CH0SEMA field descriptions

Field	Description
31–24 -	Reserved, always set to zero.

Table continues on the next page...

DCP_CH0SEMA field descriptions (continued)

Field	Description
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

17.3.18 DCP Channel 0 Status Register (DCP_CH0STAT)

The DCP Channel 0 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

DCP_CH0STAT: 0x120

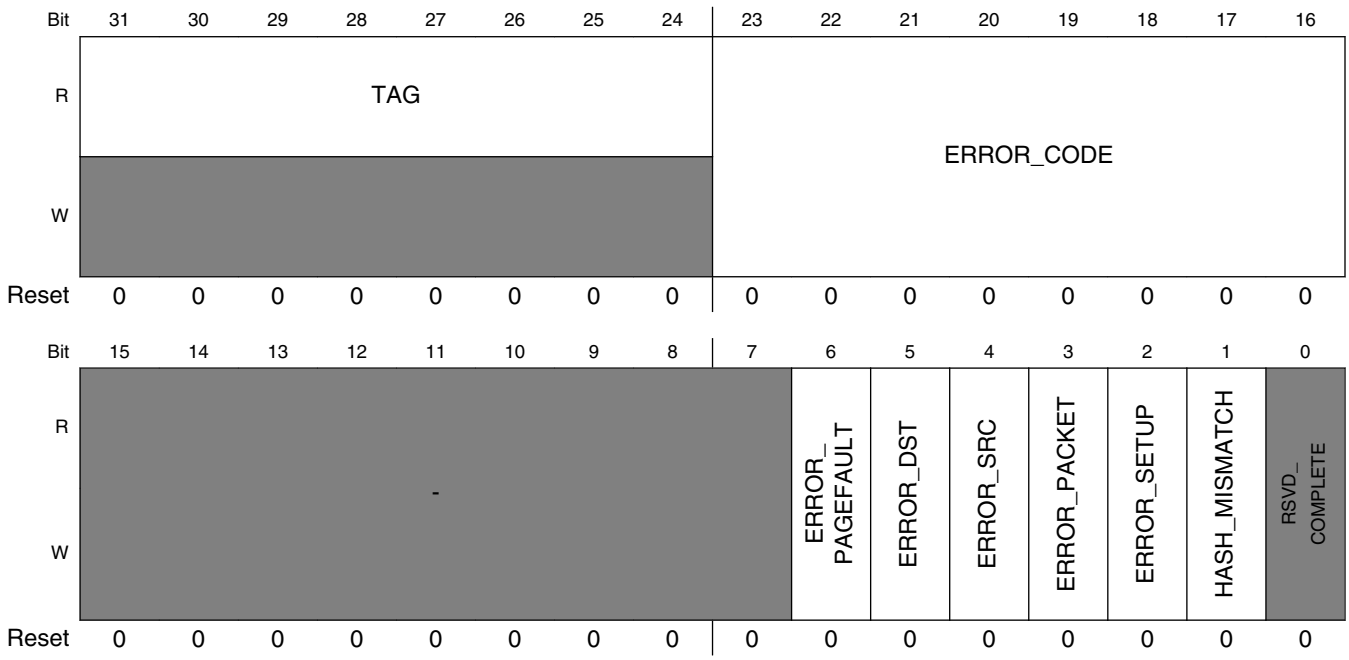
CH0STAT_SET: 0x124

CH0STAT_CLR: 0x128

CH0STAT_TOG: 0x12C

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: DCP_CH0STAT is 0h base + 120h offset = 0000_0120h



DCP_CH0STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a a bus error occurred when reading the packet or payload or when writing status back to the packet payload. When an error is detected, the channel's processing will stop until the error is handled by software.

Table continues on the next page...

DCP_CH0STAT field descriptions (continued)

Field	Description
2 ERROR_SETUP	This bit indicates that the hardware has detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

17.3.19 DCP Channel 0 Options Register (DCP_CH0OPTS)

The DCP Channel 0 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

DCP_CH0OPTS: 0x130

CH0OPTS_SET: 0x134

CH0OPTS_CLR: 0x138

CH0OPTS_TOG: 0x13C

The options register can be used to control optional features of the channels.

Address: DCP_CH0OPTS is 0h base + 130h offset = 0000_0130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD																RECOVERY_TIMER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH0OPTS field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate another operation for the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

17.3.20 DCP Channel 1 Command Pointer Address Register (DCP_CH1CMDPTR)

The DCP channel 1 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 1 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
            DCP_CHn_CMDPTR_WR(1, v);   // Write channel 1 command pointer
            pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(1);  // Read current command
pointer
```

Address: DCP_CH1CMDPTR is 0h base + 140h offset = 0000_0140h

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

DCP_CH1CMDPTR field descriptions

Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 1.

17.3.21 DCP Channel 1 Semaphore Register (DCP_CH1SEMA)

The DCP Channel 1 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjuction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: DCP_CH1SEMA is 0h base + 150h offset = 0000_0150h

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

DCP_CH1SEMA field descriptions

Field	Description
31–24 -	Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the

Table continues on the next page...

DCP_CH1SEMA field descriptions (continued)

Field	Description
	DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

17.3.22 DCP Channel 1 Status Register (DCP_CH1STAT)

The DCP Channel 1 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

CH1STAT: 0x160

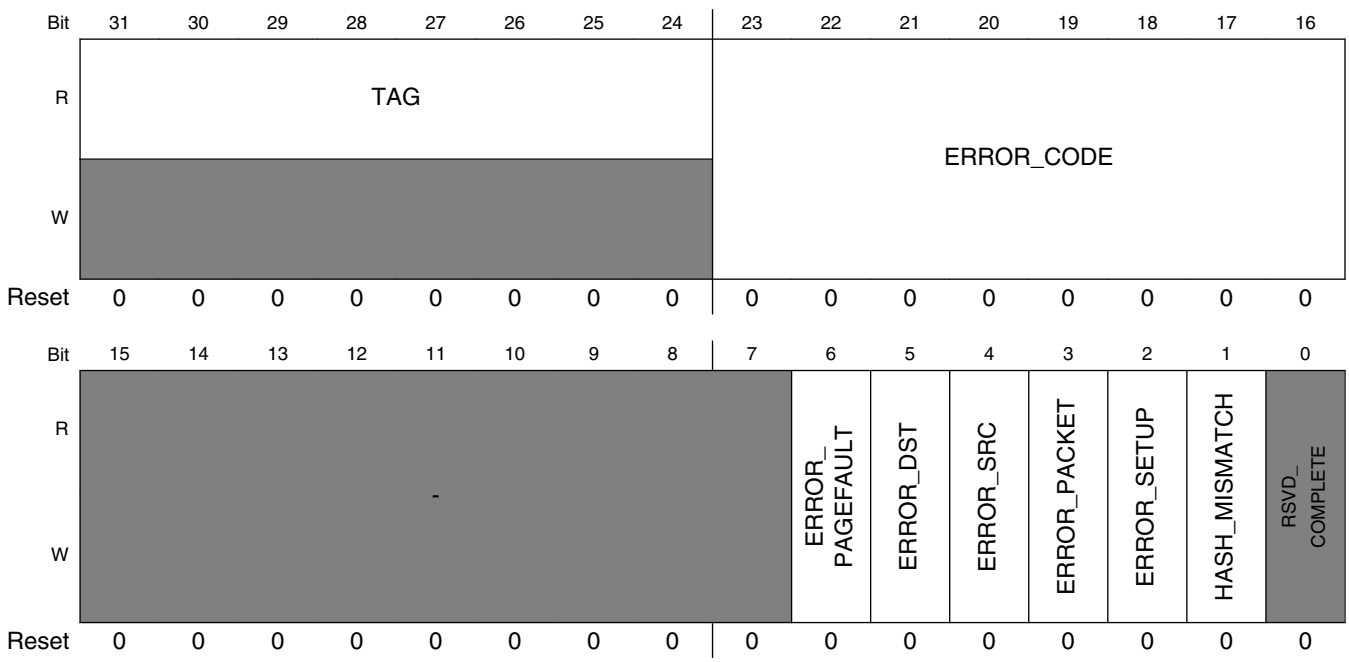
CH1STAT_SET: 0x164

CH1STAT_CLR: 0x168

CH1STAT_TOG: 0x16C

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: DCP_CH1STAT is 0h base + 160h offset = 0000_0160h



DCP_CH1STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurs when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

17.3.23 DCP Channel 1 Options Register (DCP_CH1OPTS)

The DCP Channel 1 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

DCP_CH1OPTS: 0x170

DCP_CH1OPTS_SET: 0x174

CH1OPTS_CLR: 0x178

CH1OPTS_TOG: 0x17C

The options register can be used to control optional features of the channels.

Address: DCP_CH1OPTS is 0h base + 170h offset = 0000_0170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RECOVERY_TIMER																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH1OPTS field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

17.3.24 DCP Channel 2 Command Pointer Address Register (DCP_CH2CMDPTR)

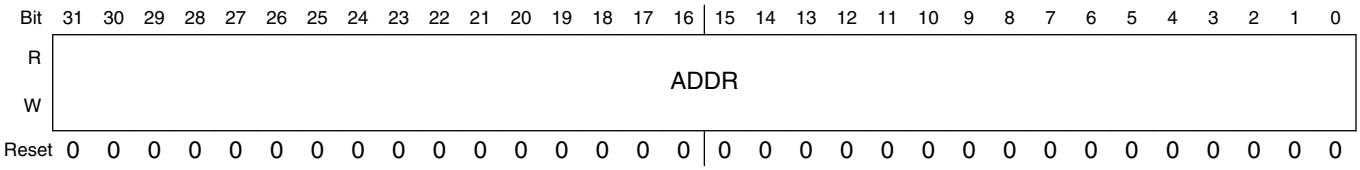
The DCP channel 2 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 2 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
            DCP_CHn_CMDPTR_WR(2, v);   // Write channel 2 command pointer
            pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(2);  // Read current command
pointer
```

Address: DCP_CH2CMDPTR is 0h base + 180h offset = 0000_0180h



DCP_CH2CMDPTR field descriptions

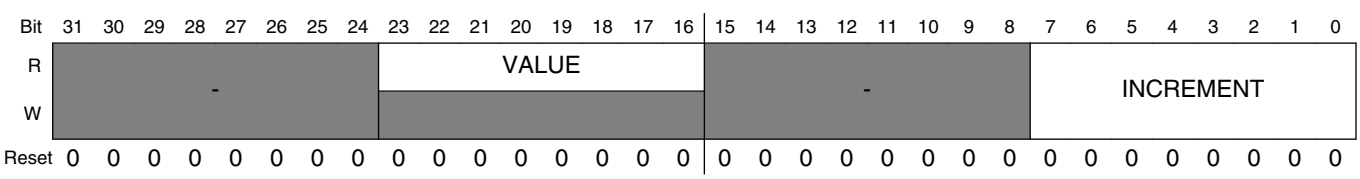
Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 2.

17.3.25 DCP Channel 2 Semaphore Register (DCP_CH2SEMA)

The DCP Channel 2 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjuction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: DCP_CH2SEMA is 0h base + 190h offset = 0000_0190h



DCP_CH2SEMA field descriptions

Field	Description
31–24 -	Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

17.3.26 DCP Channel 2 Status Register (DCP_CH2STAT)

The DCP Channel 2 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

CH2STAT: 0x1A0

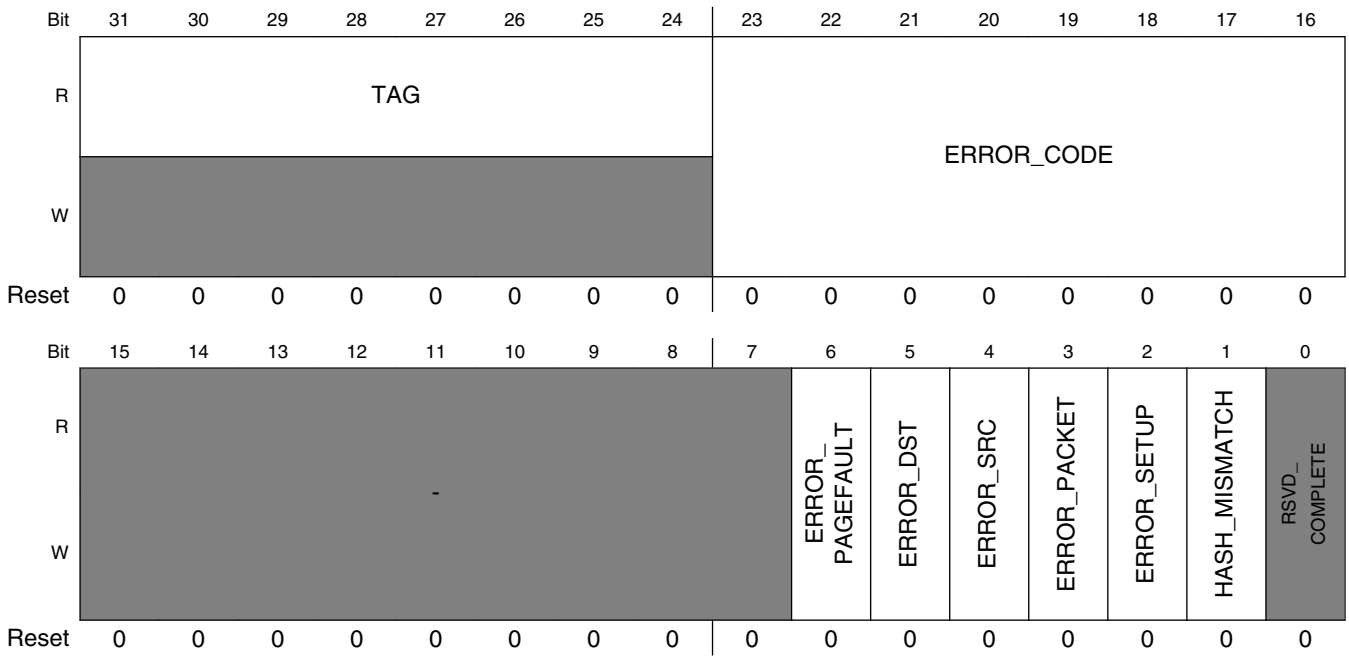
CH2STAT_SET: 0x1A4

CH2STAT_CLR: 0x1A8

CH2STAT_TOG: 0x1AC

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: DCP_CH2STAT is 0h base + 1A0h offset = 0000_01A0h



DCP_CH2STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.

Table continues on the next page...

DCP_CH2STAT field descriptions (continued)

Field	Description
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

17.3.27 DCP Channel 2 Options Register (DCP_CH2OPTS)

The DCP Channel 2 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

CH2OPTS: 0x1B0

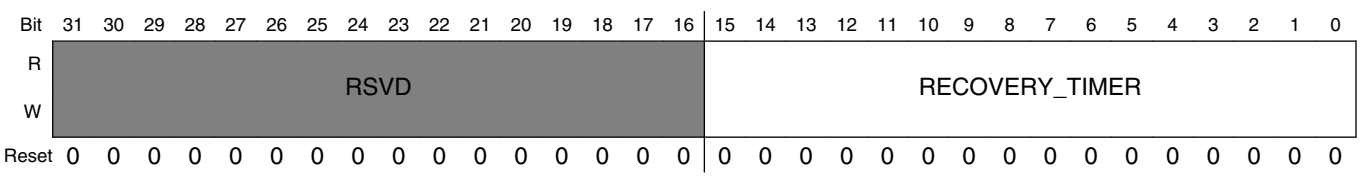
CH2OPTS_SET: 0x1B4

CH2OPTS_CLR: 0x1B8

CH2OPTS_TOG: 0x1BC

The options register can be used to control optional features of the channels.

Address: DCP_CH2OPTS is 0h base + 1B0h offset = 0000_01B0h



DCP_CH2OPTS field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

17.3.28 DCP Channel 3 Command Pointer Address Register (DCP_CH3CMDPTR)

The DCP channel 3 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 3 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
            DCP_CHn_CMDPTR_WR(3, v);   // Write channel 3 command pointer
            pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(3);  // Read current command
pointer
```

Address: DCP_CH3CMDPTR is 0h base + 1C0h offset = 0000_01C0h

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

DCP_CH3CMDPTR field descriptions

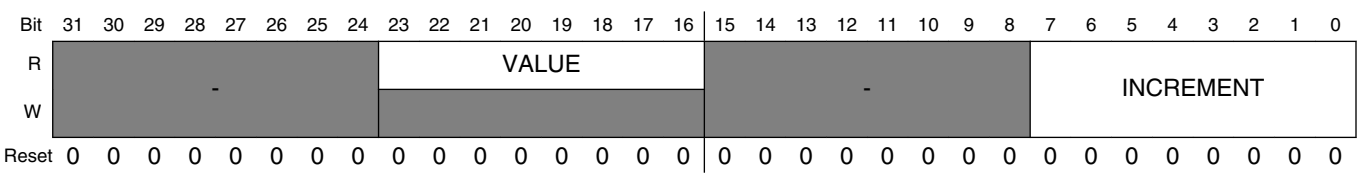
Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 3.

17.3.29 DCP Channel 3 Semaphore Register (DCP_CH3SEMA)

The DCP Channel 3 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjuction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: DCP_CH3SEMA is 0h base + 1D0h offset = 0000_01D0h



DCP_CH3SEMA field descriptions

Field	Description
31–24 -	Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the

Table continues on the next page...

DCP_CH3SEMA field descriptions (continued)

Field	Description
	DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

17.3.30 DCP Channel 3 Status Register (DCP_CH3STAT)

The DCP Channel 3 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

DCP_CH3STAT: 0x1E0

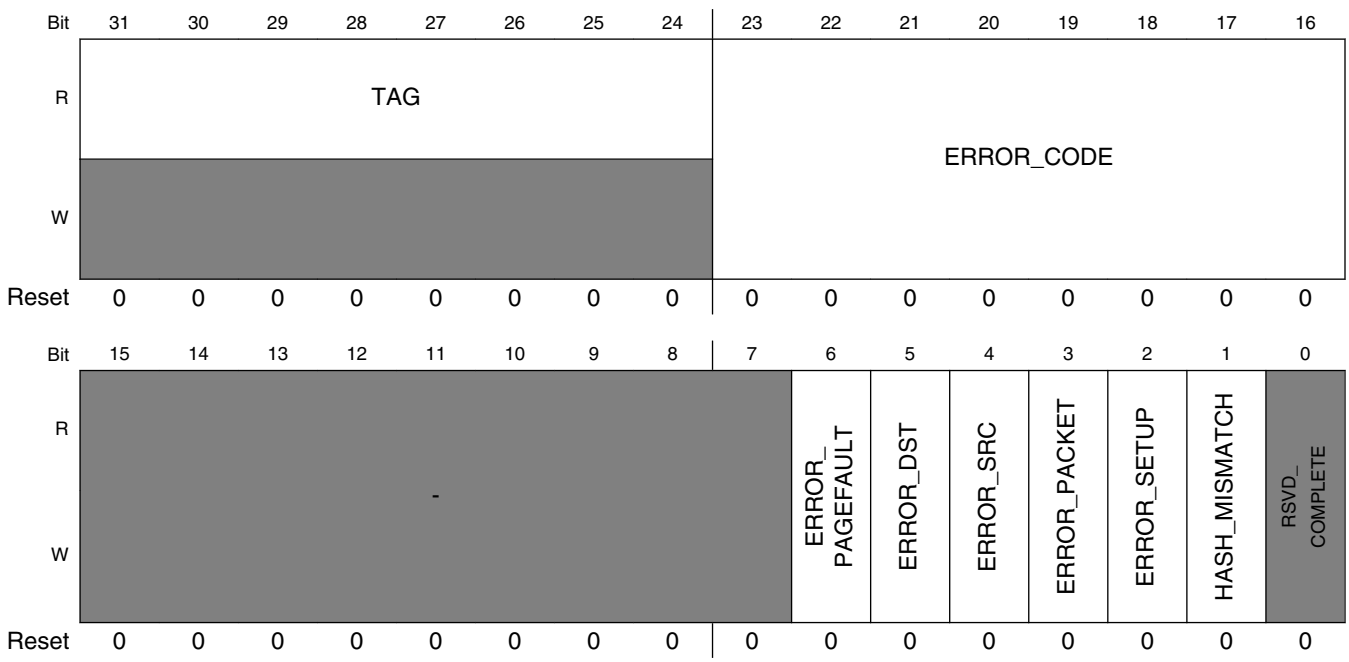
CH3STAT_SET: 0x1E4

CH3STAT_CLR: 0x1E8

CH3STAT_TOG: 0x1EC

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: DCP_CH3STAT is 0h base + 1E0h offset = 0000_01E0h



DCP_CH3STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

17.3.31 DCP Channel 3 Options Register (DCP_CH3OPTS)

The DCP Channel 3 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

DCP_CH3OPTS: 0x1F0

CH3OPTS_SET: 0x1F4

CH3OPTS_CLR: 0x1F8

CH3OPTS_TOG: 0x1FC

The options register can be used to control optional features of the channels.

Address: DCP_CH3OPTS is 0h base + 1F0h offset = 0000_01F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RECOVERY_TIMER																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH3OPTS field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

17.3.32 DCP Debug Select Register (DCP_DBGSELECT)

This register selects a debug register to view.

This register selects debug information to return in the debug data register.

Address: DCP_DBGSELECT is 0h base + 400h offset = 0000_0400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																								INDEX							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_DBGSELECT field descriptions

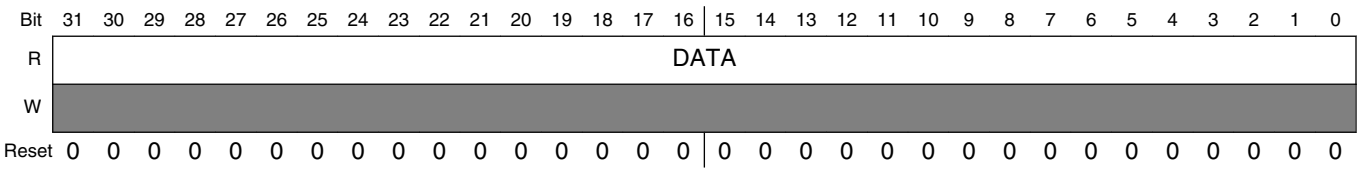
Field	Description
31–8 RSVD	Reserved, always set to zero.
7–0 INDEX	Selects a value to read via the debug data register. 0x01 **CONTROL** — 0x10 **OTPKEY0** — 0x11 **OTPKEY1** — 0x12 **OTPKEY2** — 0x13 **OTPKEY3** —

17.3.33 DCP Debug Data Register (DCP_DBGDATA)

Reading this register returns the debug data value from the selected index.

This register returns the debug data from the selected debug index source.

Address: DCP_DBGDATA is 0h base + 410h offset = 0000_0410h



DCP_DBGDATA field descriptions

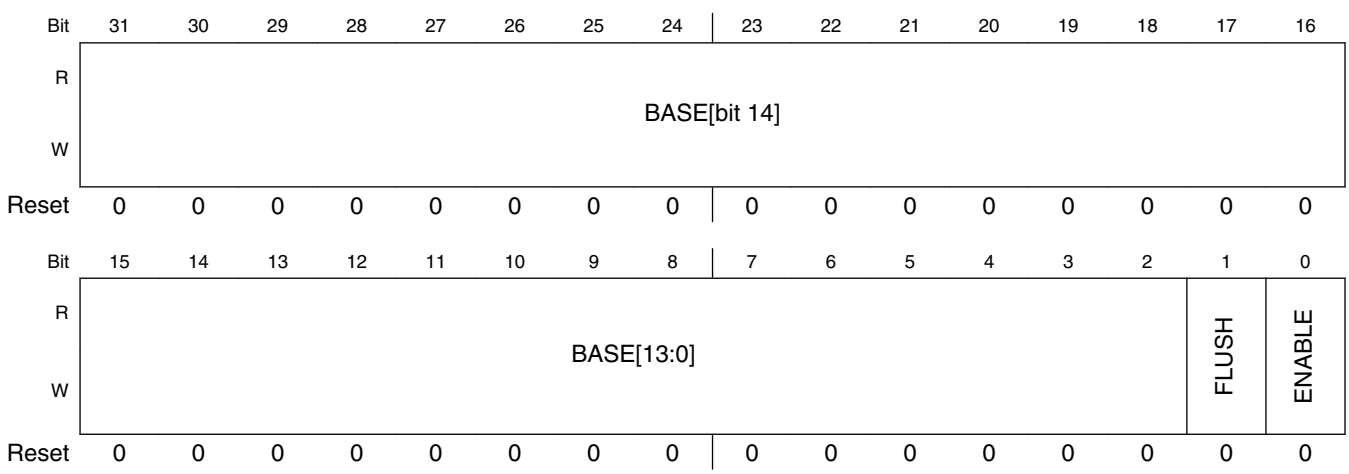
Field	Description
31–0 DATA	Debug Data

17.3.34 DCP Page Table Register (DCP_PAGETABLE)

The DCP Page Table register controls the virtual memory functionality of the DCP. It provides a base address for the page table as well as an enable/disable bit and the ability to flush the cached page table entries.

This register returns the debug data from the selected debug index source.

Address: DCP_PAGETABLE is 0h base + 420h offset = 0000_0420h



DCP_PAGETABLE field descriptions

Field	Description
31–2 BASE	Page Table Base Address. The page table must be word aligned and the pointer should reference a page table in the standard ARM format.
1 FLUSH	Page Table Flush control. To flush the TLB, write this bit to a 1 then back to a 0.
0 ENABLE	Page Table Enable control. Virtual addressing will only be used when this bit is set to a 1. Disabling the page table will not flush any cached entries, so software should write the FLUSH high and enable LOW when updating page tables.

17.3.35 DCP Version Register (DCP_VERSION)

Read-only register indicating implemented version of the DCP.

This register returns the debug data from the selected debug index source.

Address: DCP_VERSION is 0h base + 430h offset = 0000_0430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-onlyl value reflecting the MAJOR version of the design implementation.
23–16 MINOR	Fixed read-onlyl value reflecting the MINOR version of the design implementation.
15–0 STEP	Fixed read-onlyl value reflecting the stepping of version of the design implementation.

Chapter 18
Digital Control (DIGCTL) and On-Chip RAM

18.1 Overview

The digital control block provides miscellaneous control within the top digital block of the chip, including:

- Specifies which version of the eSDHC block should be used. In i.MX50, use the default selected eSDHCv3 version.
- Read/Write delay control for On-Chip RAM
- Speed Sensor functionality control and status.

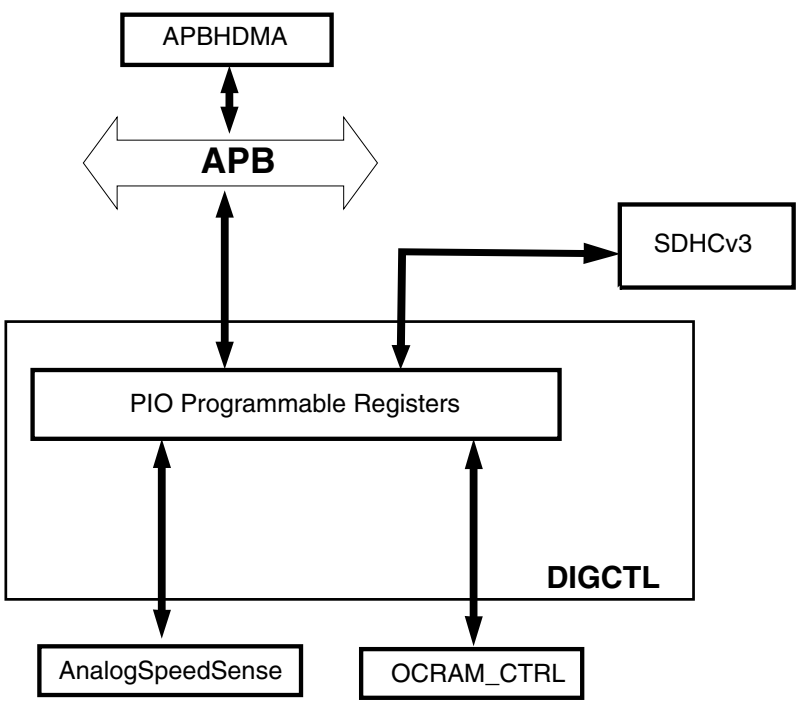


Figure 18-1. Block Diagram

18.2 eSDHC Version control

The ESDHC_VERSION bit of the DIGCTL_CTL register switches between two onboard versions of the SD/SDIO/MMC memory card interface controller, namely eSDHCv3 and uSDHC. By default v3 is set. This bit should only be set when the SDHC controller is idle.

18.3 OCRAM Pipeline Controls

The on-chip 128 Kbyte RAM is comprised of four physical banks of 16 K x 64-bit each. The bits in the DIGCTL_OCRAM register can be used to effect the pipeline delay for reading and writing this memory. These delays can be used to avoid memory timing violations when the OCRAM clock is programmed at high frequencies. Setting the *_EN bits will add 1 cycle of delay to the address or data phases of each read or write transaction to the memory. The corresponding status (STAT) bits indicate whether it is safe to change a delay bit. The enable bit should only be changed when its corresponding status bit is polled as not busy. Please refer to chapter entitled "On-Chip RAM" for a more detailed functional description.

18.4 Speed Sensor Controls

The SoC integrates three silicon speed sensors to measure the performance characteristics of an individual die at its ambient temperature and process parametrics. They each consist of a ring oscillator and a frequency counter. The frequency tracks the silicon performance as it changes in response to changes in operating voltage and temperature. The crystal oscillator is directly used as the precision time base for measuring the frequency of a ring oscillator. The ring oscillator is normally disabled. There is a 32-bit counter connected to each of the ring oscillators that performs the frequency measurement.

Thus, the counter holds the number of cycles the ring oscillator was able to generate during one crystal clock period. The natural frequency of the ring oscillator strongly tracks the silicon process parametrics, i.e., faster silicon processes yield ring oscillators that run faster and thereby yield larger count values. The natural frequency tracks junction temperature effects on silicon speed as well.

The information given by the speed sensor can be used with the silicon temperature and process parameters, which can also be monitored by system software. Freescale can provide a power management application note and firmware that takes full advantage of the on-chip monitoring functions to enable minimum-voltage operation.

To take a speed sensor measurement, first the DIGCTL_SPEEDCTL[SELECT] should be set to a value of either 0, 1 or 2 to select the desired sensor. Then the DIGCTL_SPEEDCTL[CTRL] field should be written three times writng it with the sequence of 00b, 01b and 11b. At this point the DIGCTL_SPEEDSTAT[STATUS] field can be read and will contain the speed sensor count value. This value is only valid when the DIGCTL_SPEEDCTL[CTRL] = 11b. For repeated sensor measurements, the sequence of 00b-01b-11b must be rewritten to the DIGCTL_SPEEDCTL [CTRL] field. For proper measurement, the back-to-back writes to the DIGCTL_SPEEDCTL register should not proceed faster than a frequency of 1.5 MHz.

18.5 Programmable Registers

i.MX50 DIGCTL Register Format Summary

DIGCTL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_4000	DIGCTL Control Register (DIGCTL_CTRL)	32	R/W	C000_0000h	18.5.1/ 844
4100_4004	DIGCTL Control Register (DIGCTL_CTRL_SET)	32	R/W	C000_0000h	18.5.1/ 844
4100_4008	DIGCTL Control Register (DIGCTL_CTRL_CLR)	32	R/W	C000_0000h	18.5.1/ 844
4100_400C	DIGCTL Control Register (DIGCTL_CTRL_TOG)	32	R/W	C000_0000h	18.5.1/ 844
4100_4010	DIGCTL OCRAM Register (DIGCTL_OCRAM)	32	R/W	0000_000Fh	18.5.2/ 845
4100_4014	DIGCTL OCRAM Register (DIGCTL_OCRAM_SET)	32	R/W	0000_000Fh	18.5.2/ 845
4100_4018	DIGCTL OCRAM Register (DIGCTL_OCRAM_CLR)	32	R/W	0000_000Fh	18.5.2/ 845
4100_401C	DIGCTL OCRAM Register (DIGCTL_OCRAM_TOG)	32	R/W	0000_000Fh	18.5.2/ 845
4100_4020	Transistor Speed Control Register (DIGCTL_SPEEDCTL)	32	R/W	0000_0000h	18.5.3/ 847

Table continues on the next page...

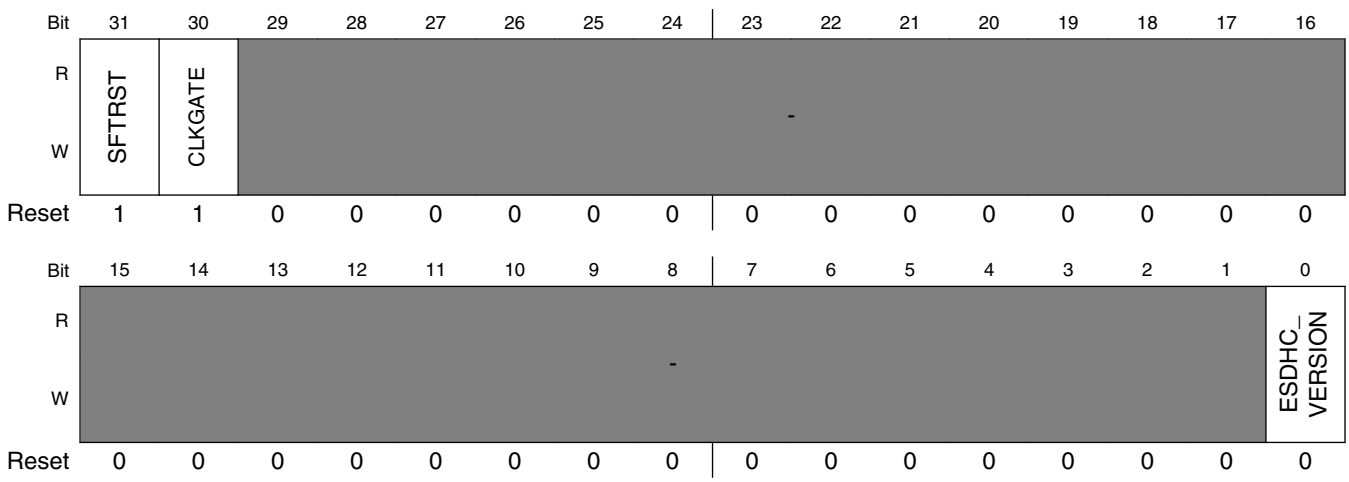
DIGCTL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_4024	Transistor Speed Control Register (DIGCTL_SPEEDCTL_SET)	32	R/W	0000_0000h	18.5.3/ 847
4100_4028	Transistor Speed Control Register (DIGCTL_SPEEDCTL_CLR)	32	R/W	0000_0000h	18.5.3/ 847
4100_402C	Transistor Speed Control Register (DIGCTL_SPEEDCTL_TOG)	32	R/W	0000_0000h	18.5.3/ 847
4100_4030	Transistor Speed Status Register (DIGCTL_DIGCTL_SPEEDSTAT)	32	R	0000_0000h	18.5.4/ 847

18.5.1 DIGCTL Control Register (DIGCTL_CTRL*n*)

The DIGCTL Control Register provides overall control of various functions throughout the digital portion of the chip. This register controls various functions throughout the digital portion of the chip.

Addresses: DIGCTL_CTRL is 4100_4000h base + 0h offset = 4100_4000h
DIGCTL_CTRL_SET is 4100_4000h base + 4h offset = 4100_4004h
DIGCTL_CTRL_CLR is 4100_4000h base + 8h offset = 4100_4008h
DIGCTL_CTRL_TOG is 4100_4000h base + Ch offset = 4100_400Ch



DIGCTL_CTRL*n* field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal DIGCTL operation. Set this bit to one (default) to disable clocking with the DIGCTL and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the DIGCTL block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.

Table continues on the next page...

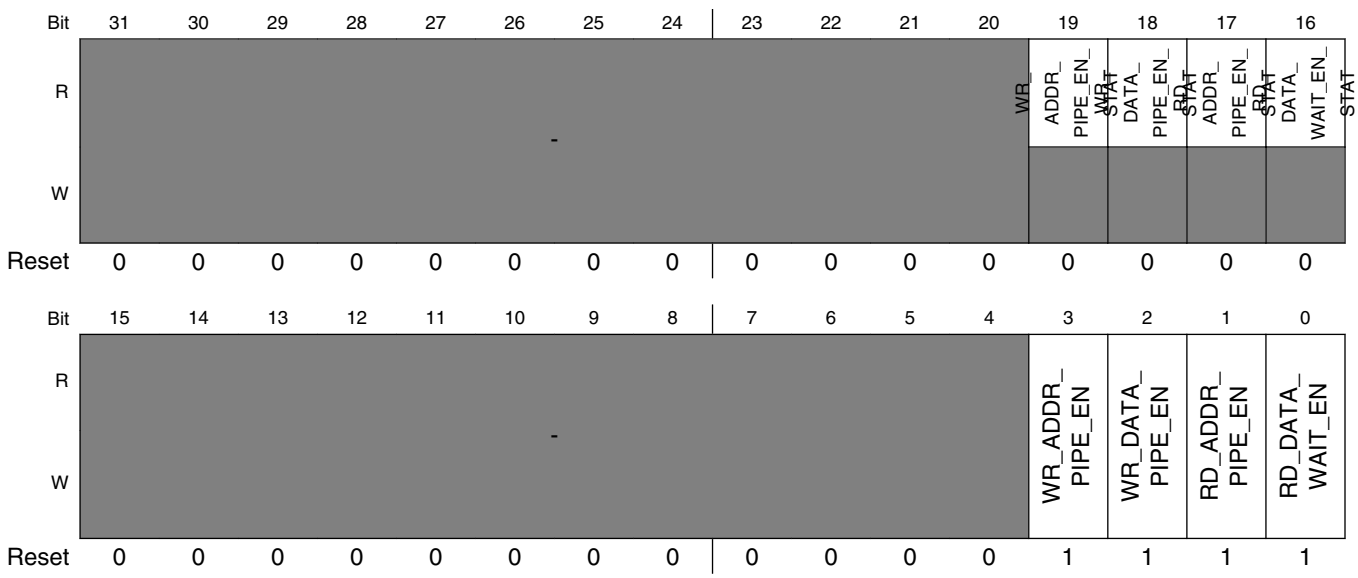
DIGCTL_CTRL*n* field descriptions (continued)

Field	Description
29–1 -	Reserved.
0 ESDHC_ VERSION	Specifies which version of the eSDHC block to use. This setting will enable the selected version's functionality and register access. 0x0 Use ESDHCv3 0x1 Use uSDHC

18.5.2 DIGCTL OCRAM Register (DIGCTL_OCRAM*n*)

The DIGCTL OCRAM Register contains control bits and reports read-only status for the OCRAM control block.

Addresses: DIGCTL_OCRAM is 4100_4000h base + 10h offset = 4100_4010h
DIGCTL_OCRAM_SET is 4100_4000h base + 14h offset = 4100_4014h
DIGCTL_OCRAM_CLR is 4100_4000h base + 18h offset = 4100_4018h
DIGCTL_OCRAM_TOG is 4100_4000h base + 1Ch offset = 4100_401Ch



DIGCTL_OCRAM*n* field descriptions

Field	Description
31–20 -	Reserved.
19 WR_ADDR_ PIPE_EN_STAT	Configuration status of OCRAM timing parameter: Write Address Pipeline Enable Status 0x0 NOT_BUSY 0x1 BUSY

Table continues on the next page...

DIGCTL_OCRAM*n* field descriptions (continued)

Field	Description
18 WR_DATA_ PIPE_EN_STAT	Configuration status of OCRAM timing parameter: Write Data Pipeline Enable Status 0x0 NOT_BUSY 0x1 BUSY
17 RD_ADDR_ PIPE_EN_STAT	Configuration status of OCRAM timing parameter: Read Address Pipeline Enable Status 0x0 NOT_BUSY 0x1 BUSY
16 RD_DATA_ WAIT_EN_STAT	Configuration status of OCRAM timing parameter: Read Data Wait Enable Status 0x0 NOT_BUSY 0x1 BUSY
15–4 -	Reserved.
3 WR_ADDR_ PIPE_EN	OCRAM timing parameter: Write Address Pipeline Enable 0x0 DISABLE 0x1 ENABLE
2 WR_DATA_ PIPE_EN	OCRAM timing parameter: Write Data Pipeline Enable 0x0 DISABLE 0x1 ENABLE
1 RD_ADDR_ PIPE_EN	OCRAM timing parameter: Read Address Pipeline Enable 0x0 DISABLE 0x1 ENABLE
0 RD_DATA_ WAIT_EN	OCRAM timing parameter: Read Data Wait Enable 0x0 DISABLE 0x1 ENABLE

18.5.3 Transistor Speed Control Register (DIGCTL_SPEEDCTL*n*)

This register contains the setup and controls needed to measure silicon speed.

Addresses: DIGCTL_SPEEDCTL is 4100_4000h base + 20h offset = 4100_4020h
DIGCTL_SPEEDCTL_SET is 4100_4000h base + 24h offset = 4100_4024h
DIGCTL_SPEEDCTL_CLR is 4100_4000h base + 28h offset = 4100_4028h
DIGCTL_SPEEDCTL_TOG is 4100_4000h base + 2Ch offset = 4100_402Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								SELECT				-		CTRL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DIGCTL_SPEEDCTL*n* field descriptions

Field	Description
31–8 -	-
7–4 SELECT	Speed Sensor Status Select. This defines the meaning on the STATUS field in this register. 0x0 **STAT_SENSOR0** — Sensor 0 0x1 **STAT_SENSOR1** — Sensor 1
3–2 -	-
1–0 CTRL	Speed Control bits. 00: Speed sensor off, 0b01: Speed sensor enabled, 11: Enable speed sensor measurement. Every time a measurement is taken, the sequence of 0x00 ; 01 ; 11 must be repeated. This sequence should proceed no faster than 1.5 MHz to ensure proper operation.

18.5.4 Transistor Speed Status Register (DIGCTL_DIGCTL_SPEEDSTAT)

This register contains the measurment status for speed control tests.

Address: DIGCTL_DIGCTL_SPEEDSTAT is 4100_4000h base + 30h offset = 4100_4030h

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

DIGCTL_DIGCTL_SPEEDSTAT field descriptions

Field	Description
31–0 STATUS	Result from the speed sensor. This result is only valid when SPEEDCTRL=0b11; otherwise this field contains debug information from the switching DC-DC converter.

Chapter 19
DPLL Controller (DPLLC)

19.1 Overview

The Digital Phase-Locked Loop Control (DPLLC) is the control interface to a DPLL. It generates the control signals required for the DPLL operations.

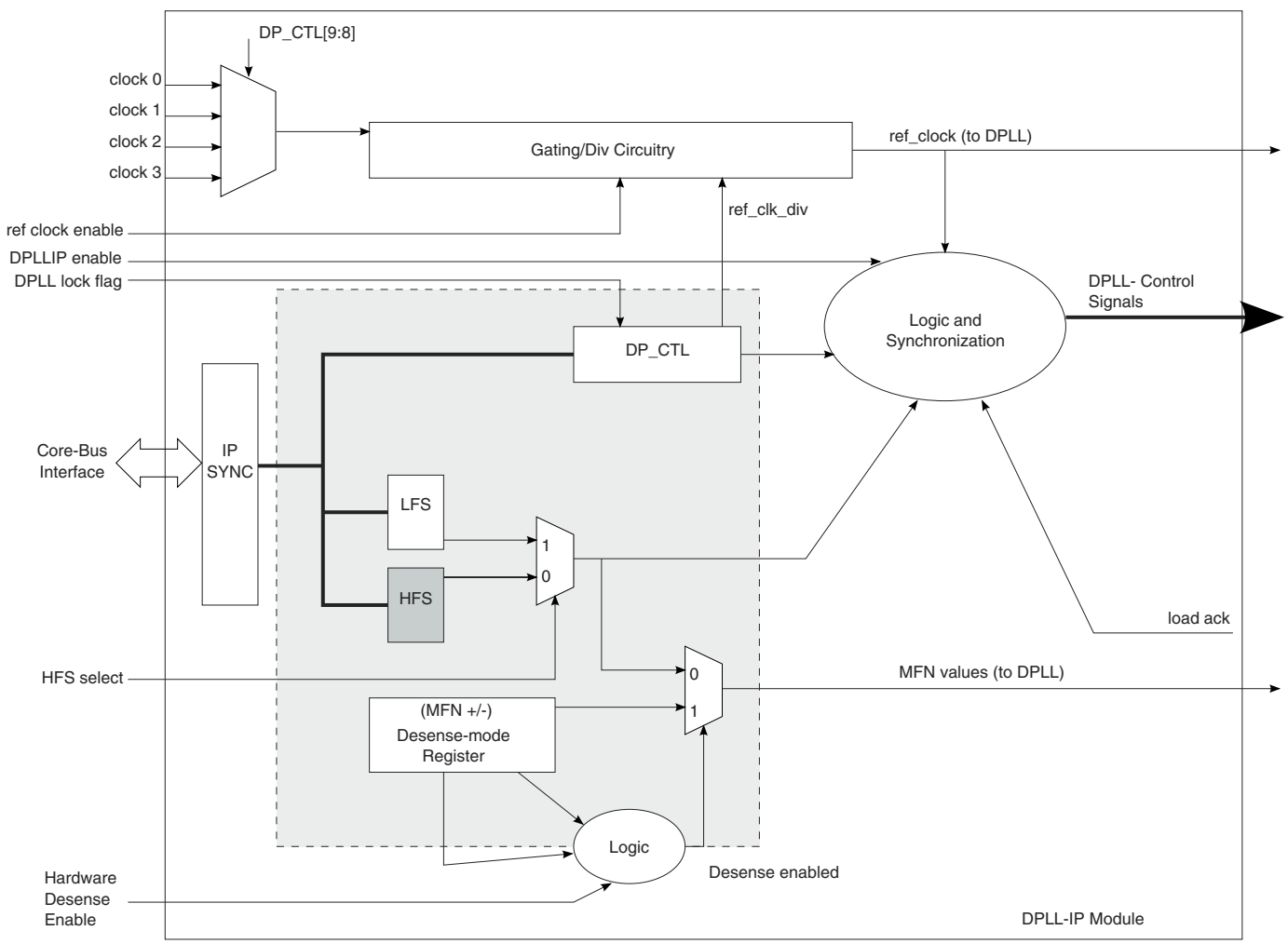


Figure 19-1. Block Diagram of the DPLLC

19.1.1 Feature Description

Key features of this module include:

- Controls the operation of DPLLs.
- Provides controlled phase modulation of clocks to reduce receiver desensitization using desense circuit for DPLLs.
- If auto restart is enabled (AREN bit of the DPLL Config Register is '1'), a restart sequence is issued automatically whenever one of the DPLL registers (DPLLC_DP_CTL, DP_OP, DPLLC_DP_MFD or DPLLC_DP_HFS_OP, DP_HFS_MFD) is written. Otherwise, software needs to set the restart bit of DPLL Control Register manually.

19.1.2 Modes and Operation

The DPLLC supports the following modes of operation:

- Normal Mode
- DPLL Desense Mode
- DVFS Support: HFS Mode

Refer to Operations for DPLLC operations.

19.2 Functional Description

The DPLLC module is mainly divided in the following blocks as shown in Figure 19-1.

- Register read/write: Core bus reads and writes the registers of the module.
- Reference clock selection circuitry: This circuitry comprises of clock selection mux, gating circuit and divide by 2 logic. The reference clock is selected from one of the four external input clocks based on register configuration. This reference clock goes to DPLL. See Figure 19-2 for more details.
- DPLL restart logic and synchronization of data: Based on input signal, registers corresponding to low frequency or high frequency modes are selected. Data goes to the DPLL port, is synchronized on the reference clock.
- Desense mode logic: Desense mode is enabled based on DPLLC configuration. The output to DPLL toggles between predefined values (configured in DPLLC registers)

which determines the DPLL reference clock frequency. See DPLL Desense Mode for mode details.

- Desense mode logic: Desense mode is enabled based on DPLLC configuration. The output to DPLL toggles between predefined values (configured in DPLLC registers) which determines the DPLL reference clock frequency. See DPLL Desense Mode for mode details.

19.2.1 Operating Modes

19.2.1.1 Normal Mode

In DPLLC normal mode out of the four external input clocks, one is selected depending upon DPLLC register configuration. This is followed by a divider circuitry and then by a gating circuitry as shown below. If reference clock enable signal is HIGH, the clock is passed, else it is gated to a value'1'. Refer to Figure 19-2 for details.

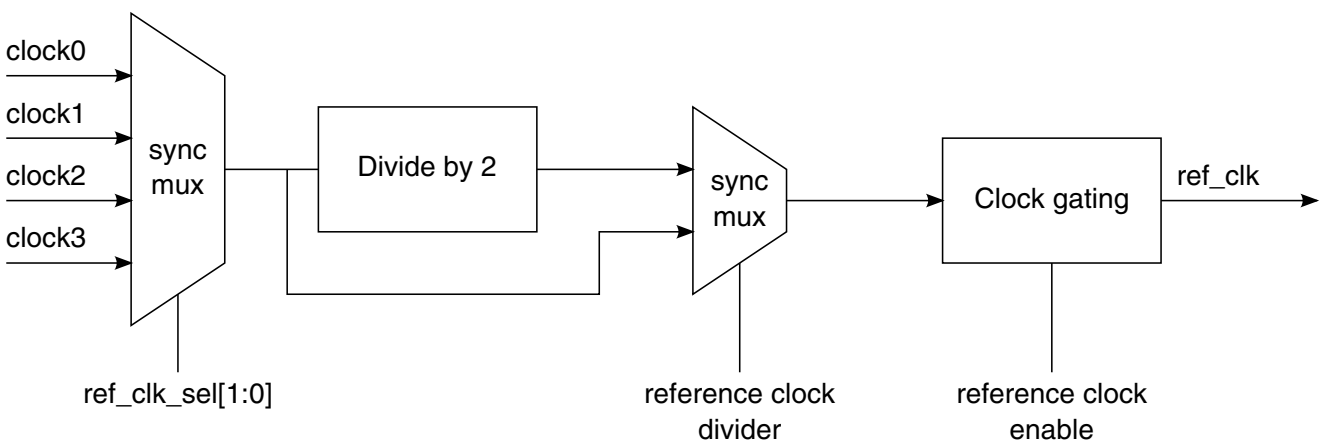


Figure 19-2. Clock MUXing and Gating/Divider Circuitry

19.2.1.2 DPLL Desense Mode

Radiation of clock harmonics causes receiver desensitization. Controlled phase modulation of the clocks coming out of the DPLL will reduce this desensitization. The desense circuit when enabled provides a MFN value that toggles between two programmed MFN_PLUS and MFN_MINUS values for a programmed toggle frequency. This will help achieve the phase modulation of the DPLL clocks.

The disable bit in the DPLLC_DP_MFN_TOGC register should be clear in order to enable the desense logic. Once the disable bit is cleared, desense can be enabled by programming the TOG_EN bit of the DPLLC_DP_MFN_TOGC register for the corresponding PLL. (The desense logic can be activated either by writing to TOG_EN, or by asserting hardware desense enable signal).

Once desense is enabled, the DPLLC module provides the DPLL with a MFN value that switches between MFN_PLUS and MFN_MINUS for n number of correct reference clock cycles. MFN_PLUS and MFN_MINUS are programmed by writing to DPLLC_DP_MFNPLUS and DPLLC_DP_MFNMINUS registers. TOG_CNT (Bits [15:0] of the DPLLC_DP_MFN_TOGC register) determines the toggle rate n. The ref_pll receives a MFN value equal to MFN_PLUS for the first n/2 cycles and MFN_MINUS value for the remaining n/2 cycles.

In the middle of the dither sequence, if the desense mode is disabled through toggle enable (disable) bit or desense hardware request, the circuit completes the ongoing MFN_PLUS, MFN_MINUS cycle before exiting the desense mode. PLL runs on normal MFN after the completion of desense mode. If desense disables and enables again in the middle of the sequence, toggle counter will not stop the current cycle. Only even values for the counter are allowed. If the counter is programmed to an odd value, the counter value will default to a value one less from the odd value.

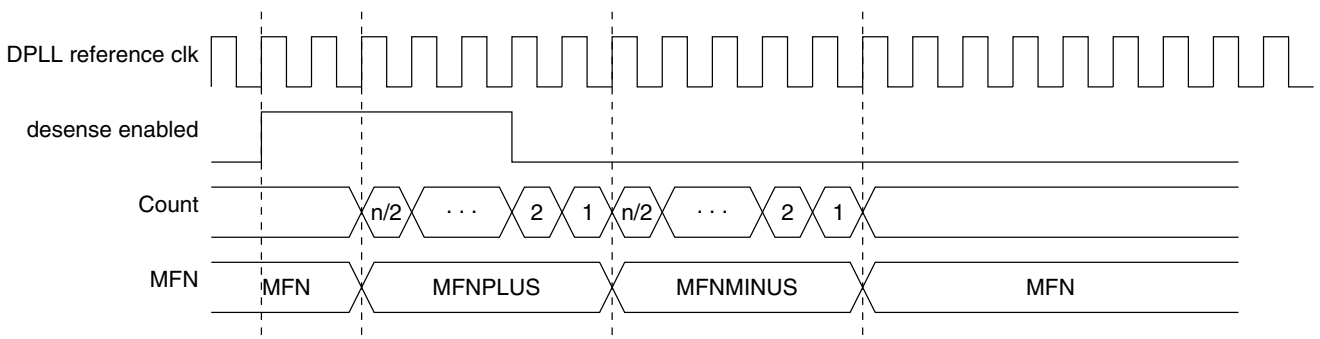


Figure 19-3. Desense Mode MFN Timing

When in desense mode, the DPLLC generates an automatic load request to the PLL whenever the MFN value changes. This signal stays asserted till the load acknowledge signal from the PLL is received. The load request asserts for a minimum of 3 reference clock cycles. Hence for a counter value of 6, where the MFN value toggles between MFN_PLUS and MFN_MINUS every 3 reference clock cycles, the load request may not reflect the updated MFN value and therefore, there is a possibility that the new MFN values are not loaded. Hence it is recommended that the counter be programmed for value greater than 8.

It is required that the toggle counter values not change after the toggle enable is asserted. If toggle counter value changes, the functioning of the desense circuit may be unpredictable.

When any of the 3 MFN values, that is, MFN_PLUS, MFN_MINUS, or MFN values change after the desense mode is enabled, they are buffered till the desense mode is disabled. Therefore to update the MFN_PLUS and MFN_MINUS from the next sequence, desense must be disabled and enabled again.

The desense status register DPLLC Desense Status Register (DPLLC_DESTAT) shows the status of the toggle_enable and the MFN value sent to the DPLL.

19.2.1.3 DVFS Support: HFS Mode

DPLL supports DVFS settings through an alternate set of registers which can be used to store high frequency settings corresponding to high voltage. This alternate set of HFS registers is used to restart the PLL with high frequency settings stored in HFS registers whenever system detects a voltage change from low to high level. Following diagram describes this configuration.

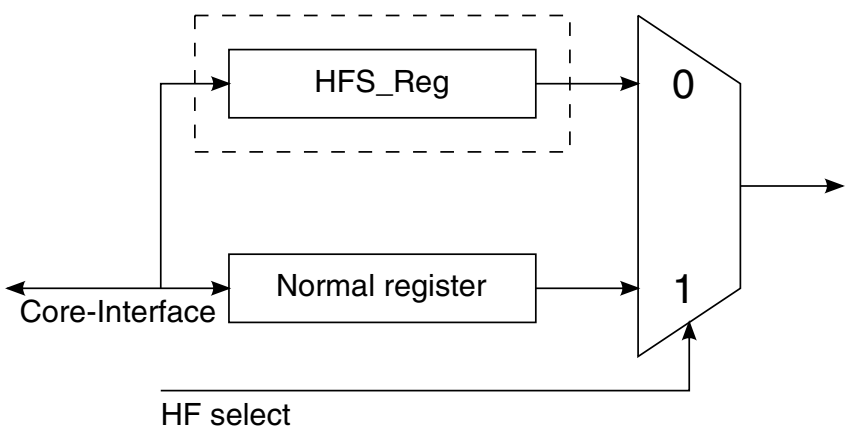


Figure 19-4. DVFS Support for High Frequency Settings

19.2.2 Operations

19.2.2.1 DPLLC_DP_CTL, DPLLC_DP_OP, DPLLC_DP_MFD Register Update

The DPLLs require special timing considerations when updating their ports. Whenever one of the DPLL control register DPLLC_DP_CTL, DPLLC_DP_OP, or DPLLC_DP_MFD, or the shadow registers DPLLC_DP_HFS_OP or DPLLC_DP_HFS_MFD is written (even if the bits don't change), a restart is issued to the DPLL before the actual control bits are sent to the DPLL. The restart assertion and port update will be synchronous to the reference input clock of the DPLL. Writing to any of these registers will cause following procedure to come into effect.

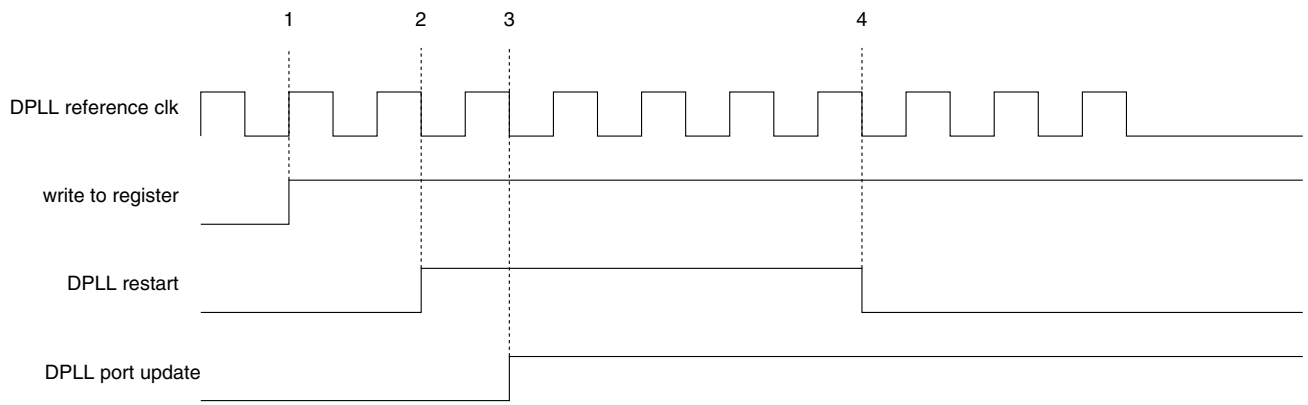


Figure 19-5. DPLL Port Timings

1. Register is written.
2. One clock cycle later restart is asserted.
3. One clock cycle later the appropriate port is updated.
4. Four clock cycles later restart is released.

NOTE

Setting the RST bit of DPLL Control Register will cause the restart (crstrt) signal to assert immediately. In case AREN bit of DPLL Config Register is LOW, the programmer will have to take care of restart of DPLL by setting the RST bit of DPLL Control Register appropriately.

19.2.2.2 DP_MFN Register Update

The MFN bits of DP_MFN register are the only bits in the DPLL that can be changed after the DPLL has been locked without a restart. As shown in diagram below, after writing a new value to MFN, the procedure to update the DPLL is as follows:

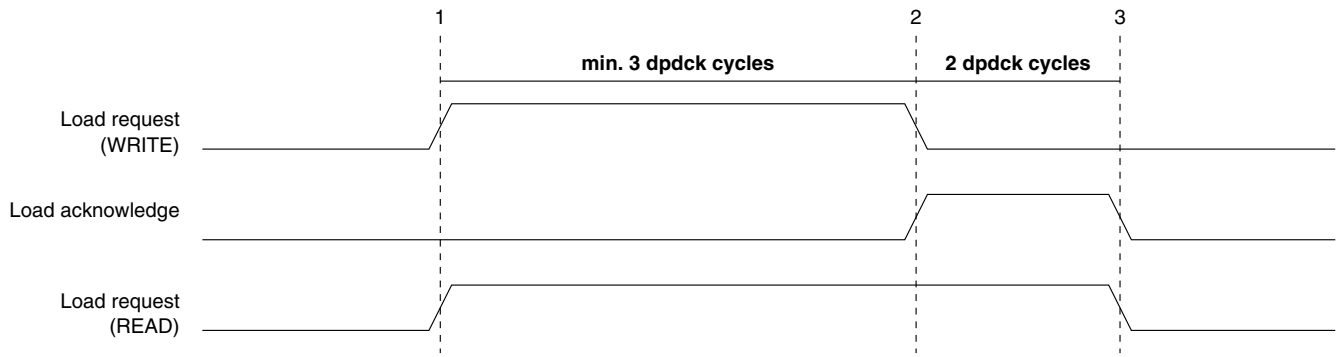


Figure 19-6. MFN Load Request Timings

1. The control bit LDREQ in DP_CONFIG is set. This sends a load request signal to the DPLL.
2. When the DPLL loads the new MFN, it sends a load acknowledge signal back to the DPLL-IP module and clears the write value of load request. At this point load request becomes a status bit, reflecting the state of the acknowledge signal.

NOTE

This write value of load request is a "sticky" bit. When set, it will remain at '1' until a load acknowledgement pulse is received.

3. The read value of load request is automatically cleared when the DPLL has completed updating MFN and has removed the load acknowledge signal, allowing the next load request to take place.

19.2.2.3 Multiple Options for DPLL Control

The DPLLC module provides multiple options for controlling the ON/OFF logic of the DPLLs from the Core and the DSM, making it possible to switch off the PLL by software and still be able to switch it on by DSM. Mul_ctrl (Bit [13]) of DPLLC_DP_CTL register provides this flexibility as shown in the following table.

Table 19-1. Options for DPLL Control

DPLLC Enable	UPEN Bit	mul_ctrl bit (Self Clearing)	DPLL Enable
X	X	1	0
X	0	X	0
posedge	1	0	1
0	X	X	0

As shown in the table above, once the DPLL is switched off by Software using the Mul_ctrl bit, the software cannot turn the PLL on thereafter, unless there is a posedge on the DPLLC enable signal, during DSM entry sequence.

19.2.2.4 Calculating the Output Frequency

The output frequency is given by the formula:

$$f_{dck=2} = 4 \bullet f_{ref} \bullet \left[\frac{\text{MultiplicationFactor}}{\text{PreDividerFactor}} \right]$$

Where:

$$\text{MultiplicationFactor} = MF_I + \frac{MF_N}{MF_D + 1}$$

$$\text{PreDividerFactor} = PDF + 1$$

For normal mode, registers DPLLC_DP_OP, DPLLC_DP_MFN, and DPLLC_DP_MFD are used.

For HFS mode, the shadow registers DPLLC_DP_HFS_OP, DPLLC_DP_HFS_MFN, and DPLLC_DP_HFS_MFD are used.

19.3 Programmable Registers

19.3.1 DPLLC Memory Map/Register Definition

DPLLC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63F8_0000	DPLLC Control Register (DPLLC-1_CTL)	32	R/W	00AA_0222h	19.31.1/ 859
63F8_0004	DPLLC Configuration Register (DPLLC-1_CONFIG)	32	R/W	00AA_0006h	19.31.2/ 861
63F8_0008	DPLLC Operation Register (DPLLC-1_OP)	32	R/W	00AA_0000h	19.31.3/ 862
63F8_000C	DPLLC Multiplication Factor Denominator Register (DPLLC-1_MFD)	32	R/W	00AA_0000h	19.31.4/ 863
63F8_0010	DPLLC Multiplication Factor Numerator Register (DPLLC-1_MFN)	32	R/W	00AA_0000h	19.31.5/ 863
63F8_0014	DPLLC Multiplication Factor Numerator PLUS/MINUS Registers (DPLLC-1_MFNMINUS)	32	R/W	0000_0000h	19.31.6/ 864
63F8_0018	DPLLC Multiplication Factor Numerator PLUS/MINUS Registers (DPLLC-1_MFNPLUS)	32	R/W	0000_0000h	19.31.6/ 864
63F8_001C	DPLLC High Frequency Support, Operation Register (DPLLC-1_HFS_OP)	32	R/W	00AA_0000h	19.31.7/ 865
63F8_0020	DPLLC High Frequency Support Multiplication Factor Denominator Register (DPLLC-1_HFS_MFD)	32	R/W	00AA_0000h	19.31.8/ 866
63F8_0024	DPLLC High Frequency Support Multiplication Factor Numerator Register (DPLLC-1_HFS_MFN)	32	R/W	00AA_0000h	19.31.9/ 867
63F8_0028	DPLLC Multiplication Factor Numerator Toggle Control Register (DPLLC-1_MFN_TOGC)	32	R/W	00AA_0000h	19.31.10/ 867
63F8_002C	DPLLC Desense Status Register (DPLLC-1_DESTAT)	32	R	00AA_0000h	19.31.11/ 869
63F8_4000	DPLLC Control Register (DPLLC-2_CTL)	32	R/W	00AA_0222h	19.31.1/ 859
63F8_4004	DPLLC Configuration Register (DPLLC-2_CONFIG)	32	R/W	00AA_0006h	19.31.2/ 861
63F8_4008	DPLLC Operation Register (DPLLC-2_OP)	32	R/W	00AA_0000h	19.31.3/ 862
63F8_400C	DPLLC Multiplication Factor Denominator Register (DPLLC-2_MFD)	32	R/W	00AA_0000h	19.31.4/ 863
63F8_4010	DPLLC Multiplication Factor Numerator Register (DPLLC-2_MFN)	32	R/W	00AA_0000h	19.31.5/ 863
63F8_4014	DPLLC Multiplication Factor Numerator PLUS/MINUS Registers (DPLLC-2_MFNMINUS)	32	R/W	0000_0000h	19.31.6/ 864
63F8_4018	DPLLC Multiplication Factor Numerator PLUS/MINUS Registers (DPLLC-2_MFNPLUS)	32	R/W	0000_0000h	19.31.6/ 864
63F8_401C	DPLLC High Frequency Support, Operation Register (DPLLC-2_HFS_OP)	32	R/W	00AA_0000h	19.31.7/ 865

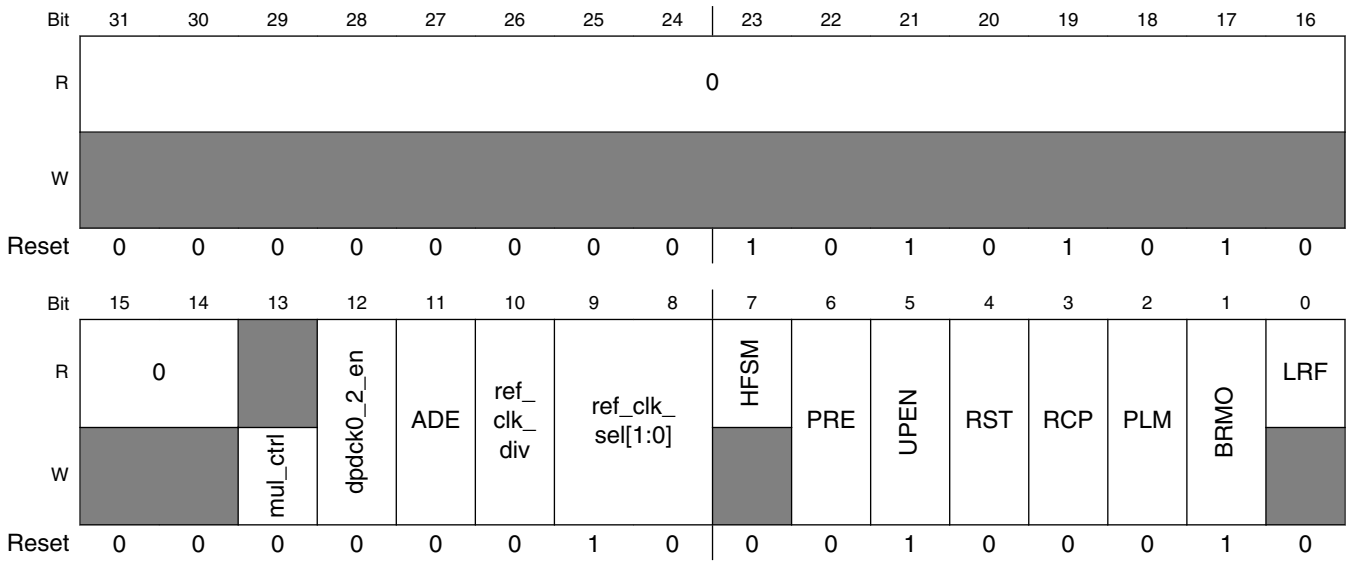
Table continues on the next page...

DPLLC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63F8_4020	DPLLC High Frequency Support Multiplication Factor Denominator Register (DPLLC-2_HFS_MFD)	32	R/W	00AA_0000h	19.31.8/ 866
63F8_4024	DPLLC High Frequency Support Multiplication Factor Numerator Register (DPLLC-2_HFS_MFN)	32	R/W	00AA_0000h	19.31.9/ 867
63F8_4028	DPLLC Multiplication Factor Numerator Toggle Control Register (DPLLC-2_MFN_TOGC)	32	R/W	00AA_0000h	19.31.10/ 867
63F8_402C	DPLLC Desense Status Register (DPLLC-2_DESTAT)	32	R	00AA_0000h	19.31.11/ 869
63F8_8000	DPLLC Control Register (DPLLC-3_CTL)	32	R/W	00AA_0222h	19.31.1/ 859
63F8_8004	DPLLC Configuration Register (DPLLC-3_CONFIG)	32	R/W	00AA_0006h	19.31.2/ 861
63F8_8008	DPLLC Operation Register (DPLLC-3_OP)	32	R/W	00AA_0000h	19.31.3/ 862
63F8_800C	DPLLC Multiplication Factor Denominator Register (DPLLC-3_MFD)	32	R/W	00AA_0000h	19.31.4/ 863
63F8_8010	DPLLC Multiplication Factor Numerator Register (DPLLC-3_MFN)	32	R/W	00AA_0000h	19.31.5/ 863
63F8_8014	DPLLC Multiplication Factor Numerator PLUS/MINUS Registers (DPLLC-3_MFNMINUS)	32	R/W	0000_0000h	19.31.6/ 864
63F8_8018	DPLLC Multiplication Factor Numerator PLUS/MINUS Registers (DPLLC-3_MFNPLUS)	32	R/W	0000_0000h	19.31.6/ 864
63F8_801C	DPLLC High Frequency Support, Operation Register (DPLLC-3_HFS_OP)	32	R/W	00AA_0000h	19.31.7/ 865
63F8_8020	DPLLC High Frequency Support Multiplication Factor Denominator Register (DPLLC-3_HFS_MFD)	32	R/W	00AA_0000h	19.31.8/ 866
63F8_8024	DPLLC High Frequency Support Multiplication Factor Numerator Register (DPLLC-3_HFS_MFN)	32	R/W	00AA_0000h	19.31.9/ 867
63F8_8028	DPLLC Multiplication Factor Numerator Toggle Control Register (DPLLC-3_MFN_TOGC)	32	R/W	00AA_0000h	19.31.10/ 867
63F8_802C	DPLLC Desense Status Register (DPLLC-3_DESTAT)	32	R	00AA_0000h	19.31.11/ 869

19.31.1 DPLLC Control Register (DPLLC*x*_CTL)

Addresses: DPLLC-1_CTL is 63F8_0000h base + 0h offset = 63F8_0000h
DPLLC-2_CTL is 63F8_4000h base + 0h offset = 63F8_4000h
DPLLC-3_CTL is 63F8_8000h base + 0h offset = 63F8_8000h



DPLLC*x*_CTL field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value zero. Reserved
13 mul_ctrl	Multiple Control. This is a self clearing bit cleared on the next posedge of PLL reference clock. Setting this bit deasserts the enable signal to the DPLL. Using this bit instead of UPEN to disable the DPLL gives DSM the control to independently switch on the DPLL. See Multiple Options for DPLL Control for more details. 0 No Effect 1 DPLL disabled
12 dpdck0_2_en	dpdck0_2 Enable. To enable PLL output before divide by 2 flip-flop. It will enable clock dpdck0_2 which will have double the frequency of dpdck/dpgdck clock. 0 Disable dpdck0_2. 1 Enable dpdck0_2.
11 ADE	The output corresponding to this bit is left unconnected for future use.
10 ref_clk_div	Ref Clk Division factor. Decides whether selected reference clock input needs to be divided by two or one. 0 Divide by 1 1 Divide by 2

Table continues on the next page...

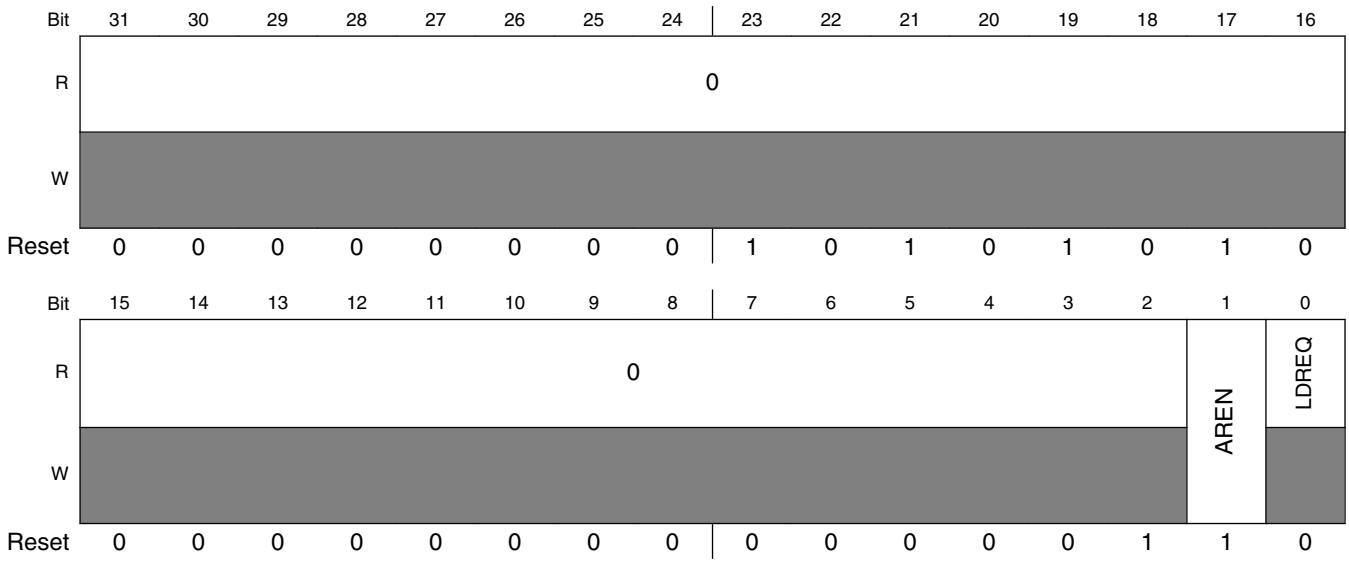
DPLLCx_CTL field descriptions (continued)

Field	Description
9–8 ref_clk_sel[1:0]	ref_clock_select. Used to select the reference clock for DPLL. 00 clock0 is selected. 01 clock1 is selected. 10 clock2 is selected. 11 clock3 is selected.
7 HFSM	HFS-Mode Status bit. Indicates which frequency mode of operation is active. In HFS mode all the shadow registers (DPLLC_DP_HSF_OP, DPLLC_DP_HSF_MFD, DPLLC_DP_HSF_MFN) are used whereas in LFS mode normal registers are used. Reset value of this bit is '0'. 0 LFS mode, normal registers are read. 1 HFS mode is active.
6 PRE	Power Up Reset. Asserts a hard reset to DPLL. 0 Reset cleared. 1 Reset asserted
5 UPEN	PLL Enable. Enables DPLL operation. 0 DPLL disabled. 1 DPLL enabled.
4 RST	Restart. Restarts DPLL. It should not be used as a status bit. 0 DPLL not in restart. 1 DPLL restarted.
3 RCP	Reference Clock Polarity. Selects which edge the chip clock is adjusted to. 0 Positive edge of reference clock. 1 negative edge of reference clock.
2 PLM	Phase Lock Mode. Selects if phase is to be considered (in addition to frequency) during lock-in. 0 DPLL in frequency only lock mode. 1 DPLL in frequency and phase lock mode.
1 BRMO	BRM Order Bit. Binary Rate Modulator order 0 BRM in first order. 1 BRM in second order.
0 LRF	Lock Ready Flag. Shows if the DPLL is in lock. This is a sticky bit. Following events will break the already established lock: (1) DPLLC_DP_CTL, DPLLC_DP_OP, DPLLC_DP_MFD, DPLLC_DP_HSF_OP or DPLLC_DP_HSF_MFD is written (2) Hard reset (3) pll_lvs toggle (4) DPLL enable Reset value of this bit is '0' 0 DPLL not locked. 1 DPLL locked.

19.31.2 DPLLC Configuration Register (DPLLCx_CONFIG)

Writing to this register does not generate an automatic 'restart.'

Addresses: DPLLC-1_CONFIG is 63F8_0000h base + 4h offset = 63F8_0004h
DPLLC-2_CONFIG is 63F8_4000h base + 4h offset = 63F8_4004h
DPLLC-3_CONFIG is 63F8_8000h base + 4h offset = 63F8_8004h



DPLLCx_CONFIG field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 AREN	Auto Restart Enable. If auto restart is enabled, then a restart sequence will be issued automatically whenever a write is performed to one of the DPLL registers (DPLLC_DP_CTL, DPLLC_DP_OP, DPLLC_DP_MFD) in normal mode. In HFS mode, same is true if a write is performed to one of the HFS mode registers (DPLLC_DP_CTL, DPLLC_DP_HFS_OP or DPLLC_DP_HFS_MFD). Otherwise, software will need to set the restart bit manually. Bit 4 of DPLL Control Register. Default Value of this bit is '1'. 0 Auto-restart is disabled 1 Auto-restart is enabled
0 LDREQ	Load Request. Notifies the DPLL that the numerator factor of the fractional part (MFN) has changed. This bit is cleared by an acknowledge from the DPLL. Writing a '0' to it has no effect. Default Value of this bit is '0' 0 DPLL has finished updating MFN. 1 Request that the DPLL updates MFN.

19.31.3 DPLLC Operation Register (DPLLCx_OP)

Addresses: DPLLC-1_OP is 63F8_0000h base + 8h offset = 63F8_0008h
DPLLC-2_OP is 63F8_4000h base + 8h offset = 63F8_4008h
DPLLC-3_OP is 63F8_8000h base + 8h offset = 63F8_8008h

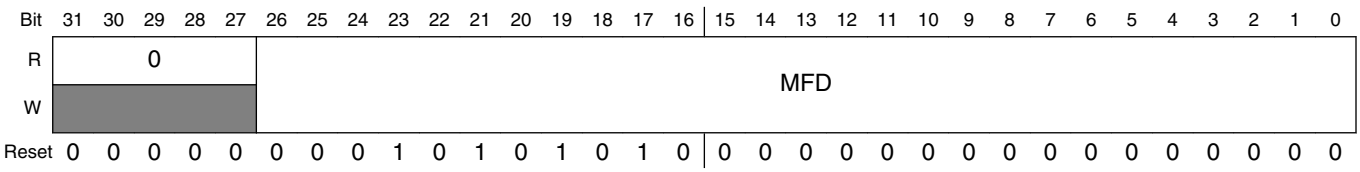
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																								MFI				PDF			
W																																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DPLLCx_OP field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–4 MFI	Multiplication Factor, Integer part (MFI). If MFI[3:0] is written with a value of less than 5, MFI will default to 5. See Calculating the Output Frequency Example settings: 0000 Integer part equals 5 0001 Integer part equals 5 0101 Integer part equals 5 0110 Integer part equals 6 1111 Integer part equals 15
3–0 PDF	Pre-division Factor (PDF). PDF determines the value of the PreDividerFactor used in the output frequency formula. See Calculating the Output Frequency Example settings: 0000 PreDividerFactor equals 1 0001 PreDividerFactor equals 2 0010 PreDividerFactor equals 3 1110 PreDividerFactor equals 15 1111 PreDividerFactor equals 16

19.31.4 DPLLC Multiplication Factor Denominator Register (DPLLCx_MFD)

Addresses: DPLLC-1_MFD is 63F8_0000h base + Ch offset = 63F8_000Ch
DPLLC-2_MFD is 63F8_4000h base + Ch offset = 63F8_400Ch
DPLLC-3_MFD is 63F8_8000h base + Ch offset = 63F8_800Ch



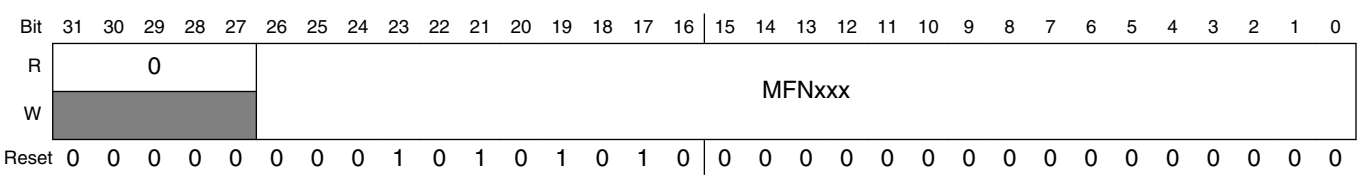
DPLLCx_MFD field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved.
26–0 MFD	Multiplication Factor Denominator Bits. The MFD[26:0] bits, in a 2's complement format, give the denominator of the fractional part. See Calculating the Output Frequency **NOTE:** Bit 26 is always 0. MFD should be in a range from 0 to 67108863; otherwise, the output clock frequency will differ from the desired frequency. Example settings: 0x000000 MFD equals 0 0x000001 MFD equals 1 0x000002 MFD equals 2 0x3FFFFFE MFD equals 67108862 0x3FFFFFF MFD equals 67108863

19.31.5 DPLLC Multiplication Factor Numerator Register (DPLLCx_MFN)

MFNxxx denotes the registers DPLLC_DP_MFN, DPLLC_DP_MFNMINUS and DPLLC_DP_MFNPLUS.

Addresses: DPLLC-1_MFN is 63F8_0000h base + 10h offset = 63F8_0010h
DPLLC-2_MFN is 63F8_4000h base + 10h offset = 63F8_4010h
DPLLC-3_MFN is 63F8_8000h base + 10h offset = 63F8_8010h



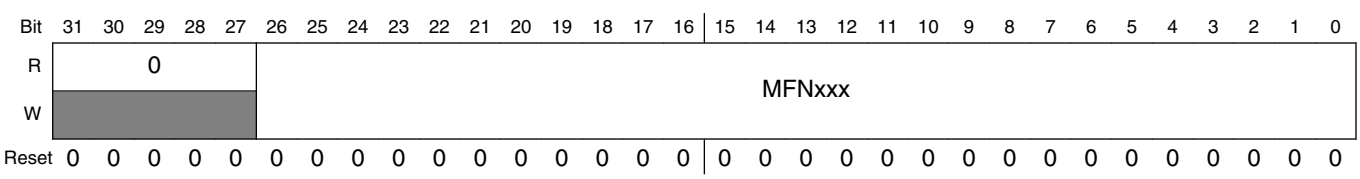
DPLLCx_MFN field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved.
26–0 MFNxxx	Multiplication Factor Numerator Bits (nominal, "MINUS", or "PLUS"). The MFNxxx[26:0] bits, in a 2's complement format, give the numerator of the fractional part. See Calculating the Output Frequency **NOTE:** MFN should be in a range from -67108864 to 67108863. If the absolute value of the MFN is larger than MFD, the output clock frequency will differ from the desired frequency. Example settings: 0x4000000 MFN equals -67108864 0x4000001 MFN equals -67108863 0x4000002 MFN equals -67108862 0xFFFFFFF MFN equals -1 0x0000000 MFN equals 0 (default) 0x0000001 MFN equals 1 0x3FFFFFE MFN equals 67108862 0x3FFFFFF MFN equals 67108863

19.31.6 DPLLC Multiplication Factor Numerator PLUS/MINUS Registers (DPLLCx_MFNn)

MFNxxx denotes the registers DPLLC_DP_MFN, DPLLC_DP_MFNMINUS and DPLLC_DP_MFNPLUS.

Addresses: DPLLC-1_MFNMINUS is 63F8_0000h base + 14h offset = 63F8_0014h
DPLLC-1_MFNPLUS is 63F8_0000h base + 18h offset = 63F8_0018h
DPLLC-2_MFNMINUS is 63F8_4000h base + 14h offset = 63F8_4014h
DPLLC-2_MFNPLUS is 63F8_4000h base + 18h offset = 63F8_4018h
DPLLC-3_MFNMINUS is 63F8_8000h base + 14h offset = 63F8_8014h
DPLLC-3_MFNPLUS is 63F8_8000h base + 18h offset = 63F8_8018h



DPLLCx_MFNn field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved.
26–0 MFNxxx	Multiplication Factor Numerator Bits (nominal, "MINUS", or "PLUS"). The MFNxxx[26:0] bits, in a 2's complement format, give the numerator of the fractional part. See Calculating the Output Frequency

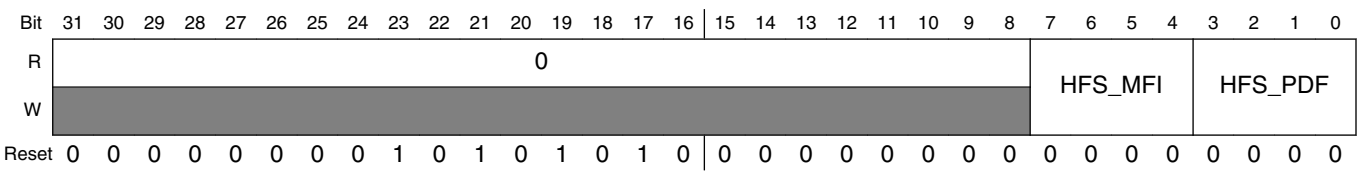
Table continues on the next page...

DPLLCx_MFN*n* field descriptions (continued)

Field	Description
	NOTE: MFN should be in a range from -67108864 to 67108863. If the absolute value of the MFN is larger than MFD, the output clock frequency will differ from the desired frequency. Example settings: 0x4000000 MFN equals -67108864 0x4000001 MFN equals -67108863 0x4000002 MFN equals -67108862 0xFFFFFFF MFN equals -1 0x0000000 MFN equals 0 (default) 0x0000001 MFN equals 1 0x3FFFFFE MFN equals 67108862 0x3FFFFFF MFN equals 67108863

19.31.7 DPLLC High Frequency Support, Operation Register (DPLLCx_HFS_OP)

Addresses: DPLLC-1_HFS_OP is 63F8_0000h base + 1Ch offset = 63F8_001Ch
DPLLC-2_HFS_OP is 63F8_4000h base + 1Ch offset = 63F8_401Ch
DPLLC-3_HFS_OP is 63F8_8000h base + 1Ch offset = 63F8_801Ch



DPLLCx_HFS_OP field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value zero. Reserved.
7–4 HFS_MFI	HFS Mode-Multiplication Factor, Integer part (MFI). If HFS_MFI[3:0] is written with a value of less than 5, MFI will default to 5. See Calculating the Output Frequency Example settings: 0000 Integer part equals 5 0101 Integer part equals 5 0110 Integer part equals 6 1111 Integer part equals 15
3–0 HFS_PDF	HFS mode-Pre-division Factor (HFS_PDF). HFS_PDF determines the value of the PreDividerFactor used in the output frequency formula. See Calculating the Output Frequency Example settings: 0000 PreDividerFactor equals 1 0001 PreDividerFactor equals 2

Table continues on the next page...

DPLLCx_HFS_OP field descriptions (continued)

Field	Description
	0010 PreDividerFactor equals 3 1110 PreDividerFactor equals 15 1111 PreDividerFactor equals 16

19.31.8 DPLLC High Frequency Support Multiplication Factor Denominator Register (DPLLCx_HFS_MFD)

Addresses: DPLLC-1_HFS_MFD is 63F8_0000h base + 20h offset = 63F8_0020h
DPLLC-2_HFS_MFD is 63F8_4000h base + 20h offset = 63F8_4020h
DPLLC-3_HFS_MFD is 63F8_8000h base + 20h offset = 63F8_8020h

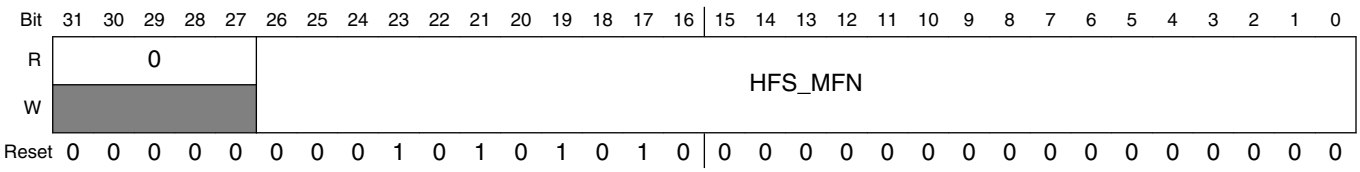
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																HFS_MFD																
W																																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DPLLCx_HFS_MFD field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved.
26–0 HFS_MFD	Multiplication Factor Denominator Bits. The HFS_MFD[26:0] bits, in a 2's complement format, give the denominator of the fractional part. See Calculating the Output Frequency **NOTE:** Bit 26 is always 0. HFS_MFD should be in a range from 0 to 67108863 otherwise the output clock frequency will differ from the desired frequency. Example settings: 0x000000 MFD equals 0 0x000002 MFD equals 1 0x000010 MFD equals 2 0x3FFFFFE MFD equals 67108862 0x3FFFFFF MFD equals 67108863

19.31.9 DPLLC High Frequency Support Multiplication Factor Numerator Register (DPLLCx_HFS_MFN)

Addresses: DPLLC-1_HFS_MFN is 63F8_0000h base + 24h offset = 63F8_0024h
DPLLC-2_HFS_MFN is 63F8_4000h base + 24h offset = 63F8_4024h
DPLLC-3_HFS_MFN is 63F8_8000h base + 24h offset = 63F8_8024h

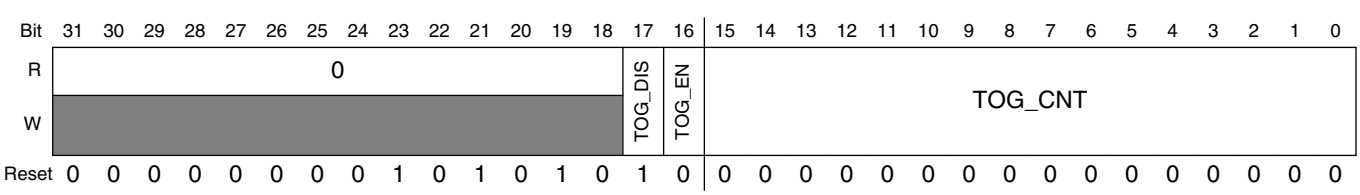


DPLLCx_HFS_MFN field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved
26–0 HFS_MFN	Multiplication Factor Numerator Bits. The HFS_MFN[26:0] bits, in a 2's complement format, give the numerator of the fractional part. See Calculating the Output Frequency **NOTE:** HFS_MFN should be in a range from -67108864 to 67108863. If the absolute value of the HFS_MFN is larger than HFS_MFD, the output clock frequency will differ from the desired frequency. Example settings: 0x4000000 MFN equals -67108864 0x4000001 MFN equals -67108863 0x4000002 MFN equals -67108862 0xFFFFFFF MFN equals -1 0x0000000 MFN equals 0 (default) 0x0000001 MFN equals 1 0x3FFFFFE MFN equals 67108862 0x3FFFFFF MFN equals 67108863

19.31.10 DPLLC Multiplication Factor Numerator Toggle Control Register (DPLLCx_MFN_TOGC)

Addresses: DPLLC-1_MFN_TOGC is 63F8_0000h base + 28h offset = 63F8_0028h
DPLLC-2_MFN_TOGC is 63F8_4000h base + 28h offset = 63F8_4028h
DPLLC-3_MFN_TOGC is 63F8_8000h base + 28h offset = 63F8_8028h



DPLLCx_MFN_TOGC field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>31–18
Reserved</td><td>This read-only field is reserved and always has the value zero.
Reserved.</td></tr>
<tr><td>17
TOG_DIS</td><td>Desense Global Disable. If TOG_DIS is set, then the desense logic for the DPLL is disabled, and software (TOG_EN) or hardware desense requests to enable the logic will be ignored.
(default value of this bit is '1')
<table>
<tr><th>Hardware</th><th>TOG_
EN</th><th>TOG_
DIS</th><th>Desense logic</th></tr>
<tr><td>x</td><td>x</td><td>1</td><td>OFF</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>OFF</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>on</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>on</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>on</td></tr>
</table></td></tr>
<tr><td>16
TOG_EN</td><td>Desense On. Software request to activate desense logic.
(default value of this bit is '0')
NOTE: Desense logic may also be activated by a hardware request ( )
<table>
<tr><th>Hardware</th><th>TOG_
EN</th><th>TOG_
DIS</th><th>Desense logic</th></tr>
<tr><td>x</td><td>x</td><td>1</td><td>OFF</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>OFF</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>on</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>on</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>on</td></tr>
</table></td></tr>
<tr><td>15–0
TOG_CNT</td><td>Toggle counter value. If the value programmed for the count is odd, the next lowest even value is used (original value minus 1).
(default value of tog_counter is '0')
Example settings:
0x0000 Count value equals 0
0x0001 Count value equals 0
0x0002 Count value equals 2
0x0003 Count value equals 2</td></tr>
</table>

19.31.11 DPLLC Desense Status Register (DPLLCx_DESTAT)

Addresses: DPLLC-1_DESTAT is 63F8_0000h base + 2Ch offset = 63F8_002Ch
DPLLC-2_DESTAT is 63F8_4000h base + 2Ch offset = 63F8_402Ch
DPLLC-3_DESTAT is 63F8_8000h base + 2Ch offset = 63F8_802Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TOG_SEL	0				TOG_MFN[11:16]										
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TOG_MFN[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DPLLCx_DESTAT field descriptions

Field	Description
31 TOG_SEL	Toggle set status 0 Desense circuit is inactive. 1 Desense circuit is active.
30–27 Reserved	This read-only field is reserved and always has the value zero. Reserved
26–0 TOG_MFN	Indicates the current MFN value that is sent to the DPLL in desense mode.

Chapter 20
DRAM Memory Controller (DRAM MC)

20.1 Overview

This chapter describes the DRAM MC implemented on the i.MX50. See the Programmable Registers section for all registers.

The i.MX50 supports off-chip DRAM storage using the DRAM MC, which is connected to the internal AXI bus. The DRAM MC supports multiple external memory types, including:

- Standard 1.8 V DDR2
- 1.8 V Mobile DDR1 (LP-DDR1)
- 1.2 V Mobile DDR2 (LP-DDR2)

The DRAM MC consists of 3 major components:

- AXI bus interface
- DRAM Controller
- DRAM PHY

The DRAM MC uses 3 primary clocks: the AHB bus clock HCLK, the AXI bus clock SYS_CLK, and, the ddr interface clock DDR_CLK.

- Supports up to 266 MHz clock frequency at the ddr interface. (532 MHz data rate)
- Supports up to 266 MHz clock frequency at the AXI interface.
- Supports up to 133 MHz clock frequency at the AHB interface.
- The SYS_CLK & DDR_CLK could be either Synchronous or Asynchronous, configurable.
- The HCLK & SYS_CLK are always treated asynchronously by the DRAM MC.

20.1.1 Block Diagram

A high level block diagram of the DRAM MC is shown in Figure 20-1.

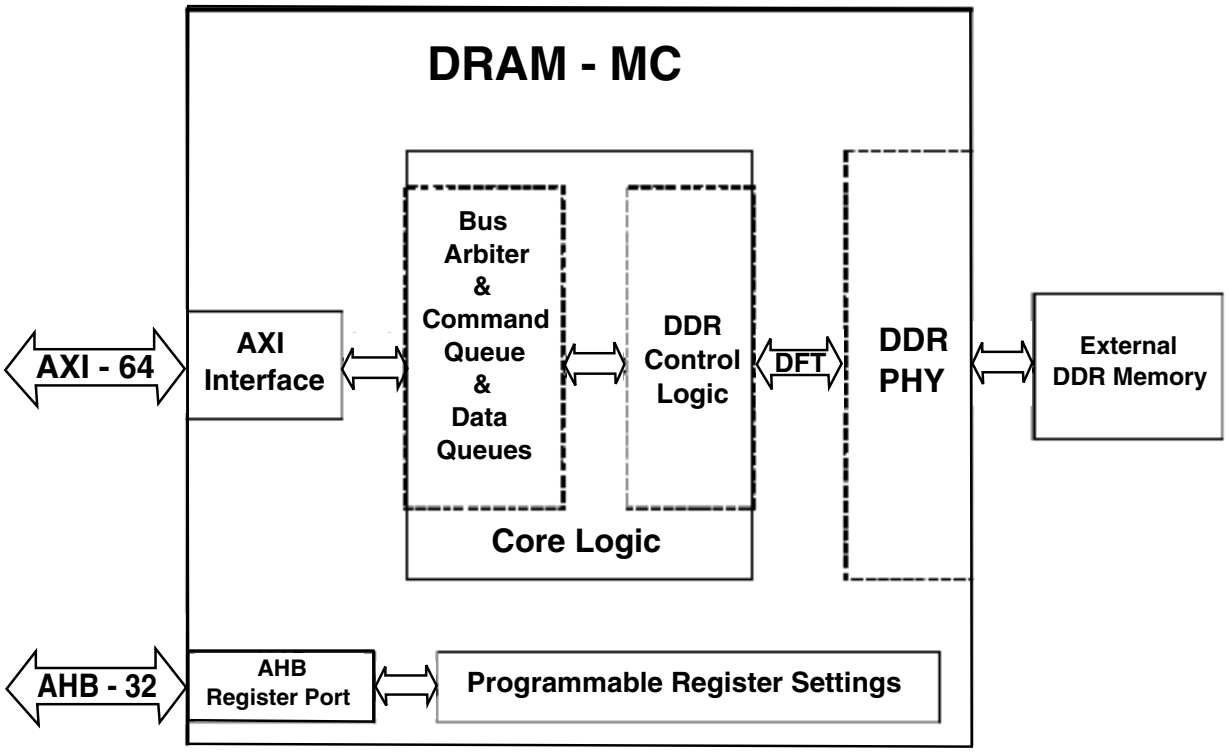


Figure 20-1. DRAM MC Top-Level Block Diagram

DRAM MC implements 2 bus interface. The 64-bit AXI bus interface to read/write data from/to external DDR memory. The AHB interface is dedicated to read/write PIO registers.

20.1.2 DRAM MC Special Signal Considerations

This section list some special signal considerations for the i.MX50 DRAM MC:

- **DRAM_OPEN,DRAM_OPENFB**-These are the echo gating output and feedback pins used by the DRAM PHY to put a window around the DQS transition. They should be routed so that the sum of the trace lengths of DRAM_OPEN + DRAM_OPENFB is equal to the sum of the trace lengths of DRAM_SDCLK0 + DRAM_SDQS0. This connection is required for mDDR, LPDDR2, and DDR2. For the i.MX50 PoP package, these signals are connected on the substrate.
- **DRAM_SDODT0, DRAM_SDODT1**-These are the On-die termination outputs from the i.MX50. For DDR2, these pins should be connected to the DDR2 DRAM ODT pins. For LPDDR2 and mDDR, these pins should be left floating.
- **DRAM_CALIBRATION**-This pin is the ZQ calibration used to calibrate DRAM Ron and ODT. For LPDDR2, this pin should be connected to ground through a 240 ohm 1% resistor. For LPDDR1, this pin should be connected to ground through a 300

ohm 1% resistor. For DDR2, this pin should be connected to ground through a 180 ohm 1% resistor.

- **VREF**-This pin is the VREF input. For LPDDR2 and DDR2, this pin should be connect to ½ of NVCC_EMI_DRAM. For mDDR, this pin should be left floating.

20.2 DRAM MC Address Mapping

The DRAM MC automatically maps AXI bus addresses to the DRAM memory in a contiguous block. Addressing starts at user address 0 and ends at the highest available address according to the size and number of DRAM devices present. This mapping is dependent on how the memory controller was configured and how the parameters in the internal DRAM MC registers are programmed. The exact number and values of these parameters depends on the configuration and the type of memory for which the memory controller was designed.

The mapping of the address space to the internal data storage structure of the DRAM devices is based on the actual size of the DRAM devices available. The size is stored in user-programmable parameters that must be initialized at power up. Certain DRAM devices allow for different mapping options to be chosen, while other DRAM devices depend on the burst length chosen.

20.2.1 DDR SDRAM Address Mapping Options

The address structure of DDR SDRAM devices contains five fields. Each of these fields can be individually addressed when accessing the DRAM. The address map for this DRAM MC is ordered as follows:

Chip Select => Row => Bank => Column => Datapath

The maximum widths of the fields are based on the configuration settings. The actual widths of the fields may be smaller if the device address width parameters (addr_pins, eight_bank_mode and column_size) are programmed differently.

- Chip Select-DRAM MC supports 2 chip-select; "Chip Select" field occupy 1-bit.
- Datapath-DRAM MC supports 16 or 32-bit data width for external DDR memory; "Datapath" field occupy 1 or 2-bit.
- Bank-The DDR memory device defines the bank number. The DRAM MC supports 4 or 8 banks DDR memory, occupy 2 or 3-bits.
- Row, Column-The DDR memory device defines the width of these fields;

20.2.2 Maximum Address Space

The maximum user address range is determined by the width of the memory datapath, the number of chip select pins, and the address space of the DRAM device. The maximum amount of memory can be calculated by the following formula:

chip-selects x bank x $2^{(row+column)}$ x datapath (unit: byte)

For this DRAM controller, the maximum values for these fields are as follows:

- Chip Selects = 2
- Bank-4 or 8, max 8
- Row-max 15-bits
- Column-max 10-bits
- Datapath width = 2 bytes (16-bits) or 4 bytes (32-bits)

As a result, the maximum accessible memory area could reach 2 Gbyte for DRAM MC module.

20.2.3 Memory Mapping to Address Space

An example of address space mapping from 32-bit bus address to DRAM address fields is shown in Figure 20-2 to Figure 20-5.

Example 1: This example corresponds to a memory device with 8 banks, 13 row bits, and 10 column bits; 16-bit datapath mode;

don't care [31:28]	chip-select [27]	row [26:14]	bank [13:11]	column [10:1]	datapath [0]

Figure 20-2. memory address mapping example 1

Example 2: 8 banks, 14 row bits and 10 column bits; 16-bit datapath mode;

don't care [31:29]	chip-select [28]	row [27:14]	bank [13:11]	column [10:1]	datapath [0]

Figure 20-3. memory address mapping example 2

Example 3: 8 banks, 14 row bits and 9 column bits; 32-bit datapath mode;

don't care [31:28]	chip-select [28]	row [27:14]	bank [13:11]	column [10:2]	datapath [1:0]

Figure 20-4. memory address mapping example 3

Example 4: 8 banks, 15 row bits and 10 column bits; 32-bit datapath mode;

don't care [31]	chip-select [30]	row [29:15]	bank [14:12]	column [11:2]	datapath [1:0]

Figure 20-5. memory address mapping example 4

20.2.4 Out-of-Range Address Checking

It is possible that the master attempts to write to an invalid address. For this reason, all incoming addresses are always checked against the addressable physical memory space. If a transaction is addressed to an out-of-range memory location, then a bus error response would be returned by the MC. This feature help debugging. For example: this allows the use of abort exceptions which can be tracked to specific instructions when a debugger is used.

20.3 DRAM MC Clocking

20.3.1 Changing the Input Clock Frequency

The operating frequency of the DRAM MC is dependent on an ASIC-level input clock. There are situations in which the user may wish to modify the frequency of the clock without resetting the memory controller. To change the clock frequency at which the DRAM MC operates, the memory controller must stop processing requests, the clock must be adjusted, the memory controller's timing parameters must be reprogrammed and then the memory controller can be restarted. The procedure to follow for changing the clock frequency is as follows:

1. Ensure that the DRAM MC is idle, that is when the controller_busy parameter is zero. (CTL_BUSY: Bit [8] of DRAM_CTL79)
2. Put the memory devices into self-refresh mode by asserting the srefresh parameter to b1. It is imperative that the memory devices be placed into self-refresh through this parameter, and not through any other means. If the devices have been placed into self-refresh mode through one of the low power modes (programming of the

lowpower_control parameter), then the user should bring the devices out of that low power mode manually, then program the srefresh parameter with a b1.

3. Wait until the memory devices have been placed into self-refresh mode, indicated by the cke_status parameter (Bit [16] of DRAM_CTL63). This parameter is the value of the control_cke signal inside the memory controller, delayed by the number of clocks specified in the cke_delay parameter.
4. Mask the DLL lock state change bit (bit 8) in the int_status parameter, by setting bit 8 in the int_mask parameter to b1.
5. Stop the MC by writing a b0 to the start parameter.
6. Clear the DLL lock state change bit (bit 8) in the int_status parameter, by writing bit 8 in the int_ack parameter to b1. Also clear any other interrupts in the int_status parameter by writing to the int_ack parameter.
7. If the user wishes to use a controller interrupt to signal when the DLL has re-locked, then unmask the DLL lock state change bit (bit 8) in the int_status parameter by clearing bit 8 in the int_mask parameter to a b0.
8. The clock frequency may now be changed. The user should modify any parameters that may be affected by the frequency change such as caslat_lin, caslat_lin_gate, any of the timing parameters, and so on. The user should review the parameters carefully to ensure that all parameters that require modification are changed.
9. After all parameters have been updated, restart the MC by writing a b1 to the start parameter. This forces the DLL to lock to the new frequency.
10. Once the DLL has locked and the PHY has initialized, the memory controller input signal dfi_init_complete will be asserted. If the user is using a controller interrupt to monitor DLL re-lock, then wait for a controller interrupt or poll the int_status parameter for the DLL lock state change bit (bit 8) to be set.
11. At this point, the user may bring the memory devices out of self-refresh by clearing the srefresh parameter to b0. The user should poll the cke_status bit (bit 16 in the DRAM_CTL63 register) to ensure that the self-refresh has been exited before issuing any memory commands. This bit will be set high to indicate that self-refresh has been exited.
12. If any of the memory mode registers require updating at this point, the values must be set in the mode register parameters, and then written to the memory devices by setting the write_modereg parameter to b1.

20.4 DRAM MC AHB and AXI Interface

20.4.1 AHB Register Port

The register port is an independent AHB port to the DRAM MC. This port operates Asynchronously.

The register port only supports the AHB SINGLE access type. The register port only supports transfer types of NONSEQ or IDLE.

There is no support for INCR or WRAP burst types.

There is no support for SEQ or BUSY transfer types.

20.4.2 AXI Interface

The AXI interface function as AXI slave. Transfers are burst-based of variable byte counts. The transfer types INCR and WRAP are fully supported. FIXED burst types are not supported.

Each interface contains five separate channels of traffic to/from the memory: write response, read command, write command, read data, and write data.

20.4.2.1 Internal Command Handling

The DRAM MC uses placement logic to fill the command queue with a command order that maximizes the throughput and efficiency of the core logic.

If the placement logic is being used, the DRAM MC will optimize the core by reordering read and write commands as necessary based on these rules:

- Two read commands from the same thread ID on a port will not be re-ordered and will always execute in the same order as they were accepted into their port.
- Two read commands from different thread IDs on a port may be automatically re-ordered in the core logic to execute out-of-order. When commands from different thread IDs are reordered, read data returned to the AXI port interfaces will also be out-of-order and may be interleaved. To avoid reordering within a port, the AXI bus master should use one thread ID for all commands from any port.
- Two write commands from the same thread ID on a port must remain sequential and will not be reordered. These transactions occur in the order that the commands were received at the port.
- Two write commands from different thread IDs on a port must remain sequential and will not be re-ordered. These transactions will always occur in the order that the commands were received at the port.

- A read command that follows a write command from the same thread ID may be reordered to execute in an optimal order, as long as there are no collisions between the commands.
- A read command that follows a write command from a different thread ID on a port may be reordered to execute in an optimal order, as long as there are no collisions between the commands.
- A write command that follows a read command from the same thread ID may be re-ordered to execute in an optimal order, as long as there are no collisions between the commands.
- A write command that follows a read command from a different thread ID on a port may be re-ordered to execute in an optimal order, as long as there are no collisions between the commands.

Command handling is summarized in Table 20-1

Table 20-1. Re-Ordering / Interleaving Behavior

COMMANDS	THREAD ID	Can Commands be Re-Arranged and Data be Interleaved?
Two Read Commands from 1 port	Same	No
Two Read Commands from 1 port	Different	Yes
Two Write Commands from 1 port	Same	No
Two Write Commands from 1 port	Different	No
Read following a Write from 1 port	Same	Yes
Read following a Write from 1 port	Different	Yes
Write following a Read from 1 port	Same	Yes
Write following a Read from 1 port	Different	Yes

An incoming AXI transaction is mapped into a core logic level transaction, then synchronized from the AXI clock domain to the core logic clock domain and stored in the AXI port FIFOs. Each instruction consists of an address, size, length and thread ID. Since a port may utilize multiple thread IDs, the source ID that is used in the core logic is a combination of both the port and thread information. This concatenation occurs in the Arbiter and this source ID is used in the placement logic. From the AXI FIFOs, the transaction is presented to the Arbiter which arbitrates requests from all ports and forwards a single transaction to the core logic.

20.4.2.2 Configured Options

Each AXI port has been defined for the requirements of the intended system. The configured options are:

Type of interface to the DRAM MC core clock

All ports are clock domain programmable relative to the core clock. These ports initialize in Synchronous operation, but can be changed by programming the associated axi_fifo_type_reg parameter. For more information on the programming of this parameter, refer to Port Clocking. Asynchronous FIFOs handle the clock domain crossing when operating in the Asynchronous modes.

NOTE

When switching between asynchronous and synchronous mode of operation, ensure that there are no outstanding transactions on the port whose behavior is being changed. If transactions are waiting, there may be unexpected loss of data or a lockout condition on that AXI port.

Datapath Width

Each port has a data interface width of 64 bits.

Width of the ID

Each port is configured with a thread ID of 16 bits.

Priority Definition

Command priority is defined based on the port and the command type. For each port Y, there is an axiY_r_priority parameter which defines priorities for all read commands and an axiY_w_priority parameter which defines priorities for all write commands. Supported priority values range from 0 to 7, with 0 as the highest priority.

Register Port

The 32-bit AHB register port is fixed in Asynchronous mode between the AHB bus clock domain and the core clock domain.

Buffering

Each AXI interface contains a command, a read and a write FIFO, and a response storage array. In addition, each programmable port contains an asynchronous response

FIFO to synchronize the memory response to the port clock domain when operating asynchronously.

Table 20-2. AXI Port FIFOs

port number	port data width	clock domain type	command fifo depth	write fifo depth	read fifo depth	response fifo depth	response Storage array depth
Port 0	64	Program mable	8	8	8	4	8
Port 1	none	none	none	none	none	none	none

Narrow Transfer FIFO

For each command from a requestor that may send a narrow transfer (based on axiY_en_size_lt_width_instr parameter programming), an entry is created in the port narrow transfer FIFO with the address, bytes-per-beat and a narrow transfer flag. This will include transfers which may not be narrow in size, but originating from a requestor enabled to support narrow transfers. Each port has been configured with a set number of entries in this FIFO. Therefore, if multiple requestors enabled for narrow transfers issue enough commands to fill the narrow transfer FIFO, the associated port command FIFO may be stalled from accepting more commands. Therefore, the user should only program the axiY_en_size_lt_width_instr parameters carefully with only the requestors expected to use these transfers enabled. The Depth of the Narrow Transfer FIFO is 8.

Exclusive Access Buffer Depth

Exclusive access is an optional AXI feature. This type of access will only be used if exclusive access commands are issued to the memory controller by driving the axiY_ARLOCK signal to 'b10 with a read command. Each native AXI port contains 4 exclusive buffer and therefore each port may monitor the exclusivity of up to 4 transaction at any time. Refer to Exclusive Access, for more information.

Error Detection

When an illegal operational condition is detected on a new AXI transaction entering the port, the port responds through an AXI error signal and the controller interrupt signal, and the error signature is recorded in the register space. The AXI error signal flagged is dependent on the type of transaction that caused the error (read or write). The controller interrupt and the signature information is dependent on type of error (command or data). Refer to Error Responses, for more information on error detection.

20.4.3 Port Clocking

There are 2 user-selectable modes of operation for the AXI port interface: Synchronous & Asynchronous. The mode is selected by programming the axi0_fifo_type_reg parameter. A clock mux was implemented to select the clock mode.

Figure 20-6 shows the DDR clock mux.

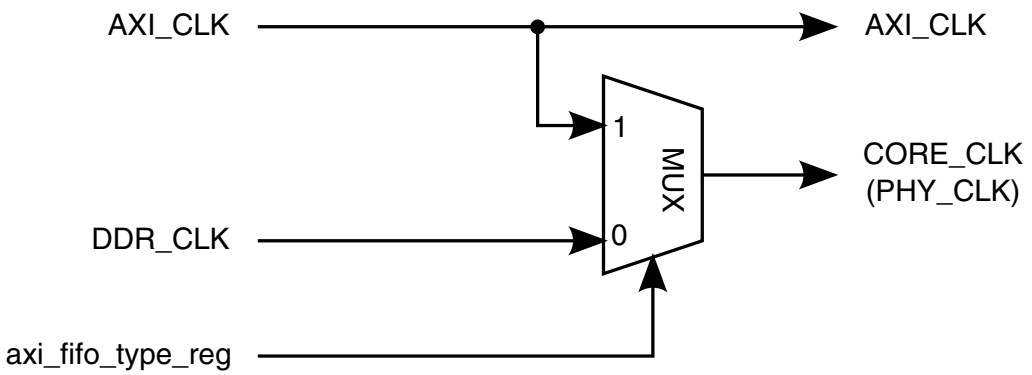


Figure 20-6. ddr clock mux

Synchronous (b11)

CORE_CLK = SYS_CLK. The DDR_CLK was unused. The AXI port interface block will not be required to perform any clock synchronization in any of the FIFOs.

Reserved (b10)

Reserved (b01)

Asynchronous (b00)

CORE_CLK = DDR_CLK. The SYS_CLK & DDR_CLK are treated as Asynchronous to each other. The AXI port FIFOs use two stages of synchronization logic to synchronize commands, write data and read data from SYS_CLK to the CORE_CLK domain.

20.4.4 AXI Interface FIFOs

Each AXI port contains four FIFOs for commands, read data, write data and response synchronization. The response synchronization FIFOs are only used when operating in asynchronous mode. In addition to the FIFOs, each port contains a storage array to hold the read and write responses. The depths and clock domain relativity of the FIFOs and the array are shown in Table 20-2.

The five channels of traffic and their relationship to the port FIFOs are shown in Figure 20-7.

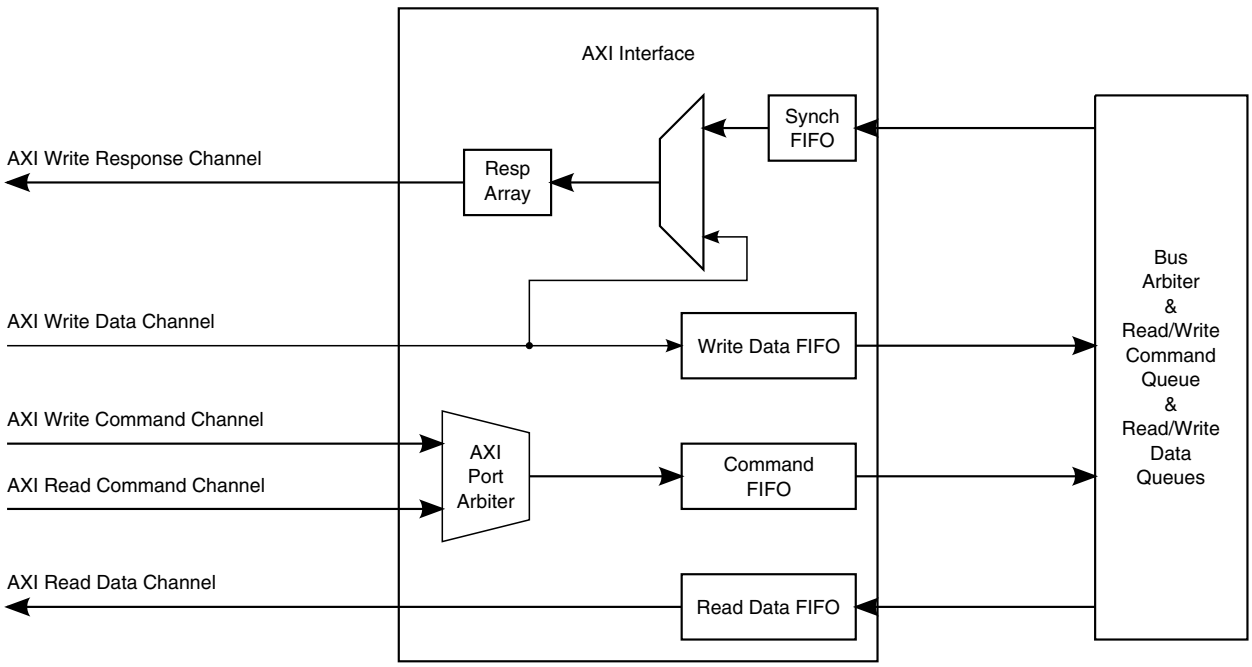


Figure 20-7. AXI Interface FIFOs

20.4.4.1 In-Port Arbitration

Within each port, it is possible that both the read and write command channel of the AXI bus are active concurrently. If this occurs, each port performs a simple arbitration to select the command to pass through. This arbitration is based on the following factors, in order of importance:

1. Order of command acceptance into the port

   The "VALID" signal is set to 'b1 when the master has sent a valid command to the port. The "READY" signal indicates that the port has registered the command. Therefore, a command is considered accepted when both the VALID and READY signals are both asserted. (axiY_AWREADY and axiY_AWVALID both asserted or axiY_ARREADY and axiY_ARVALID both asserted).

2. Priority of read command versus priority of the write command (values of the signals axiY_ARPRIORITY versus axiY_AWPRIORITY).
3. Default Read over Write preference

   If both read and write commands are accepted simultaneously, and their priorities match, then the read channel will be selected first.

Table 20-3 shows the system behavior when the port FIFO is full, allowing commands to accumulate on both channels. The commands will be arbitrated into the port command FIFO based on the order that they are accepted at the port.

Table 20-3. In-Port Arbitration Example 1

CYCLE	PORT COMMAND FIFO	READ CHANNEL		WRITE CHANNEL		ARDITRATED RESULT
		COMMAND ACCEPTED	PRIORITY	COMMAND ACCEPTED	PRIORITY	
1	FULL	1st Read	Any	-	-	None (FIFO full)
2	FULL	-	-	1st Write	Any	None (FIFO full)
3	Available	-	-	-	-	1st Read (Arrived First)
4	Available	2nd Read	Any	-	-	1st Write (Arrived First)
5	Available	-	-	-	-	2nd Read

Table 20-4 shows the system behavior when read and write commands are accepted simultaneously. The commands will be arbitrated into the port command FIFO based on their port priority. When the commands are not simultaneous, they will be arbitrated based on the order that they are accepted at the port.

Table 20-4. In-Port Arbitration Example 2

CYCLE	PORT COMMAND FIFO	READ CHANNEL		WRITE CHANNEL		ARDITRATED RESULT
		COMMAND ACCEPTED	PRIORITY	COMMAND ACCEPTED	PRIORITY	
1	Available	1st Read	Lower	1st Write	Higher	1st Write (Higher Priority)
2	Available	-	-	2nd Write	Higher	1st Read (Arrived First)
3	Available	-	-	-	-	2nd Write
4	Available	2nd Read	Lower	3rd Write	Higher	3rd Write (Higher Priority)
5	Available	-	-	-	-	2nd Read

Table 20-5 shows the system behavior when read and write commands are accepted simultaneously and their priorities are the same. Read commands will automatically be arbitrated ahead of write commands when both other conditions match. When the commands are not simultaneous, they will be arbitrated based on the order that they are accepted at the port.

Table 20-5. In-Port Arbitration Example 3

CYCLE	PORT COMMAND FIFO	READ CHANNEL		WRITE CHANNEL		ARDITRATED RESULT
		COMMAND ACCEPTED	PRIORITY	COMMAND ACCEPTED	PRIORITY	
1	FULL	1st Read	Same	1st Write	Same	1st Read (Read over Write)

Table continues on the next page...

Table 20-5. In-Port Arbitration Example 3 (continued)

CYCLE	PORT COMMAND FIFO	READ CHANNEL		WRITE CHANNEL		ARDITRATED RESULT
		COMMAND ACCEPTED	PRIORITY	COMMAND ACCEPTED	PRIORITY	
2	FULL	2nd Read	Same	-	-	1st Write (Arrived First)
3	Available	-	-	2nd Write	Same	2nd Read (Arrived First)
4	Available	-	-	-	-	2nd Write
5	Available	3rd Read	Same	3rd Write	Same	3rd Read (Read over Write)

20.4.5 AXI Write Response Channel

When a write request is accepted into the AXI interface, an entry will be created in the Response Storage Array for that command. The array will maintain information on the order of received commands from each thread ID for each port. Write responses will be returned in the same order as the commands were received, regardless of the thread ID or the type of response requested (bufferable, coherent bufferable or non-bufferable).

When a write response is ready, the array will verify that this is oldest command for that port and if so, the response will be sent out. The timing of this response is dependent on the type of response requested and the contents of the command queue. Responses for bufferable commands will be available when the port has received all of the data, responses for coherent bufferable commands will be available when the core has received the command and responses for non-bufferable commands will be available when all data has been sent to the PHY. It is possible that the response for a newer command for that port will be ready and waiting until the responses for all older commands for that port are ready and sent to the user interface.

Write responses will be returned to the AXI master through the signal axiY_BRESP and its associated valid indicator, the axiY_BVALID signal.

20.4.5.1 Bufferable, Coherent Bufferable and Non-Bufferable Response Types

Each data port is configured with two signals that work together to determine when an instruction is considered complete and the write completion response (axiY_BRESP) will be returned to the master. These signals are axiY_AWCACHE [0] and axi_AWCOBUF. Table 20-6 details the relationship between these signals. "AXI_AWCOBUF" is a programmable register bit.

Table 20-6. Write Response Types

axi_awcache[0]	axi_awcobuf	Response information[1]
0	0 or 1	Non-bufferable write Command. Response will be ready when the write data has been issued to the PHY.
1	0	Standard bufferable write command. Response will be ready when the command and all associated data have been received by the AXI data port. There is no guarantee of data coherency across all AXI ports.
1	1	Coherent bufferable write command. Response will be ready when the command has been accepted by the command queue in the core logic. This guarantees data coherency across all ports but reduces the overall write response latency relative to the non-bufferable option.

1. The response will only be sent if all of the older write responses have been issued for any thread ID on that port.

20.4.6 AXI Transactions

The AXI Interface supports burst tpye of INCR and WRAP.

The AXI Interface supports burst length of 1-16 bits.

20.4.7 Exclusive Access

The exclusive access feature allows a master to monitor if a memory area has been altered since its last read. Exclusive access does not imply that the memory area is locked; other thread IDs of that port, or other ports, may access the area for reads or writes even though an exclusive access exists. If any writes occur to a memory area with a valid exclusive access request, the master will lose exclusivity and be informed of this status when it attempts to write to the area again. A loss of exclusivity does not trigger an interrupt or any error conditions; however, the AXI protocol requires that the write data is not written to memory if an exclusive write fails its exclusivity check. The master that has lost exclusivity must determine whether to restart the sequence by requesting another exclusive read or to write the data to the memory regardless via a non-exclusive write.

20.4.7.1 Initiating an Exclusive Access Request

Exclusivity is enabled by issuing a read command to memory with the axi_ARLOCK signal driven to b01.

20.4.7.2 Verifying the Request

The AXI specification dictates restrictions for exclusive access requests. After being accepted in the port, an exclusive read request will be checked against these requirements. If any of these conditions are violated, the command will be passed to the core logic as a non-exclusive read. These restrictions are as follows:

- The address must be aligned to the total byte count. (Byte Count = (axi_ARLEN + 1) x 2^{axi_ARSIZE})
- The byte count must be a power-of-two, and less than 128 bytes.

In addition, the AXI specification states that exclusive accesses can not be cacheable. However, since the DRAM MC does not support cacheable commands, this bit is ignored and any cacheable exclusive access requests will be processed as non-cacheable exclusive accesses.

20.4.7.3 Processing Exclusive Accesses

Within the MC core, an exclusive read will behave similarly to any other read data command, accessing memory and returning data to the requestor when the command is processed. The command type is distinguished by its access to the exclusive access buffer, and by the response sent to the master.

An exclusive read will trigger the MC core to store the source ID (axiY_ARID and port number), beat length (axiY_ARLEN), start address (axiY_ARADDR) and beat size (axiY_ARSIZE) for the transaction into an entry of the exclusive access buffer of that port. In addition, each buffer entry contains a single bit to track validity. This bit is set when the exclusive access is requested.

NOTE

The core logic will only maintain one exclusive access request for each source ID. If a new exclusive read request is from a source ID with an existing exclusive access buffer entry, the previous entry will be overwritten with the new transaction.

NOTE

Exclusive access buffer entries may overlap the memory space across a single port or across multiple ports.

If the exclusive access buffer is full when a new exclusive access request is received, one entry of the buffer will be overwritten by the new request. The core logic will attempt to overwrite invalidated entries first, but if all regions are still valid, it will overwrite one of the older valid regions. When the master attempts an exclusive write to the overwritten region, it will fail the exclusivity check.

All read transactions return a response on the read response channel (axiY_RRESP). Successful normal access respond with an OKAY ('b00) whereas successful exclusive accesses respond with EXOKAY (b01). The type of response issued will inform the master how the command was processed.

20.4.7.4 Validity of an Exclusive Request

The exclusive access will be considered valid from the point that the entry is created in the buffer until an activity occurs to invalidate it. These activities cause the "valid" bit of the exclusive access buffer entry to be cleared:

- The same source ID from the same port issues an exclusive write command for the same starting addressing of the same length and beat size, completing the exclusive access request.
- Any source ID from the same port issues a standard (non exclusive) write which partially or completely falls within the range of the exclusive read request.
- The same source ID from the same port issues another exclusive read request for another memory region. This will not clear the valid bit, but will actually overwrite the entire exclusive access buffer entry (and then set the valid bit). The core logic will only track one exclusive region for each source ID.
- Any other source ID from the same port, or any source ID from another port, issues a standard write or a passing exclusive write to any memory address within the range of the exclusive read request. Since the monitored region has been altered, exclusivity is lost.
- The core logic is tracking enough exclusive access commands that the exclusive access buffer for the port is full. Newer commands will be stored, overriding a previous exclusive access entry.

20.4.7.5 Read Commands to Exclusive Access Regions

A memory area is not locked while an exclusive access is valid and therefore other source IDs may issue read commands to a memory area identified as exclusive. Reads, even other exclusive reads, do not affect the exclusivity of a region. Normal read commands will be processed as usual and, if successful, respond to the master with an OKAY ('b00) on the axi_RRESP signal. Exclusive read commands will result in additional entries being created in the exclusive access buffers and, if successful, an EXOKAY ('b01) response will be sent on the axi_RRESP signal.

Note: An exclusive read request from any source ID, from the same port or another port, even to an overlapping address within an exclusive access region, will not affect the initial exclusive access. Another entry will be created in that port's buffer for this request.

20.4.7.6 Non-Exclusive Write Commands to Exclusive Access Regions

Any master in the system may issue a write command to a memory area identified as exclusive. However, a standard write to any address within an active exclusive access region will invalidate the exclusivity of that region. In this case, the "valid" bit of any exclusive access buffer entry that spans the modified locations in memory will be cleared and future exclusive write requests to this region will fail their exclusivity check.

20.4.7.7 Exclusive Writes

An exclusive write command is only distinguished from a standard write command by the axi_AWLOCK signal being driven to 'b01. When an exclusive write is received, the DRAM MC will first compare the source ID (axi_AWID and port number), transaction beat length (axi_AWLEN), transaction start address (axi_AWADDR) and transaction beat size (axi_AWSIZE) to the entries in the exclusive access buffer of that port.

20.4.7.8 Passing Exclusive Access Check

If any of the valid buffer entries exactly match the incoming command, then the following activities will occur:

- That buffer entryís "valid" bit will be cleared.
- All other buffer entries, from this port or other ports, will be checked to see if this write invalidates any other exclusive access regions. If so, the "valid" bits of these entries will be cleared.
- The transaction will be processed.
- If successful, the write response channel (axi_BRESP) will return an EXOKAY response ('b01) to the master.

20.4.7.9 Failing Exclusive Access Check

If an invalid buffer entry exactly matches the incoming command, or no buffer entries exactly match the incoming command, then the write will fail. The command will be processed internally as a flushed write command.

For a flushed write, the write data will be cleared out of the memory controller FIFOs but the data stored in external DRAM memory will NOT change.

The user will be informed of this condition through the write response channel (axi_BRESP). However, instead of an EXOKAY response, the memory controller will issue an OKAY response. This indicates to the master that the exclusive write did not occur. The master may re-issue the request as a non-exclusive write, or may restart the process by re-issuing the exclusive read.

20.4.8 Error Responses

20.4.8.1 AXI Error Response

When an illegal operational condition is detected on a new AXI transaction entering the port, the port responds with an error condition. Instructions that generate AXI errors will result in unpredictable behavior and may cause memory corruption and/or hang conditions.

20.4.8.2 AXI Error Reporting

If an AXI command error occurs, a bit will be set in the int_status parameter and the address and source ID of the command are saved in the port_cmd_error_addr and port_cmd_error_id parameters, respectively. In addition, the access type that relate to the error are stored in the port_cmd_error_type parameter.

Similarly, when a data error occurs, the source ID of the command is saved in the port_data_error_id parameter. The access type that relate to the error are stored in the port_data_error_type parameter.

The bits in the error type parameters are not exclusive. Multiple bits may be set to indicate the type of errors that occurred. Reading the int_ack parameter will allow future errors to be captured in these error parameters.

If multiple errors occur prior to an acknowledgment of the first error, the parameters will still represent the first error attributes. Other error signatures will be lost. If multiple errors occur simultaneously on different ports, the error information will represent the lowest numbered erring port.

20.4.9 Arbiter

From the port interface blocks, commands are presented to the Arbiter, which is responsible for arbitrating between the port requests and sending a single command to the core logic. Refer to DRAM MC Core-Logic for details.

20.4.10 Write Data Queue

The write data queue is a write data storage array for transactions. The queue consists of multiple buffers holding write data for the write requests of a particular port. Write data is stored in these buffers for commands in the command queue until the command is processed in the placement logic and needed by the DRAM command arbitration logic. The buffers can accept data whenever any space is available. The buffers are defined to hold 8 entries.

20.4.11 DRAM Command Processing

The DRAM command processing logic is used to process the commands in the Command Queue. The logic organizes the commands to the memory devices in such a way that data throughput is maximized. DRAM-Bank opening and closing cycles are used for data transfers.

The logic uses a variety of factors to determine when to issue bank open and close commands. The logic reviews the entire Command Queue for look-ahead of which banks are to be accessed in the future. The timing is then set to meet the trc and tras_min timing parameters of the memory devices, values which were programmed into the memory controller on initialization. This flexibility allows the memory controller to be tuned to extract the maximum performance out of memory devices. The parameters that relate to DRAM device protocol are listed in the "DRAM MC Parameter Descriptions" Chapter.

20.4.12 Latency

By using the placement logic of the command queue in the core logic, a new request through any port can be immediately placed at the top of the command queue or can interrupt an ongoing request. This scheme allows a high priority request to be serviced in the shortest possible time.

However, since there are many factors that determine the placement into the command queue, there are also many factors that affect the actual latency of the command. These factors include:

- The coherency status of the transactions already in the command queue: If there is a data coherency conflict with a transaction already in the command queue, the new transaction will be placed after the transaction that produced the conflict. The position of the conflicting transaction determines the latency of the high priority read or write command.
- The priority status of the transactions already in the command queue: If the new command has a higher priority than those already in the command queue, the new request will be serviced ahead of the lower priority command. As a result, the latency of the new command will be lower than the latency of the older command.
- The read, write, and bank information of the transactions already in the command queue: In general, reads will be placed ahead of writes when both are of the same priority level. Read commands are grouped with other read commands of similar priorities and write commands are grouped with other write commands of similar priorities. Among these groupings, transactions with similar bank and different row destinations are separated as much as possible.

If all of the placement conditions are met, then a new command would be placed at the top of the command queue. However, if the new command is of a higher priority than the transaction executing, the current command will be interrupted and the new command will execute first. The interruption will occur at a natural burst boundary of the DRAMs. The interrupted transaction will placed at the top of the placement queue and it will resume after the new request is completed. The page status of the new transaction determines when the current transaction is interrupted. If the page for the new transaction is already open, then the current transaction will be interrupted at the next natural burst boundary of the DRAM device. If the page is not currently open, then the new request will be placed at the top of the command queue while its page is prepared.

20.5 DRAM MC Core-Logic

Figure 20-8 shows the DRAM MC core-logic.

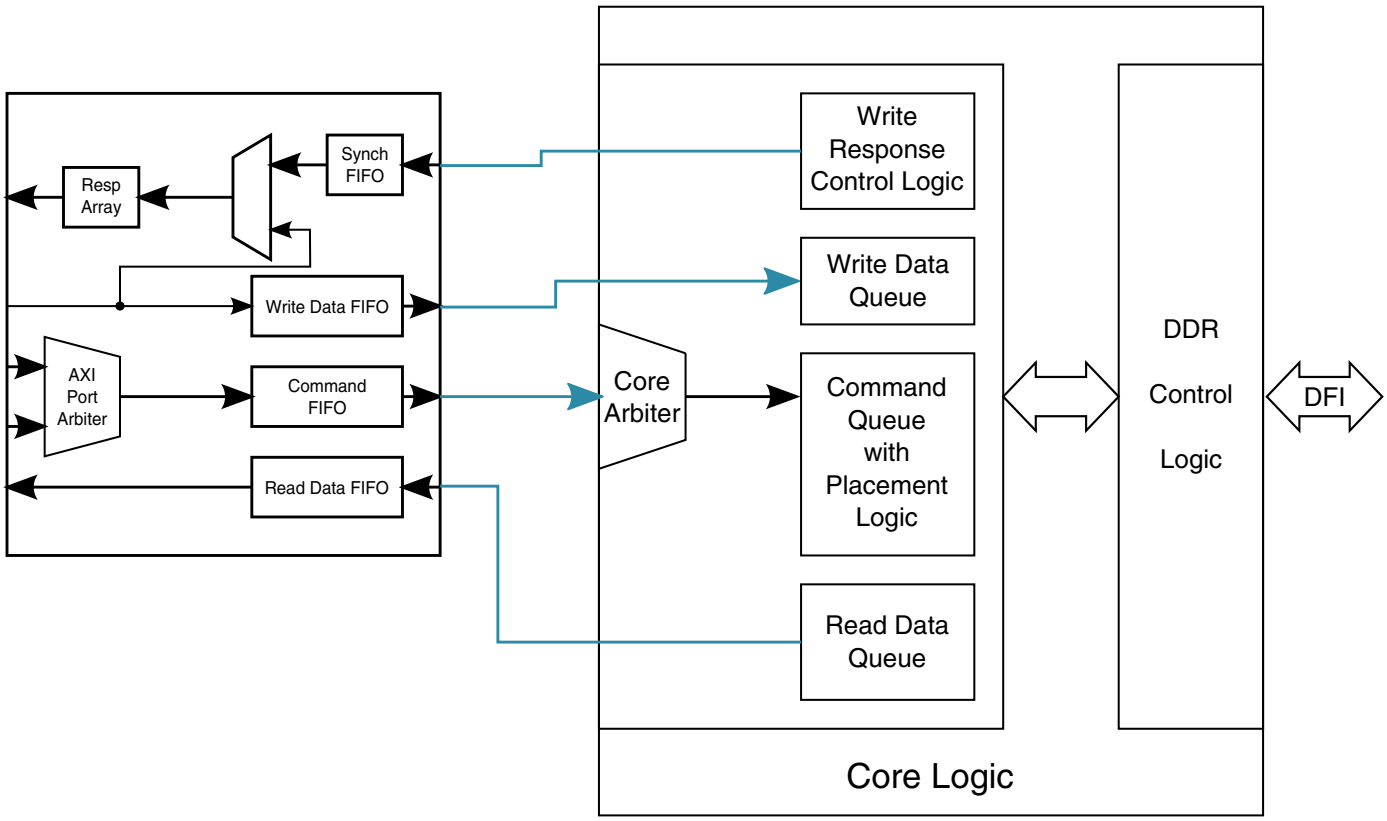


Figure 20-8. DRAM MC Core Logic

The core-logic suports either multi-ports or single-port usecases. A single-port scheme is implemented in this DRAM MC.

The Arbiter is responsible for arbitrating requests from the ports and sending requests to the core logic. Each transaction received at the Arbiter logic has an associated priority, which works with each port's arbitration logic to determine how ports issue requests to the core logic. This memory controller supports the Bandwidth Allocation/Priority Round-Robin arbitration scheme.

The Arbiter logic routes read data from the core logic to the appropriate port. The requesting port is assumed able to receive the data. Write data from each port is connected directly to its own write data interface in the core logic, allowing the ports to independently pass write data to the core buffers.

20.5.1 Command Queue with Placement Logic

From the Arbiter, commands are routed to the command queue of the core logic. The command queue is fed using a placement algorithm. For more information on this algorithm, refer to Core Command Queue with Placement Logic.

20.6 Core Command Queue with Placement Logic

The core logic contains a command queue that accepts commands from the Arbiter. This command queue uses a placement algorithm to determine the order that commands will execute in the core logic. The placement logic follows many rules to determine where new commands should be inserted into the queue, relative to the contents of the command queue at the time. Placement is determined by considering address collisions, source collisions, data collisions, command types and priorities. In addition, the placement logic attempts to maximize efficiency of the core logic through command grouping and bank splitting. Once placed into the command queue, the relative order of commands is constant.

Many of the rules used in placement may be individually enabled/disabled. In addition, the queue may be disabled by clearing the placement_en parameter, resulting in an in-line queue that services requests in the order they are received. If the placement_en parameter is cleared to 'b0, the placement algorithm will be ignored.

20.6.1 Rules of the Placement Algorithm

The factors affecting command placement all work together to identify where a new command fits into the execution order. They are listed in order of importance.

20.6.1.1 Address Collision/Data Coherency Violation

The order in which read and write commands are processed in the memory controller is critical to proper system behavior. While reads and writes to different addresses are independent and may be re-ordered without affecting system performance, reads and writes that access the same address are significantly related. If the port requests a read after a write to the same address, then repositioning the read before the write would return the original data, not the changed data. Similarly, if the read was requested ahead of the write but accidentally positioned after the write, then the read would return the new data, not the original data prior to being overwritten. These are significant data coherency mistakes.

To avoid address collisions, reads or writes that access the same chip select, bank and row as a command already in the command queue will be inserted into the command queue after the original command, even if the new command is of a higher priority.

This factor may be enabled/disabled through the addr_cmp_en parameter and should only be disabled if the system can guarantee coherency of reads and writes.

20.6.1.2 Source ID Collision

Each port is assigned a specific source ID that is a combination of the port and thread ID information, and identifies the source uniquely. This allows the memory controller to map data from/to the correct source/destination.

Note that a source ID does contain port identification information which means that the rules for placement are dependent on the requesting port. There will not be source ID collisions between ports.

In general, read commands from the same source ID will be placed in the command queue in order. Therefore, a read command with the same source ID as a read command already in the command queue will be processed after the original read command. All write commands from a port, even with different source IDs, will be executed in order.

The behavior of commands of different types from the same source ID is dependent on the user configuration. For this Memory Controller, the placement of new read/write commands that collide in terms of source ID with existing entries in the command queue will only depend on other commands of the same type, not on different types. This means that, if there are no address conflicts, a read command could be executed ahead of a write command with the same source ID, and likewise a write command could be executed ahead of a read command with the same source ID.

This feature will always be enabled.

20.6.1.3 Write Buffer Collision

Incoming write requests in the command queue are allocated to one of the 4 write buffers of the core logic automatically based on availability. New write commands will be designated to any available buffer. However, back-to-back write requests from a particular source ID will be allocated to the same write buffer as the previous command.

Since the core logic must pull data out of the buffers in the order it was stored, if a write command is linked to a buffer that is associated with another command in the queue, then the new command will be placed in the command queue after that command, regardless of priority. This feature will always be enabled.

20.6.1.4 Priority

Priorities are used to distinguish important commands from less important commands. Each command is given a priority based on the command type through the programmable parameters axiY_r_priority and axiY_w_priority (where Y represents the port number). A priority value of 0 is the highest priority, and a priority value of 7 is the lowest priority.

The placement algorithm will attempt to place higher priority commands ahead of lower priority commands, as long as they have no source ID, write buffer or address collisions. Higher priority commands will be placed lower in the command queue if they access the same address, are from the same requestor or use the same buffer as lower priority commands already in the command queue.

20.6.1.5 Bank Splitting

Before accesses can be made to two different rows within the same bank, the first active row must be closed (pre-charged) and the new row must be opened (activated). Both activities require some timing overhead; therefore, for optimization, the placement queue will attempt to insert the new command into the command queue such that commands to other banks may execute during this timing overhead. The placement of the new commands will still follow priority, source ID, write buffer and address collision rules.

The placement logic will also attempt to optimize the core logic by inserting a command to the same bank as an existing command in the command queue immediately after the original command. This reduces the overall timing overhead by potentially eliminating one pre-charging/activating cycle. This placement will only be possible if there are no priority, source ID, write buffer or address collisions or conflicts with other commands in the command queue.

All bank splitting features are enabled through the bank_split_en parameter.

20.6.1.6 Read/Write Grouping

The memory suffers a small timing overhead when switching from read to write mode. For efficiency, the placement queue will attempt to place a new read command sequentially with other read commands in the command queue, or a new write command sequentially with other write commands in the command queue. Grouping will only be possible if no priority, source ID, write buffer or address collision rules are violated.

This feature is enabled through the rw_same_en parameter.

20.6.2 Command Execution Order After Placement

Once a command has been placed in the command queue, its order relative to the other commands in the queue at that time is fixed. While this provides simplicity in the algorithm, there are drawbacks. For this reason, the Memory Controller offers two options that affect commands once they have been placed in the command queue.

20.6.2.1 Command Aging

Since commands can be inserted ahead of existing commands in the command queue, the situation could occur where a low priority command remains at the bottom of the queue indefinitely. To avoid such a lockout condition, aging counters have been included in the placement logic that measure the number of cycles that each command has been waiting. If command aging is enabled through the active_aging parameter, then if an aging counter hits its maximum, the priority of the associated command will be decremented by one (lower priority commands are executed first). This increases the likelihood that this command will move to the top of the command queue and be executed. Note that this command does not move relative positions in the command queue when it ages; the new priority will be considered when placing new commands into the command queue.

Aging is controlled through a master aging counter and command aging counters associated with each command in the command queue. The age_count and command_age_count parameters hold the initial values for each of these counters, respectively. When the master counter counts down the age_count value, a signal is sent to the command aging counters to decrement. When the command aging counters have completely decremented, then the priority of the associated command is decremented by one number and the counter is reset. Therefore, a command does not age by a priority level until the total elapsed cycles has reached the product of the age_count and command_age_count values. The maximum number of cycles that any command can wait in the command queue until reaching the top priority level is the product of the age_count value, the command_age_count value, and the number of priority levels in the system.

20.6.2.2 High-Priority Command Swapping

Commands are assigned priority values to ensure that critical commands are executed more quickly in the memory controller than less important commands. Therefore, it is desirable that high-priority commands pass into the core logic as soon as possible. The placement algorithm takes priority into account when determining the order of

commands, but still allows a scenario in which a high-priority command sits waiting at the top of the command queue while another command, perhaps of a lower priority, is in process.

The high-priority command swapping feature allows this new high-priority command to be executed more quickly. If the user has enabled the swapping function through the swap_en parameter, then the entry at the top of the command queue will be compared with the current command in progress. If the command queue's top entry is of a higher priority (not the same priority), and it does not have an address, source ID or write buffer conflict with the current command being executed, then the original command will be interrupted.

For this memory controller, an additional check is performed before a read command is interrupted. If the read command in progress and the read command at the top of the command queue are from the same port, then the executing command will only be interrupted if the swap_port_rw_same_en parameter is set to 'b1. If this parameter is cleared to 'b0, a read command from the same port as a read command in progress, even with a higher priority and without any conflicts, would remain at the top of the command queue while the current command completes.

NOTE

All write commands from a single port, even with different source IDs, will be executed in order. Therefore, two write commands from the same port will never be swapped regardless of the settings of the swap_en and swap_port_rw_same_en parameters.

NOTE

Priorities are assigned to read commands based on the settings in the axiY_r_priority parameters. While all read commands from a port are assigned the same priority when placed in the command queue, their priorities may change over time through command aging. While uncommon, it is possible that a higher-priority read command may be at the top of the command queue while a lower-priority read command is executing. The behavior of the system in this scenario is based on the value of the swap_en and swap_port_rw_same_en parameters.

See Table 20-7 for details.

Table 20-7. Swapping Behavior

active command priority	new command priority	originating port for commands	conflicts ?	action
Higher	Lower	Same or Different	Yes or No	Current Command Continues
Lower	Higher	Same	Yes	Current Command Continues
Lower	Higher	Same	No	Will swap IF swap_en = 1 & swap_port_rw_same_en = 1
Lower	Higher	Different	No	Will swap IF swap_en = 1

If the command is to be interrupted, it will be halted after completing the current burst, stored and placed at the top of the queue, and the new command will be executed. As long as the command queue is not full, new commands may continue to be inserted into the command queue based on the placement rules, even at the head of the queue ahead of the interrupted command. The top entry in the command queue will be executed next. Whenever the interrupted command is resumed, it will start from the point at which it was interrupted.

Note that priority 0 commands will never be interrupted, so the user should set any commands that should not be interrupted to priority 0. If supported by the port interfaces, setting the swap_port_rw_same_en parameter will enable interleaving.

20.7 Notes for Programming of the Timing Parameters

20.7.1 Command to Command Timing

For flexibility and maximum user control, the MC provides a set of timing parameters to control the timing between commands of different types. Each of these parameters should be programmed with the actual number of clocks of turn-around required for this transaction combination on the data bus. The parameters are:

- add_odt_clk_difftype_diffcs
- add_odt_clk_w2r_samecs
- add_odt_clk_r2w_samecs
- add_odt_clk_sametype_diffcs
- r2r_diffcs_dly
- r2r_samecs_dly
- r2w_diffcs_dly
- r2w_samecs_dly
- w2r_diffcs_dly
- w2r_samecs_dly

- w2w_diffcs_dly
- w2w_samecs_dly

20.7.2 Refresh Per Chip-select

While memory refresh operations are necessary for DRAM data integrity, they unfortunately draw a significant current and temporarily disable the memory from accepting other commands. A standard practice is to issue refresh commands to all chip selects simultaneously at each tref interval. This minimizes the downtime before another data command can be processed (trfc) but does increase peak current while the refresh is in progress.

As an alternative, the MC includes the option to individually refresh each chip select in a sequential manner. When using this feature, a memory refresh may be initiated from the user interface by setting the arefresh parameter to b1 or through an internal source such as the expiration of the tref counter or an exit from Self-Refresh.

Regardless of the source, when the refresh command is received, all transactions will be inhibited and a refresh sequence will be initiated. This refresh sequence issues refresh commands to one chip select at a time, moving through all of the chip selects in succession. A programmable delay (the tref_interval parameter) sets the time period between when refresh commands are issued to different chip selects.

A related memory-specific parameter (the trfc parameter) sets the time that a DRAM device must wait between receiving a refresh command to when a data transaction can be processed. When each refresh command is issued, the trfc counter inside the MC begins a countdown from the programmed value (in the trfc parameter). With each refresh command, the counter is reset and the memory controller will only accept commands trfc cycles after when the final refresh command has been issued.

Once the trfc time expires, normal read/write traffic and behavior resumes in the memory controller. All other transactions will be inhibited during this refresh sequence until all chip selects have been issued their refresh commands and the associated timing requirements have been met.

20.7.3 Programming of the tref Parameter

The tref parameter sets the average interval between refreshes. The actual interval may vary by up to 8 cycles from refresh to refresh, depending on other activities that are occurring in the controller at the time. Over an infinite time period, the average interval

will be equal to the number of clocks set by this parameter; however, if the user sets this parameter to be exactly equal to the specification value tREF, then local variations might mean that the memory tREF value may be violated by a very small amount.

Although we never seen a failure in a real system as a result of programming the tref parameter to the exact tREF parameter, but it is recommended that the user should program the tref parameter to the value a litter smaller than (tREF/tCK), which will result in slightly lower overall bandwidth and slightly higher memory power usage.

20.7.4 Programming of the tref_interval Parameter

The interval is a critical value in achieving real current savings. The following considerations should be taken for programming the tref_interval parameter:

- A larger interval will increase the time that the MC is held off from processing data transactions.
- Peak current restrictions will guide the interval value. The interval should be defined to the minimum value that still meets peak current restrictions.
- Programming the interval to zero will disable the refresh per chip select logic and refresh commands will be issued simultaneously to all chip selects.
- The refresh per chip select logic requires at least one dead cycle between refresh commands, and therefore the minimum interval used is actually 2 cycles. Programming the tref_interval parameter to 1 will default to 2 cycles.
- Other factors in the design also affect timing, and therefore the actual delay between refresh commands may exceed the programmed value. This is particularly likely when the interval value programmed is very low, such as b002.
- When the trfc counter completes its countdown, the memory controller will accept any pending commands. Since no read or write commands may occur during the entire refresh sequence, the tref_interval parameter should always be set to a value lower than the trfc parameter.

20.8 DRAM Low Power Operation

In many applications, it is desirable to minimize the power consumption of the DRAM MC and the memory devices. The MC provides various user-configurable low power options to address power savings.

20.8.1 Low Power Modes

There are four low power modes available in the MC. The low power modes are listed from least to most power saving.

Note: It is not possible to exit one low power mode and enter another low power mode simultaneously. The user should plan for a minimum delay between exit and entry between the two low power modes of 15 cycles in which the memory controller must remain stable.

1. **Memory Power-Down**

   The MC sets the memory devices into power-down, which reduces the overall power consumption of the system, but has the least effect of all the low power modes. In this mode, the MC and memory clocks are fully operational, but the CKE input bit to the memory devices is de-asserted. The MC will continue to monitor memory refresh needs and will automatically bring the memory out of power-down to perform these refreshes. When a refresh is required, the CKE input bit to the memory devices will be reenabled. This action brings the memory devices out of power-down. Once the refresh has been completed, the memory devices will be returned to power-down by de-asserting the CKE input bit.

2. **Memory Power-Down with Memory Clock Gating**

   The memory controller sets the memory devices into power-down and gates off the clock to the memory devices. Refreshes will be handled as in the Memory Power-Down mode (Mode 1), with the exception that gating on the memory clock will be removed before asserting the CKE pin. After the refresh has been completed, the memory devices will be returned to power-down with the clock gated. Before the memory devices are removed from power-down, the clock will be gated on again. The user should not use this mode for memory devices that do not support memory clock gating. Clock gating is not supported for standard DDR1, DDR2 or DDR3 devices. When set into this mode, the memory controller will attempt to place the memory devices in power-down and gate off the memory clock. The memory will function unpredictably and may hang.

3. **Memory Self-Refresh**

   The MC sets the memory devices into self-refresh. In this mode, the MC and memory clocks are fully operational and the CKE input bit to the memory devices is de-asserted. Since the memory automatically refreshes its contents, the MC does not need to send explicit refreshes to the memory.

4. **Memory Self-Refresh with Memory Clock Gating**

The MC sets the memory devices into self-refresh and gates off the clock to the memory devices. Before the memory devices are removed from self-refresh, the clock will be gated on again.

20.8.2 Low Power Mode Control

The memory controller may enter and exit the various low power modes in the following ways:

- **Automatic Entry**

  When the MC is idle, three timing counters begin counting the cycles of inactivity. If any of the counters expires, the MC will enter the low power mode associated with that counter.

- **Manual Entry**

  The user may initiate any low power mode by setting the bit of the lowpower_control parameter associated with the desired mode. The MC will enter the selected low power mode when it is has completed its current memory burst.

Automatic and Manual entry methods are both controlled by two parameters: lowpower_control and lowpower_auto_enable. The lowpower_control parameter contains individual enable/disable bits for each low power mode, and the lowpower_auto_enable parameter controls whether each mode will be entered automatically or manually.

Note: The user should not use both automatic and manual modes for the various low power modes. All modes should be entered automatically or all entered manually.

20.8.3 Automatic Entry

Automatic Entry will occur if all of the following conditions are true:

- The mode is programmed for automatic entry by setting the relevant bit in the lowpower_auto_enable parameter to b1.
- The particular mode is enabled in the lowpower_control parameter.
- The memory controller is idle.
- The counter associated with this mode expires.

There are three counters in all to cover the low power modes. There are separate counters for each of the three memory self-refresh low power modes (Modes 3 and 4). Memory Power-Down mode (Mode 1) and Memory Power-Down with Memory Clock Gating mode (Mode 2) share the same counter.

The counters determine the number of idle cycles before entry into the associated low power mode. All of these counters are re-initialized each time there is a new read or write transaction entering or executing in the MC. This ensures that the MC will not enter any of the low power modes when active.

20.8.4 Automatic Exit

All low power modes can be entered through automatic entry, and will be exited automatically when any of the following conditions occur:

- A new read or write transaction appears at the memory controller interface.
- The MC must refresh the memory when in either of the power-down modes (Modes 1 or 2). After completing the memory refresh, the memory controller re-enters power-down.
- The counter for a deeper low power mode expires. The MC must exit the current low power mode in order to enter the deeper low power mode. A minimum of 15 cycles occur between exit from one low power mode before entering into the next low power mode, even if the counters expire within 15 cycles of each other. Note that the MC will not enter a less deep low power mode, regardless of which counters expire.

20.8.5 Manual "On-Demand" Entry

Manual Entry will occur if all of the following conditions are true:

- The mode is programmed for manual entry by clearing the relevant bit in the lowpower_auto_enable parameter to b0.
- The particular mode is set to íb1 in the lowpower_control parameter.

For manual entry, the lowpower_control parameter triggers entry into the low power modes. The MC does not need to be idle when the a low power mode bit is enabled. When a particular mode that is programmed for manual entry is enabled, the memory controller will complete the current memory burst access, and then, regardless of the activity inside the MC or at the memory interface, it will enter the selected low power mode.

If new transactions enter the MC while it is in any of the low power modes, they will accumulate inside the MC's command queue until the queue is full.

20.8.6 Manual "On-Demand" Exit

Exit from a manually-entered low power mode is also manual. Clearing the lowpower_control parameter bits to b0 will trigger the MC to pull the memory devices out of powerdown or self-refresh, and command processing will resume.

If a different lowpower_control parameter bit is set to b1 while in one of the low power modes, or on clearing of the original bit to b0, the MC will exit the current low power mode. There will be at least a 15 cycle delay before the MC is fully operational, or enters the new low power mode.

20.8.7 Register Programming

The low power modes of the MC are controlled through the lowpower_control and lowpower_auto_enable parameters. These parameters each contain one bit for controlling each low power mode. The lowpower_control parameter enables the associated low power mode, and the lowpower_auto_enable parameter sets the entry method into that mode as manual or automatic. Table 20-8 shows the relationship between the lowpower_control and lowpower_auto_enable parameters and the various low power modes.

Table 20-8. Low Power Mode Parameters

Low Power Mode	Enable	Entry
Memory Power-Down (Mode 1)	lowpower_control [4] = 1	lowpower_auto_enable [4] 0 = Manual 1 = Automatic
Memory Power-Down with Memory Clock Gating (Mode 2)	lowpower_control [3] = 1	lowpower_auto_enable [3] 0 = Manual 1 = Automatic
Memory Self-Refresh (Mode 3)	lowpower_control [2] = 1	lowpower_auto_enable [2] 0 = Manual 1 = Automatic
Memory Self-Refresh with Memory Clock Gating (Mode 4)	lowpower_control [1] = 1	lowpower_auto_enable [1] 0 = Manual 1 = Automatic
Reserved	lowpower_control [0] Please keep this bit to zero.	lowpower_auto_enable [0] Please keep this bit to zero.

When a lowpower_control parameter bit is set to íb1 by the user, the MC checks the lowpower_auto_enable parameter.

- If the associated bit in the lowpower_auto_enable parameter is set to b1, then the MC will watch the associated counter for expiration, and then enter that low power mode. Table 20-9 shows the correlation between the low power modes and the counters that control each modeís automatic entry.
- If the associated bit in the lowpower_auto_enable parameter is cleared to b0, then the MC will complete its current memory burst access and then enter the specified low power mode.

Table 20-9. Low Power Mode Counters

Low Power Mode	Entry
Memory Power-Down (Mode 1)	lowpower_power_down_cnt
Memory Power-Down with Memory Clock Gating (Mode 2)	lowpower_power_down_cnt
Memory Self-Refresh (Mode 3)	lowpower_self_refresh_cnt
Memory Self-Refresh with Memory Clock Gating (Mode 4)	lowpower_external_cnt

NOTE

The counter parameters must be programmed to non-zero values for proper operation.

NOTE

The values in the lowpower_auto_enable parameter are only relevant when the associated lowpower_control bit is set to b1.

Multiple bits of the lowpower_control and lowpower_auto_enable parameters can be set to b1 at the same time. When this happens, the MC always enters the deepest low power mode of all the modes that are enabled. If the MC is already in one low power mode when a deeper low power mode is requested automatically or manually, it must first exit the current low power mode, and then enter the deeper low power mode. A minimum 15 cycle delay occurs before the second entry.

The timing for automatic entry into any of the low power modes is based on the number of idle cycles that have elapsed in the MC.

20.9 Interface for LPDDR2 Devices

This MC is designed to interface with LPDDR2 memory devices. While other memory devices may also be supported, the MC will only interface with a single type of memory at a time. The user must set the mode through the dram_class mode parameters. Certain features of LPDDR2 memories require special logic in the MC.

20.9.1 Restrictions

The MC does not support interleaving (MR1 [3]), wrapping (MR1 [4]) and only supports memory burst lengths of 4 or 8. The burst terminate command and data not valid commands are not used.

20.9.2 ZQ Calibration for off-chip LPDDR2 Devices

ZQ calibration is a feature of LPDDR2 memories used to calibrate DRAM Ron and ODT values. There are four types of ZQ commands defined by the JEDEC LPDDR2 specification:

- ZQCS, or ZQ short calibration, is used for a periodic update to the DRAM values.
- ZQCL, or ZQ long calibration, is used for longer updates to the DRAM values.
- ZQINIT, or ZQ initialization, is used during initialization of memory.
- ZQRESET, or ZQ reset, is used to reset the ZQ calibration values to their default state.

All commands are supported by the MC. Before a ZQ command is initiated, regardless of how the command was requested, pending refreshes and power down commands are all handled, any commands in process are completed, and all active memory banks are closed. While the ZQ command is in process, the MC core command queue will be halted. During this time, no other commands are allowed, CKE must be held asserted and the DQ bus must be held in high impedence state. Mode register MR10 in the LPDDR2 memory devices control ZQ calibration.

NOTE

If the refresh cycle time is long, PHY update times are extensive, and there are many chip selects in the system, it is possible that a large number of cycles may elapse without allowing any user activity.

NOTE

Although ZQ calibration is only required for LPDDR2, Freescale recommends the use of ZQ calibration for mDDR and DDR2 as well.

20.9.3 Programmable Control

- refresh_per_zq

  Defines the number of refresh cycles that will occur between automatic ZQ calibration sequences. The refresh timer will only trigger a short ZQ calibration sequence. Setting this parameter to 0x0 will disable automatic ZQCS commands. When using the rotate option, this parameter should be programmed to the total number of cycles desired between ZQCS commands divided by the number of chip selects in the system.

- zq_in_progress

  Status parameter to indicate if a ZQ calibration of any type is in progress.

- zq_on_sref_exit

  May be enabled to issue a short ZQ, long ZQ, ZQ reset or ZQ initialization calibration automatically when exiting self-refresh mode. A short calibration sequence may calibrate all chip selects in the system (if the parameter zqcs_rotate is cleared to b0) or just one chip select on each self-refresh exit, in rotating fashion. A long, initialization or reset calibration will affect all chip selects in the system. It is recommended to set this parameter to execute a ZQCL on self-refresh exit. By setting this parameter this way, a ZQCL will be issued before any other commands are permitted to execute. If this parameter is set to any other setting, the system must manage self-refresh exit and ZQ calibration. If using the ZQCS option, it is recommanded that the rotate function should be disabled (the parameter zqcs_rotate should be cleared to b0) to calibrate all chip selects.

NOTE

If a ZQ command is issued prior to a self-refresh command, once the self-refresh command is accepted, it will cause the ZQ command to be aborted. Therefore, the user should utilized the zq_on_sref_exit parameter to ensure that a ZQ calibration is executed.

NOTE

If a ZQ command is issued while the memory is in self-refresh, and the zq_on_sref_exit parameter has been

programmed to a non-zero value, then the new ZQ command will be issued only AFTER self-refresh exit and the ZQ command specified in zq_on_sref_exit has completed. Therefore, if the zq_on_sref_exit function is being used, then issuing any ZQ commands during self-refresh will only result in added latency and is not recommended.

- zq_req

  Triggers a short, long, initialization or reset ZQ calibration. A short calibration sequence may calibrate all chip selects in the system (if the parameter zqcs_rotate is cleared to b0) or just one chip select on each self-refresh exit, in rotating fashion. A long, initialization or reset calibration will affect all chip selects in the system.

- zqcl

  Specifies the number of cycles required for the memory devices to complete a long (noninitialization) ZQ calibration.

- zqcs

  Specifies the number of cycles required for the memory devices to complete a short ZQ calibration. This parameter can be programmed as needed for the user system, either to the number of cycles required to complete a ZQCS to all chip selects in the system (if the zqcs_rotate parameter will be cleared to b0) or the number of cycles to complete a ZQCS to a single chip select (if the zqcs_rotate parameter will be set to b1).

- zqcs_rotate

  Defines if all chip selects or just one (on a fixed rotational basis) will be calibrated on any tuype of ZQCS request. Setting this parameter to b1 enables the single chip select calibration and minimizes the overhead of periodic ZQ calibrations.

- zqinit

  Specifies the number of cycles required for the memory devices to complete a ZQ initialization command.

- zqreset

  Specifies the number of cycles required for the memory devices to complete a ZQ reset operation.

20.9.4 ZQ Short Calibration

A ZQ short calibration may calibrate each chip select enabled in the cs_map parameter on each request, or just one chip select on each ZQ short calibration request. The value of the zqcs_rotate parameter determines whether all chip selects or just one chip select are calibrated on each request. The MC will issue the appropriate ZQCS commands.

A ZQCS will be issued in the following cases:

- The user requests a short ZQ calibration by setting the zq_req parameter to b01.
- If the zq_on_sref_exit parameter is set to b01, then a ZQ short calibration will be issued whenever the memory devices exit self-refresh mode.
- The number of refreshes that have occurred since the last ZQCS exceeds the value specified in the refresh_per_zq parameter.

20.9.5 ZQ Long Calibration

A ZQ long calibration will calibrate each chip select in the system, one chip select at a time. The MC will issue the appropriate ZQCL commands to each chip select.

A ZQCL will be issued in the following cases:

- The user requests a long ZQ calibration by setting the zq_req [1] parameter bit to b1.
- If the zq_on_sref_exit [1] parameter bit is set to b1, then a ZQ long calibration will be issued whenever the memory devices exit self-refresh mode.

20.9.6 ZQ Initialization

A ZQ initialization calibration will calibrate each chip select in the system, one chip select at a time. The MC will issue the appropriate ZQINIT commands to each chip select.

A ZQINIT will be issued in the following cases:

- The user requests a ZQ initialization by setting the zq_req [2] parameter bit to b1.
- If the zq_on_sref_exit [2] parameter bit is set to b1, then a ZQ initialization will be issued whenever the memory devices exit self-refresh mode.

20.9.7 ZQ Reset

A ZQ reset calibration will calibrate each chip select in the system, one chip select at a time. The MC will issue the appropriate ZQRESET commands to each chip select.

A ZQRESET will be issued in the following cases:

- The user requests a ZQ reset by setting the zq_req [3] parameter bit to b1.
- If the zq_on_sref_exit [3] parameter bit is set to b1, then a ZQ reset will be issued whenever the memory devices exit self-refresh mode.

20.10 DDR PHY

The DDR PHY encapsulates all functionality required to interface to external DDR DRAM devices into a single module. This module is used to control the off-chip data capture and synchronization logic for the read data. This module performs the following functions:

- Contains all data registers used to launch data, address, and control signals to the DDR memory and the memory controller.
- Controls the off-chip data capture and synchronization logic for the read data.
- Includes DLL for timing.

20.10.1 High Level Block Diagram

DRAM MC uses a slice-based approach for the DDR PHY. Each slice manages a byte (8 bits) of data and its corresponding DQS and DM signals.

A high level block diagram of the PHY is provided in Figure 20-9.

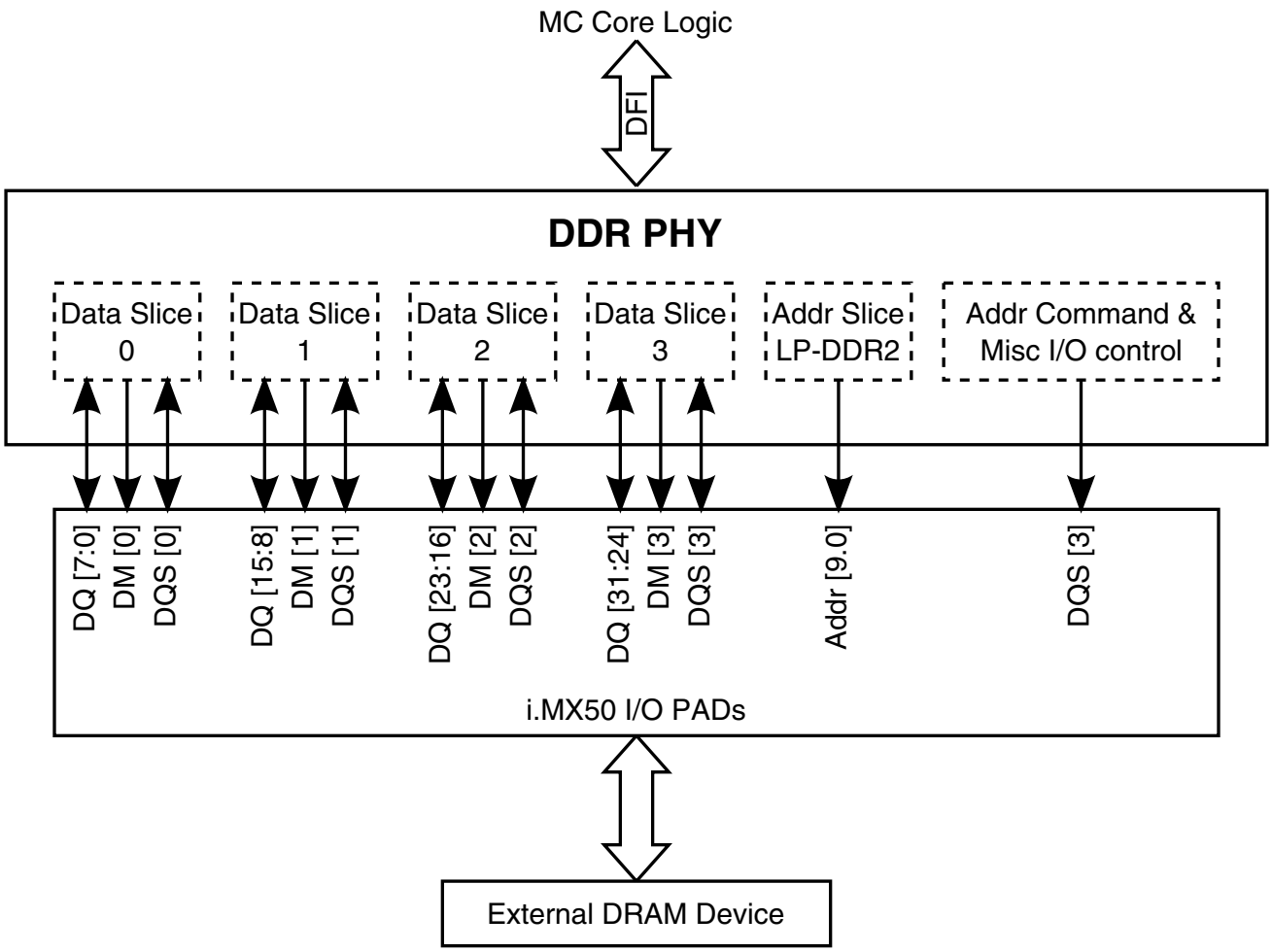


Figure 20-9. DDR PHY High Level Block Diagram

20.10.2 DFI

The interface between the DRAM MC core logic and the DDR_PHY is called "DFI". The "D" stands for "DRAM Controller" while "FI" is a reference to "PHY".

20.10.3 The I/O Timing of Address & Command

Figure 20-10 illustrates the I/O timing of Address and Command.

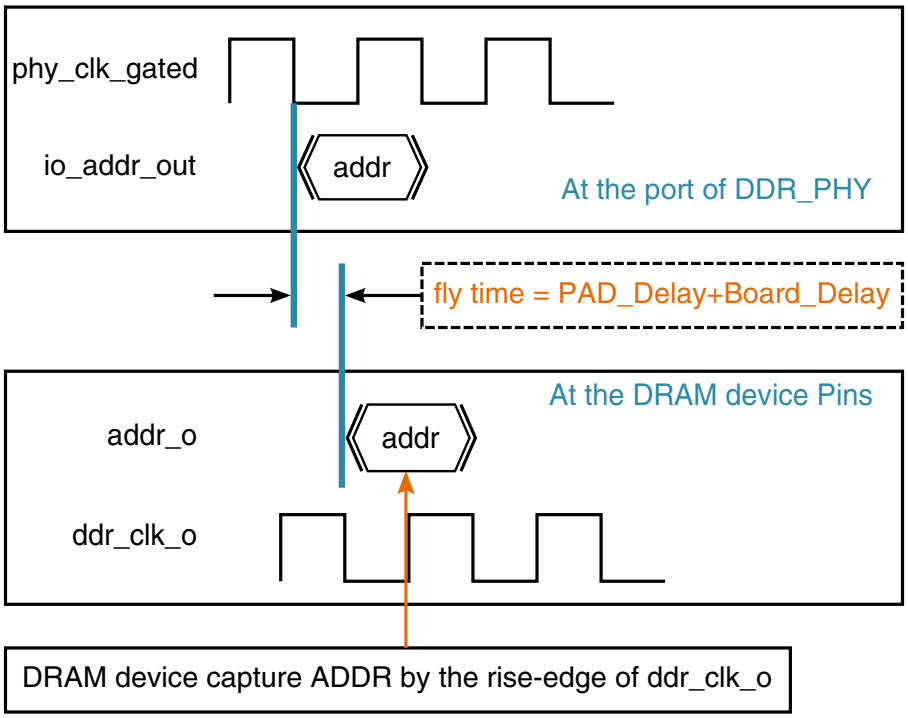


Figure 20-10. The I/O Timing of Address

The address & command signals including all DRAM interface signals except the DQ & DQS: clock-enable (CKE), chip-select, bank-address, row/cloumn_address, CASn, RASn, WEn, etc.

1. The Address & command signals is latched out by the fall-edge of PHY_CLK; In another word, the rise-edge of PHY_CLK is center-aligned with the output Address & Command signals.
2. The "fly-time" means the signals propagation time start from the DDR_PHY logic, go accross the output I/O (PAD), go through the signal trace on the board, then arrive the PIN of the DRAM device.
3. The DRAM device would capture the Address & Command signals by the rise edge of ddr_clk_o.
4. Figure 20-10 illustrates the address timing for Standard DDR2 & LP-DDR1. The timing of LP-DDR2 is quite different.

20.10.4 The Address Timing of LP-DDR2

There's a specail slice "Addr Slice" illustrated in Figure 20-9. It is designed to deal with the address[9:0] of LP-DDR2 device which require double-data-rate timing. Either the address[9:0] (in LP-DDR2 mode) or the ddr write data timing can be illustrated by Figure 20-14.

20.10.5 Data Slice Overview

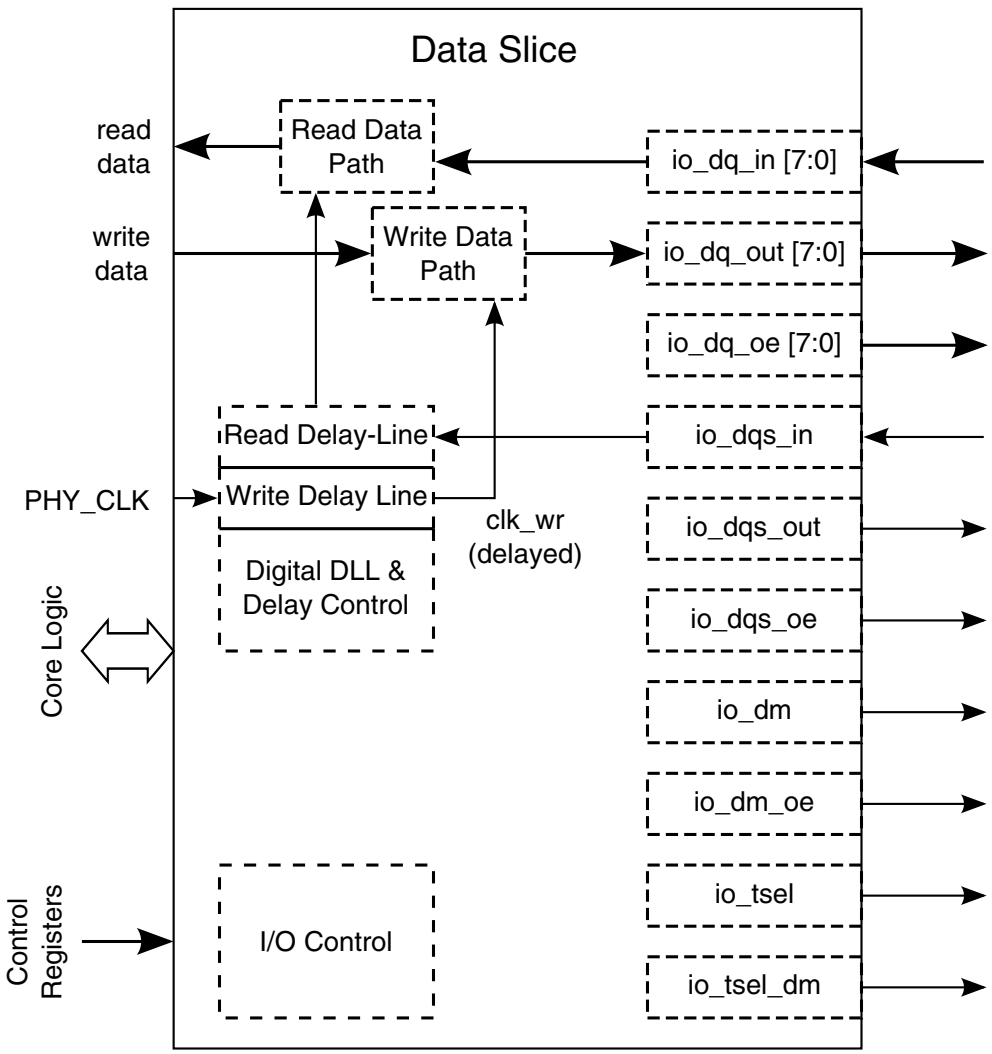


Figure 20-11. DDR PHY Data Slice

Each data-slice manages a byte (8-bit) of data and its corresponding signals.

The "Read Data Path" capture read data by latching it into read data buffers by both the posedge & negedge of "delayed_dqs". Then, the read data is synchronized from "delayed_dqs" clock domain to PHY_CLK domain.

The "Write Data Path" synchronize write data and write data mask (the "dm") from PHY_CLK domain to "dqs_out" domain.

The "Digital DLL" controls the delay values of "read delay-line" and "write delay-line". It can be bypassed and switched to manual delay control mode. In manual delay control mode, the delay values of read/write delay-line can be programmed into it's control registers separately.

20.10.6 Read Data Capture

When reading, DDR (dual data rate) devices send a data strobe (DQS) signal coincident with the read data. The edges of this DQS strobe are aligned (edge-aligned) with the data output by the DDR devices.

To latch read data into DRAM MC read data buffer, it requires that the latch clock edge is center-aligned with the data. Thus, a delayed version of the DQS strobe signal must be used to capture the data. Because the frequency of the DQS strobe signal is matched to the PHY_CLK, the delay is a relative number based on the period of the PHY_CLK. In the example shown in Figure 20-12, the delay is set to approximately 25% of the PHY_CLK cycle.

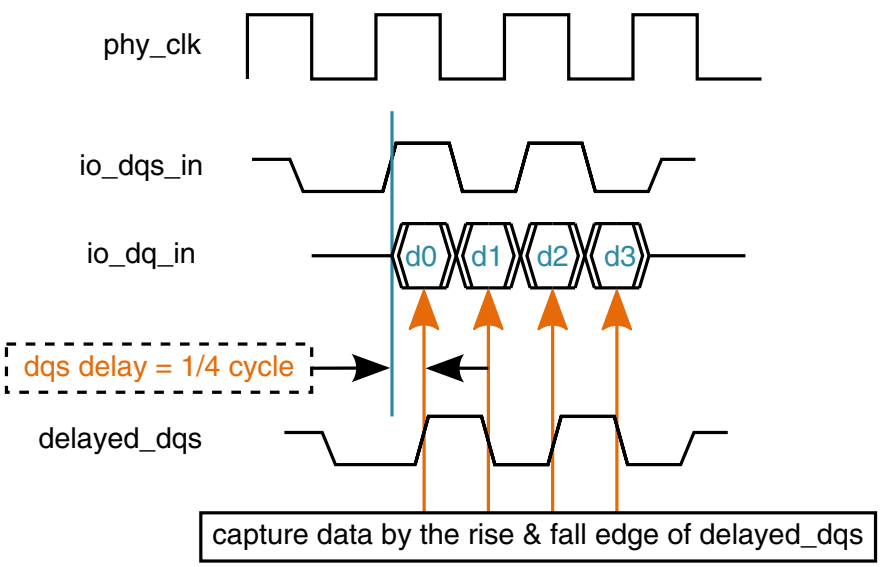


Figure 20-12. Read Data Capture

20.10.7 Synchronize Read Data From delayed_dqs To PHY_CLK domain

Read data is captured into data buffers by the "delayed_dqs". It must be synchronized into PHY_CLK domain, then returned to DRAM MC core logic.

Figure 20-13 illustrates how the read data from the external DRAM device is captured and synchronized to DRAM MC core logic.

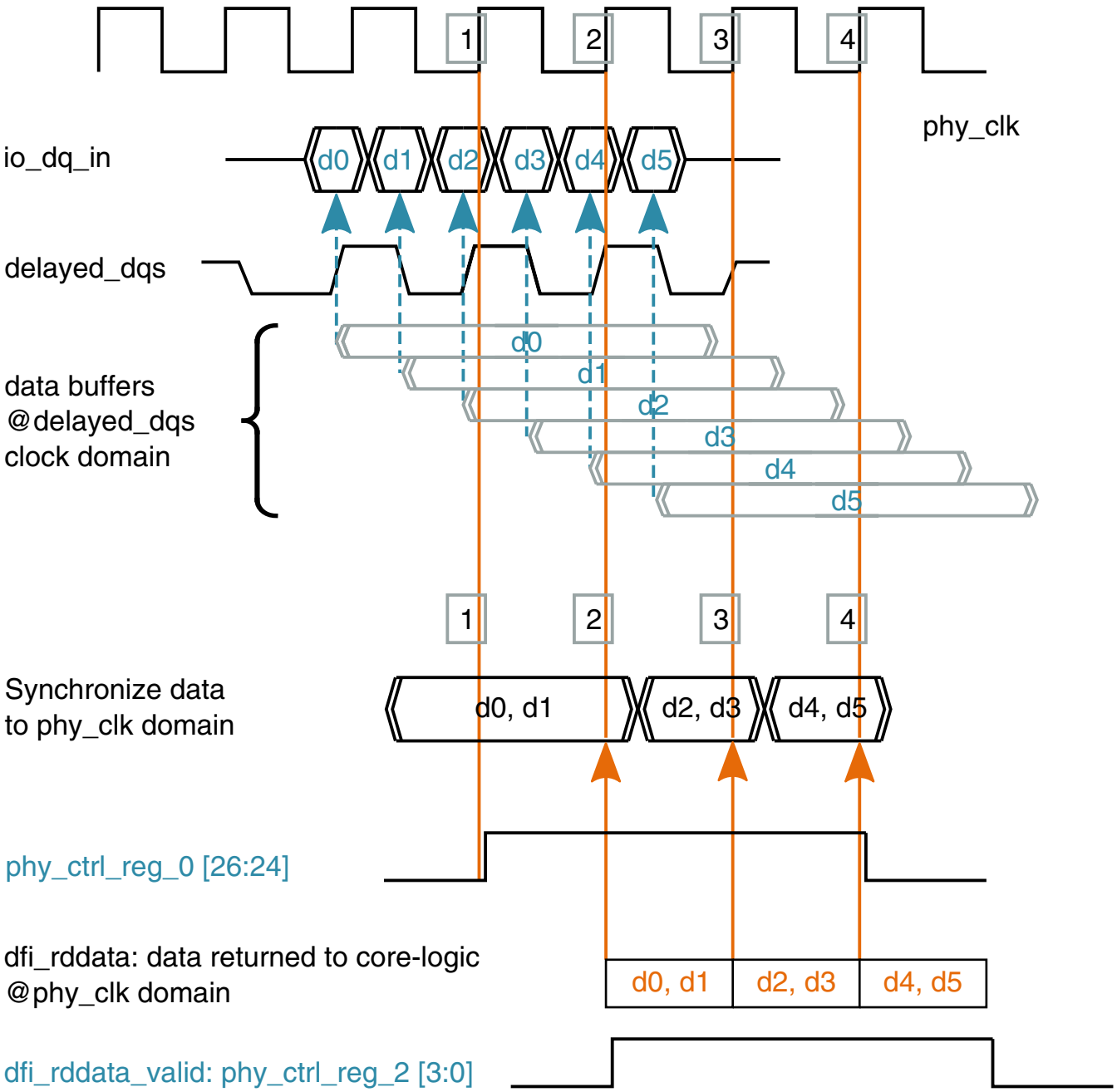


Figure 20-13. Synchronize Read Data

1. io_dq_in & io_dqs_in are not aligned with PHY_CLK in phase. Many factors affect the phase of io_dq/dqs_in, such as voltage, temperature, board layout, manufacture process, and so on.

2. Must capture the io_dq_in by delayed_dqs to meet the critical timing requirement in high frequency.
3. The data path width @PHY_CLK domain is twice of the data path width @delayed_dqs domain.
4. The DRAM MC core logic fetch read data by the rise-edge of PHY_CLK. The DRAM MC core logic fetch read data by two important signals: dfi_rddata & dfi_rddata_valid.
5. The user can program the timing position of "dfi_rddata & dfi_rddata_valid" by programmable registers "phy_ctrl_reg_0[26:24]" & "phy_ctrl_reg_2[3:0]". The unit is one cycle of PHY_CLK.
6. In this example, read data "d0,d1" getting valid before edge number "1" of PHY_CLK and is synchronized at edge "2" of PHY_CLK. Both the setup & hold time requirement are met which means read data could be fetched by core logic safely. Alternatively, read data can also be returned to core logic at edge number "1" of PHY_CLK, a cycle prior to the example. In this scenario, need to set the value of register "phy_ctrl_reg_0[26:24]" & "phy_ctrl_reg_2[3:0]" one number less. The, the "dfi_rddata" & "dfi_rddata_valid" would getting valid one cycle earlier. The benefit is that the read latency was shorten by one cycle, it help increasing the system performance. But, there's a timing risk on the "setup time", if the io_dq/dqs_in comes a little late, "setup time violation" may occur and the un-expected data would be returned to core logic, and, might lead to system crash.

20.10.8 Write Data Path

Figure 20-14 illustrates the write data path.

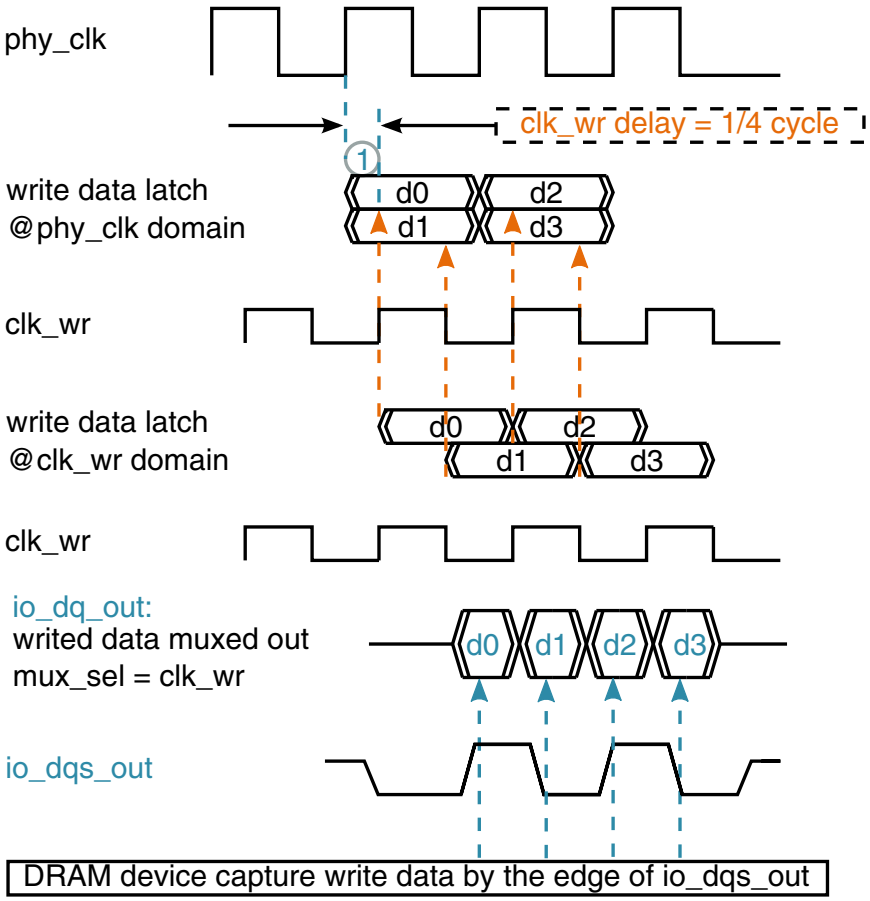


Figure 20-14. Write Data Path

NOTE

The marker "1" indicates that the "setup time" here for data "d0" is only "1/4 cycle", that would be critical in timing. The Low-Latency Option of Write Data Path , contains more detail on it.

The "io_dqs_out" and the PHY_CLK are edge-aligned in this DRAM MC design. This helps to meet the "tDQSS" timing requirement defined by the DRAM device.

The "io_dqs_out" and the "io_dq_out" are center-aligned, as required by the DRAM device.

20.10.8.1 The Low-Latency Option of Write Data Path

DDR_PHY provides 2 write data path options for the user :

1. The "Standard-Latency" option.

An extra latch at PHY_CLK domain is asserted into the write data path, right ahead of the latch at clk_wr domain (at the place of marker "1" in Figure 20-14). By this means, the asserted-latch and the latch@clk_wr are put back-to-back. The total delay in write data path is "1+1/4" cycle. It is safe in timing but decrease the performance by adding one cycle latency.

2. The "Low-Latency" option.

   The asserted latch for standard-latency is bypassed. The total delay in write data path is "1/4" cycle. There's risk in timing of write data path but it has higher performance than standard-latency case.

20.10.9 Digital DLL and the Delay-Line

Due to the asynchronous nature of the DRAM devices, the timing requirements for capturing and receiving data between the MC and the DRAM devices must be addressed. This DRAM MC contains a circuit that, in conjunction with I/O cell circuitry, can be used to meet the timing requirements for DRAM devices. The delay compensation circuit was designed with the following features:

- Programmable read strobe delay specified as a percentage of a clock cycle.
- Programmable write data delays specified as percentages of a clock cycle.
- Delay compensation circuit re-sync circuitry activated during refresh cycles to compensate for temperature and voltage drift.
- Separate delay chains for each read DQS signal from the DRAM devices.

The delay compensation circuitry relies on a master/slave approach. There is a master delay line which is used to determine how many delay elements constitute a complete cycle. This count is used, along with the programmable fractional delay settings, to determine the actual number of delay elements to program into the slave delay lines. The master and slave delay lines are identical. This approach allows the memory controller to observe a clock and then delay other signals a fixed percentage of that clock. The DLL logic does not actively generate clock signals.

The actual delay element for the delay lines is user-selectable.

Figure 20-15 shows the block diagram for a digital DLL.

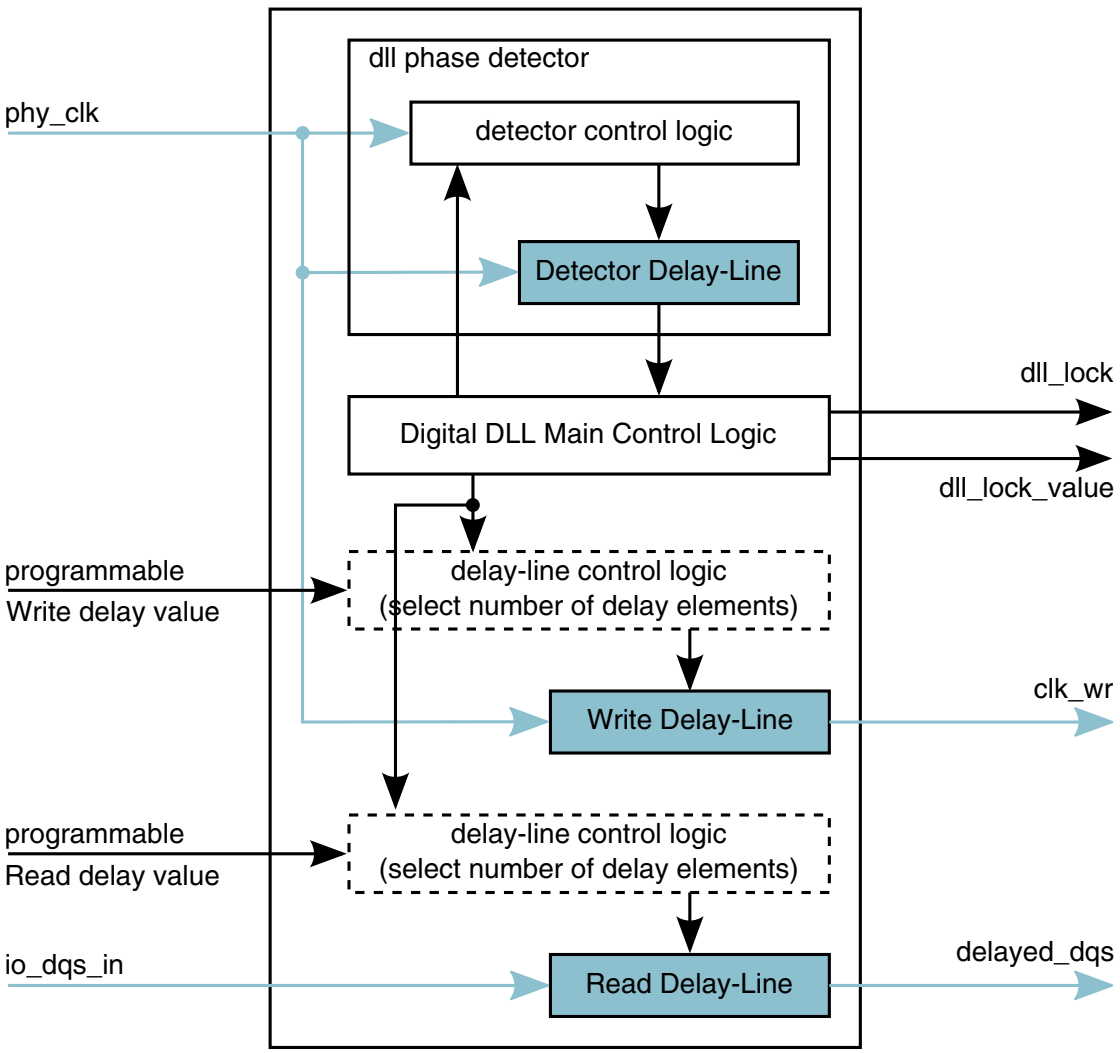


Figure 20-15. Digital DLL

The delay-line is comprised by 255 tiny delay-elements. The delay value of each delay element are almost the same. And the number of delay elements which is in use to form a delay result is configurable.

There are two modes for the delay-line control :

1. Auto-configure mode:

   This process begins by the "phase-detector". Once the phase-detector successfully complete the detection, it will raise the "dll_lock" signal and output "dll_lock_value" which indicates the delay of cone cycle. The number of elements that are needed to capture an entire clock cycle is then converted into an unsigned integer named encoder [7:0]. This integer is used as the dividend for the read and write delay parameters. The actual delay setting for the delay lines is calculated by multiplying

the encoder [7:0] integer by the parameter settings for each delay line and then dividing by 128 and rounding. These values are then encoded into a one-hot counter and updated at initialization and at every refresh interval.

2. Bypass mode, or manual configure mode:

   The phase detector can be bypassed. The number of delay elements for read & write delay line is programmed manually.

NOTE

In case the number of delay elements is fixed, the actual delay value (in nanoseconds) of a delay-line would change according to environmental conditions such as voltage, temperature, etc.

Using the auto-configure mode helps to eliminate the change of delay value because the reference delay itself also changes along with environmental conditions.

20.10.10 Configure the "output enable" of I/O Control

Some DRAM MC I/Os require a control signal named "output enable". Only when the "output enable" signal is active, the I/O can drive signals out, refer to Figure 20-16.

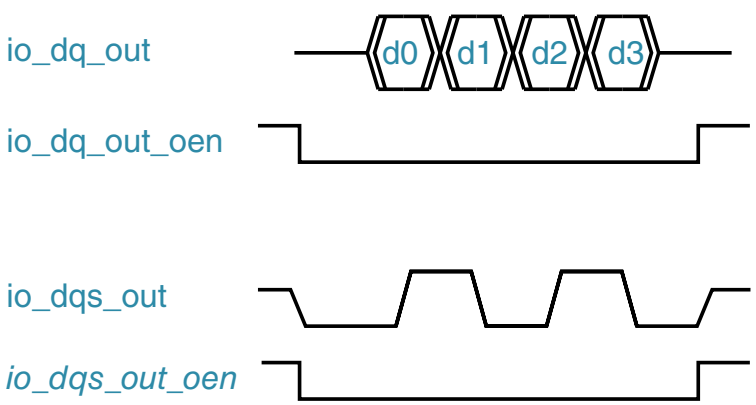


Figure 20-16. Output Enable of DRAM MC i/Os

There are 3 kinds of output enable signals in this DRAM MC design:

- io_dq_out_oen

  output enable for write data.

- io_dqm_oen

output enable for write data mask. Please configure it exactly the same as io_dq_out_oen.

- io_dqs_oen

  output enable for write data strobe.

The suffix "_oen" means the output enable signal is active low. The start time and the end time of a output enable signal can be configured separately. The start time and the end time can be adusted by 1/4 clock cycle in unit.

To meet the timing requirement of the I/O circuits, the start time of each output enable must be prior to it's corresponding output signal, and, the end time must be later than it's corresponding signal. In another word, the output enable signal must be "wider" than the output signal. The margin at start time is named "pre-amble" and the margin at end time is named "post-amble". In the example of Figure 20-16, pre-amble = 1/2 cycle while post-amble = 1/4 cycle.

20.11 Programmable Registers

DRAM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1400_0000	DRAM CTL Register 00 (DRAM_CTL00)	32	R/W	2041_0000h	20.11.1/ 928
1400_0004	DRAM CTL Register 01 (DRAM_CTL01)	32	R/W	0002_0A0Fh	20.11.2/ 929
1400_0008	DRAM CTL Register 02 (DRAM_CTL02)	32	R/W	0000_0000h	20.11.3/ 929
1400_000C	DRAM CTL Register 03 (DRAM_CTL03)	32	R/W	0000_0000h	20.11.4/ 930
1400_0010	DRAM CTL Register 04 (DRAM_CTL04)	32	R/W	0000_0000h	20.11.5/ 930
1400_0014	DRAM CTL Register 05 (DRAM_CTL05)	32	R/W	0000_0000h	20.11.6/ 931
1400_0018	DRAM CTL Register 06 (DRAM_CTL06)	32	R/W	0000_0000h	20.11.7/ 932
1400_001C	DRAM CTL Register 07 (DRAM_CTL07)	32	R/W	0000_0000h	20.11.8/ 933

Table continues on the next page...

DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1400_0020	DRAM CTL Register 08 (DRAM_CTL08)	32	R/W	0000_0000h	20.11.9/ 934
1400_0024	DRAM CTL Register 09 (DRAM_CTL09)	32	R/W	0000_0000h	20.11.10/ 935
1400_0028	DRAM CTL Register 10 (DRAM_CTL10)	32	R/W	0000_0000h	20.11.11/ 937
1400_002C	DRAM CTL Register 11 (DRAM_CTL11)	32	R/W	0000_0000h	20.11.12/ 938
1400_0030	DRAM CTL Register 12 (DRAM_CTL12)	32	R/W	0000_0000h	20.11.13/ 939
1400_0034	DRAM CTL Register 13 (DRAM_CTL13)	32	R/W	0000_0000h	20.11.14/ 940
1400_0038	DRAM CTL Register 14 (DRAM_CTL14)	32	R/W	0000_0000h	20.11.15/ 941
1400_003C	DRAM CTL Register 15 (DRAM_CTL15)	32	R/W	0000_0000h	20.11.16/ 942
1400_0040	DRAM CTL Register 16 (DRAM_CTL16)	32	R/W	0000_0000h	20.11.17/ 943
1400_0044	DRAM CTL Register 17 (DRAM_CTL17)	32	R/W	0000_0000h	20.11.18/ 944
1400_0048	DRAM CTL Register 18 (DRAM_CTL18)	32	R/W	0000_0000h	20.11.19/ 945
1400_004C	DRAM CTL Register 19 (DRAM_CTL19)	32	R/W	0000_0000h	20.11.20/ 946
1400_0050	DRAM CTL Register 20 (DRAM_CTL20)	32	R/W	0000_0000h	20.11.21/ 947
1400_0054	DRAM CTL Register 21 (DRAM_CTL21)	32	R/W	0000_0000h	20.11.22/ 948
1400_0058	DRAM CTL Register 22 (DRAM_CTL22)	32	R/W	0000_0000h	20.11.23/ 949
1400_005C	DRAM CTL Register 23 (DRAM_CTL23)	32	R/W	0000_0000h	20.11.24/ 950
1400_0060	DRAM CTL Register 24 (DRAM_CTL24)	32	R/W	0000_0000h	20.11.25/ 951
1400_0064	DRAM CTL Register 25 (DRAM_CTL25)	32	R/W	0000_0000h	20.11.26/ 952
1400_0068	DRAM CTL Register 26 (DRAM_CTL26)	32	R/W	0000_0000h	20.11.27/ 952
1400_006C	DRAM CTL Register 27 (DRAM_CTL27)	32	R/W	0000_0000h	20.11.28/ 953

Table continues on the next page...

DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1400_0070	DRAM CTL Register 28 (DRAM_CTL28)	32	R/W	0000_0000h	20.11.29/ 954
1400_0074	DRAM CTL Register 29 (DRAM_CTL29)	32	R/W	0000_0000h	20.11.30/ 955
1400_0078	DRAM CTL Register 30 (DRAM_CTL30)	32	R/W	0000_0000h	20.11.31/ 955
1400_007C	DRAM CTL Register 31 (DRAM_CTL31)	32	R/W	0000_0000h	20.11.32/ 956
1400_0080	DRAM CTL Register 32 (DRAM_CTL32)	32	R/W	0000_0000h	20.11.33/ 956
1400_0084	DRAM CTL Register 33 (DRAM_CTL33)	32	R/W	0000_0000h	20.11.34/ 957
1400_0088	DRAM CTL Register 34 (DRAM_CTL34)	32	R/W	0000_0000h	20.11.35/ 958
1400_008C	DRAM CTL Register 35 (DRAM_CTL35)	32	R/W	0000_0000h	20.11.36/ 959
1400_0090	DRAM CTL Register 36 (DRAM_CTL36)	32	R/W	0000_0000h	20.11.37/ 960
1400_0094	DRAM CTL Register 37 (DRAM_CTL37)	32	R/W	0000_0000h	20.11.38/ 961
1400_0098	DRAM CTL Register 38 (DRAM_CTL38)	32	R/W	0000_0000h	20.11.39/ 963
1400_009C	DRAM CTL Register 39 (DRAM_CTL39)	32	R/W	0000_0000h	20.11.40/ 964
1400_00A0	DRAM CTL Register 40 (DRAM_CTL40)	32	R/W	0000_0000h	20.11.41/ 965
1400_00A4	DRAM CTL Register 41 (DRAM_CTL41)	32	R/W	0000_0000h	20.11.42/ 967
1400_00A8	DRAM CTL Register 42 (DRAM_CTL42)	32	R/W	0000_0000h	20.11.43/ 968
1400_00AC	DRAM CTL Register 43 (DRAM_CTL43)	32	R/W	0000_0000h	20.11.44/ 969
1400_00B0	DRAM CTL Register 44 (DRAM_CTL44)	32	R/W	0000_0000h	20.11.45/ 969
1400_00B4	DRAM CTL Register 45 (DRAM_CTL45)	32	R/W	0000_0000h	20.11.46/ 970
1400_00B8	DRAM CTL Register 46 (DRAM_CTL46)	32	R/W	0000_0000h	20.11.47/ 970
1400_00BC	DRAM CTL Register 47 (DRAM_CTL47)	32	R/W	0000_0000h	20.11.48/ 971

Table continues on the next page...

DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1400_00C0	DRAM CTL Register 48 (DRAM_CTL48)	32	R/W	0000_0000h	20.11.49/ 971
1400_00C4	DRAM CTL Register 49 (DRAM_CTL49)	32	R/W	0000_0000h	20.11.50/ 972
1400_00C8	DRAM CTL Register 50 (DRAM_CTL50)	32	R/W	0000_0000h	20.11.51/ 973
1400_00CC	DRAM CTL Register 51 (DRAM_CTL51)	32	R/W	0000_0000h	20.11.52/ 974
1400_00D0	DRAM CTL Register 52 (DRAM_CTL52)	32	R/W	0000_0000h	20.11.53/ 975
1400_00D4	DRAM CTL Register 53 (DRAM_CTL53)	32	R/W	0000_0000h	20.11.54/ 976
1400_00D8	DRAM CTL Register 54 (DRAM_CTL54)	32	R/W	0000_0000h	20.11.55/ 977
1400_00DC	DRAM CTL Register 55 (DRAM_CTL55)	32	R/W	0000_0000h	20.11.56/ 977
1400_00E0	DRAM CTL Register 56 (DRAM_CTL56)	32	R/W	0000_0000h	20.11.57/ 978
1400_00E4	DRAM CTL Register 57 (DRAM_CTL57)	32	R/W	0000_0000h	20.11.58/ 979
1400_00E8	DRAM CTL Register 58 (DRAM_CTL58)	32	R/W	0000_0000h	20.11.59/ 980
1400_00EC	DRAM CTL Register 59 (DRAM_CTL59)	32	R/W	0000_0000h	20.11.60/ 981
1400_00F0	DRAM CTL Register 60 (DRAM_CTL60)	32	R/W	0000_0000h	20.11.61/ 982
1400_00F4	DRAM CTL Register 61 (DRAM_CTL61)	32	R/W	0000_0000h	20.11.62/ 983
1400_00F8	DRAM CTL Register 62 (DRAM_CTL62)	32	R/W	0000_0000h	20.11.63/ 984
1400_00FC	DRAM CTL Register 63 (DRAM_CTL63)	32	R/W	0000_0000h	20.11.64/ 985
1400_0100	DRAM CTL Register 64 (DRAM_CTL64)	32	R/W	0000_0000h	20.11.65/ 986
1400_0104	DRAM CTL Register 65 (DRAM_CTL65)	32	R/W	0000_0000h	20.11.66/ 987
1400_0108	DRAM CTL Register 66 (DRAM_CTL66)	32	R/W	0000_0400h	20.11.67/ 988
1400_010C	DRAM CTL Register 67 (DRAM_CTL67)	32	R/W	0000_0000h	20.11.68/ 988

Table continues on the next page...

DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1400_0110	DRAM CTL Register 68 (DRAM_CTL68)	32	R/W	0000_0000h	20.11.69/ 989
1400_0114	DRAM CTL Register 69 (DRAM_CTL69)	32	R/W	0000_0000h	20.11.70/ 989
1400_0118	DRAM CTL Register 70 (DRAM_CTL70)	32	R/W	0000_0000h	20.11.71/ 990
1400_011C	DRAM CTL Register 71 (DRAM_CTL71)	32	R/W	0000_0000h	20.11.72/ 991
1400_0120	DRAM CTL Register 72 (DRAM_CTL72)	32	R/W	0000_0000h	20.11.73/ 992
1400_0124	DRAM CTL Register 73 (DRAM_CTL73)	32	R/W	0000_0000h	20.11.74/ 993
1400_0128	DRAM CTL Register 74 (DRAM_CTL74)	32	R/W	0000_0000h	20.11.75/ 994
1400_012C	DRAM CTL Register 75 (DRAM_CTL75)	32	R/W	0000_0000h	20.11.76/ 995
1400_0130	DRAM CTL Register 76 (DRAM_CTL76)	32	R/W	0000_0000h	20.11.77/ 995
1400_0134	DRAM CTL Register 77 (DRAM_CTL77)	32	R/W	0000_0000h	20.11.78/ 996
1400_0138	DRAM CTL Register 78 (DRAM_CTL78)	32	R/W	0000_0000h	20.11.79/ 996
1400_013C	DRAM CTL Register 79 (DRAM_CTL79)	32	R/W	0000_0000h	20.11.80/ 997
1400_0140	DRAM CTL Register 80 (DRAM_CTL80)	32	R/W	0000_0000h	20.11.81/ 998
1400_0144	DRAM CTL Register 81 (DRAM_CTL81)	32	R/W	0000_0000h	20.11.82/ 998
1400_0148	DRAM CTL Register 82 (DRAM_CTL82)	32	R/W	0000_0000h	20.11.83/ 998
1400_014C	DRAM CTL Register 83 (DRAM_CTL83)	32	R/W	0000_0000h	20.11.84/ 999
1400_0150	DRAM CTL Register 84 (DRAM_CTL84)	32	R/W	0000_0000h	20.11.85/ 999
1400_0154	DRAM CTL Register 85 (DRAM_CTL85)	32	R/W	0000_0000h	20.11.86/ 1000
1400_0158	DRAM CTL Register 86 (DRAM_CTL86)	32	R/W	0000_0000h	20.11.87/ 1000
1400_0200	DRAM PHY Register 00 (DRAM_PHY00)	32	R/W	0000_0000h	20.11.88/ 1001

Table continues on the next page...

DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1400_0204	DRAM PHY Register 01 (DRAM_PHY01)	32	R/W	0000_0000h	20.11.89/ 1001
1400_0208	DRAM PHY Register 02 (DRAM_PHY02)	32	R/W	0000_0000h	20.11.90/ 1002
1400_020C	DRAM PHY Register 03 (DRAM_PHY03)	32	R/W	0000_0000h	20.11.91/ 1004
1400_0210	DRAM PHY Register 04 (DRAM_PHY04)	32	R/W	0000_0000h	20.11.92/ 1006
1400_0214	DRAM PHY Register 05 (DRAM_PHY05)	32	R/W	0000_0000h	20.11.93/ 1006
1400_0218	DRAM PHY Register 06 (DRAM_PHY06)	32	R/W	0000_0000h	20.11.94/ 1006
1400_021C	DRAM PHY Register 07 (DRAM_PHY07)	32	R/W	0000_0000h	20.11.95/ 1007
1400_0220	DRAM PHY Register 08 (DRAM_PHY08)	32	R/W	0000_0000h	20.11.96/ 1007
1400_0224	DRAM PHY Register 09 (DRAM_PHY09)	32	R/W	0000_0000h	20.11.97/ 1008
1400_0228	DRAM PHY Register 10 (DRAM_PHY10)	32	R/W	0000_0000h	20.11.98/ 1008
1400_022C	DRAM PHY Register 11 (DRAM_PHY11)	32	R/W	0000_0000h	20.11.99/ 1008
1400_0230	DRAM PHY Register 12 (DRAM_PHY12)	32	R/W	0000_0000h	20.11.100/ 1009
1400_0234	DRAM PHY Register 13 (DRAM_PHY13)	32	R/W	0000_0000h	20.11.101/ 1009
1400_0238	DRAM PHY Register 14 (DRAM_PHY14)	32	R/W	0000_0000h	20.11.102/ 1010
1400_023C	DRAM PHY Register 15 (DRAM_PHY15)	32	R/W	0000_0000h	20.11.103/ 1011
1400_0240	DRAM PHY Register 16 (DRAM_PHY16)	32	R/W	0000_0000h	20.11.104/ 1012
1400_0244	DRAM PHY Register 17 (DRAM_PHY17)	32	R/W	0000_0000h	20.11.105/ 1012
1400_0248	DRAM PHY Register 18 (DRAM_PHY18)	32	R/W	0000_0000h	20.11.106/ 1013
1400_024C	DRAM PHY Register 19 (DRAM_PHY19)	32	R/W	0000_0000h	20.11.107/ 1013
1400_0250	DRAM PHY Register 20 (DRAM_PHY20)	32	R/W	0000_0000h	20.11.108/ 1014

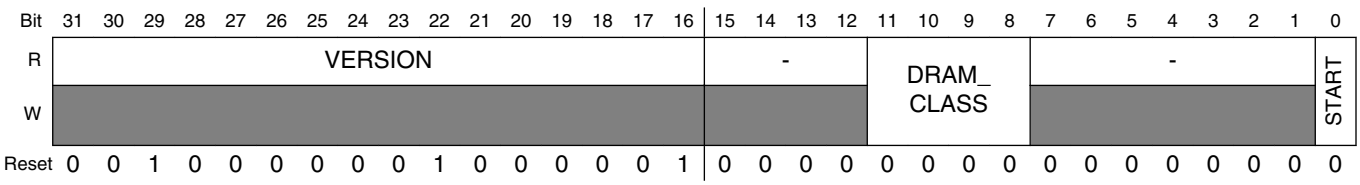
Table continues on the next page...

DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1400_0254	DRAM PHY Register 21 (DRAM_PHY21)	32	R/W	0000_0000h	20.11.109/ 1014
1400_0258	DRAM PHY Register 22 (DRAM_PHY22)	32	R/W	0000_0000h	20.11.110/ 1014
1400_025C	DRAM PHY Register 23 (DRAM_PHY23)	32	R/W	0000_0000h	20.11.111/ 1015
1400_0260	DRAM PHY Register 24 (DRAM_PHY24)	32	R/W	0000_0000h	20.11.112/ 1015
1400_0264	DRAM PHY Register 25 (DRAM_PHY25)	32	R/W	0000_0000h	20.11.113/ 1016
1400_0268	DRAM PHY Register 26 (DRAM_PHY26)	32	R/W	0000_0000h	20.11.114/ 1017
1400_026C	DRAM PHY Register 27 (DRAM_PHY27)	32	R/W	0000_0000h	20.11.115/ 1017
1400_0270	DRAM PHY Register 28 (DRAM_PHY28)	32	R/W	0000_0000h	20.11.116/ 1018
1400_0274	DRAM PHY Register 29 (DRAM_PHY29)	32	R/W	0000_0000h	20.11.117/ 1018
1400_0278	DRAM PHY Register 30 (DRAM_PHY30)	32	R/W	0000_0000h	20.11.118/ 1018
1400_027C	DRAM PHY Register 31 (DRAM_PHY31)	32	R/W	0000_0000h	20.11.119/ 1019
1400_0280	DRAM PHY Register 32 (DRAM_PHY32)	32	R/W	0000_0000h	20.11.120/ 1019
1400_0284	DRAM PHY Register 33 (DRAM_PHY33)	32	R/W	0000_0000h	20.11.121/ 1020
1400_0288	DRAM PHY Register 34 (DRAM_PHY34)	32	R/W	0000_0000h	20.11.122/ 1020
1400_028C	DRAM PHY Register 35 (DRAM_PHY35)	32	R/W	0000_0000h	20.11.123/ 1020
1400_0290	DRAM PHY Register 36 (DRAM_PHY36)	32	R/W	0000_0000h	20.11.124/ 1021
1400_0294	DRAM PHY Register 37 (DRAM_PHY37)	32	R/W	0000_0000h	20.11.125/ 1021
1400_0298	DRAM PHY Register 38 (DRAM_PHY38)	32	R/W	0000_0000h	20.11.126/ 1022

20.11.1 DRAM CTL Register 00 (DRAM_CTL00)

Address: DRAM_CTL00 is 1400_0000h base + 0h offset = 1400_0000h



DRAM_CTL00 field descriptions

Field	Description
31–16 VERSION	Controller version number. READ-ONLY
15–12 -	Reserved.
11–8 DRAM_CLASS	Defines the mode of operation of the controller. b0001 = LPDDR1 - Mobile DDR1 (supports burst lengths of 2, 4 or 8) b0100 = DDR2 (supports burst length of 4) b0101 = LPDDR2 (supports burst lengths of 4 or 8) All other settings reserved
7–1 -	Reserved.
0 START	Prior to setting this parameter to b1, the memory controller will not issue any commands to the DRAM devices or respond to any signal activity except for reading and writing PIO parameters. Once this parameter is set to b1, the memory controller will respond to inputs from the bus interface and begin to process memory access commands. Note that resetting this parameter to b0 will not shut off traffic. However, cycling this parameter to b0 and then resetting it to b1 will reset the digital DLL (if present) to a new input clock frequency if desired. This protocol is described in detail in Section 13.2, Changing the Input Clock Frequency. Note: Until the initialization complete bit (bit 4) is set in the int_status parameter and the dfi_init_complete signal is asserted from the PHY, commands will not be accepted into the controller. b0 = Controller is not in active mode. b1 = Initiate active mode for the memory controller.

20.11.2 DRAM CTL Register 01 (DRAM_CTL01)

Address: DRAM_CTL01 is 1400_0000h base + 4h offset = 1400_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-														MAX_CS_REG	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-				MAX_COL_REG				-				MAX_ROW_REG			
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	0	1	1	1	1

DRAM_CTL01 field descriptions

Field	Description
31–18 -	Reserved.
17–16 MAX_CS_REG	Displays the maximum number of chip selects configured for this MC. READ-ONLY.
15–12 -	Reserved.
11–8 MAX_COL_REG	Defines the maximum width of column address in the DRAM devices. This value can be used to set the column_size parameter. READ-ONLY. column_size = max_col_reg - (number of column bits in memory device)
7–4 -	Reserved.
3–0 MAX_ROW_ REG	Defines the maximum width of the memory address bus for the MC. Note: For LPDDR2 memories, the parameter addr_pins holds the difference between the maximum number of row pins configured and the actual number of row pins being used. This value can be used to set the addr_pins parameter. READ-ONLY. addr_pins = max_row_reg - (number of row bits in memory device)

20.11.3 DRAM CTL Register 02 (DRAM_CTL02)

Address: DRAM_CTL02 is 1400_0000h base + 8h offset = 1400_0008h

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

DRAM_CTL02 field descriptions

Field	Description
31–24 -	Reserved.
23–0 TINIT	Defines the DRAM initialization time, in cycles. This refers to the time required for clocks to be started and stabilized before clock enable becomes active. The user may set this value to CK stabilization time of 10 ns or 5 clocks before CKE active. Note: Since the counter is initially loaded with the value tinit-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum. For LPDDR2 memories, this correlates to tinit1.

20.11.4 DRAM CTL Register 03 (DRAM_CTL03)

Address: DRAM_CTL03 is 1400_0000h base + Ch offset = 1400_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								TINIT3																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL03 field descriptions

Field	Description
31–24 -	Reserved.
23–0 TINIT3	Defines the number of cycles required from CKE assertion to memory reset. This parameter should be programmed to the tinit3 value from the memory specification. Note: Since the counter is initially loaded with the value tinit3-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101)

20.11.5 DRAM CTL Register 04 (DRAM_CTL04)

Address: DRAM_CTL04 is 1400_0000h base + 10h offset = 1400_0010h

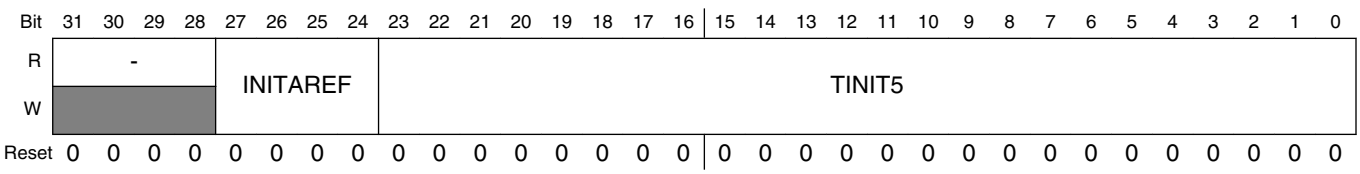
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								TINIT4																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL04 field descriptions

Field	Description
31–24 -	Reserved.
23–0 TINIT4	Defines the number of cycles required from memory reset to an MRR command. This parameter should be programmed to the tinit4 value from the memory specification. Note: Since the counter is initially loaded with the value tinit4-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101)

20.11.6 DRAM CTL Register 05 (DRAM_CTL05)

Address: DRAM_CTL05 is 1400_0000h base + 14h offset = 1400_0014h



DRAM_CTL05 field descriptions

Field	Description
31–28 -	Reserved.
27–24 INITAREF	Defines the number of auto-refresh commands needed by the DRAM devices to satisfy the initialization sequence. This parameter is only applicable when the MC is programmed for the following memory systems: DDR2 (dram_class = b0100) LPDDR1 (dram_class = b0001)
23–0 TINIT5	Defines the maximum number of cycles required from memory reset to initialization complete. This parameter should be programmed to the tinit5 value from the memory specification. Note: Since the counter is initially loaded with the value tinit5-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101)

20.11.7 DRAM CTL Register 06 (DRAM_CTL06)

Address: DRAM_CTL06 is 1400_0000h base + 18h offset = 1400_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-			TCCD					-				WRLAT				-			CASLAT_LIN_ GATE					-			CASLAT_LIN				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL06 field descriptions

Field	Description
31–29 -	Reserved.
28–24 TCCD	Defines the minimum delay between CAS commands, in cycles. This value is loaded into a counter when a memory burst is issued and a new command may be issued when the counter reaches 0. Note: Since the counter is initially loaded with the value tccd-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
23–20 -	Reserved.
19–16 WRLAT	Defines the write latency from when the write command is issued to the time the write data is presented to the DRAM devices, in cycles.
15–13 -	Reserved.
12–8 CASLAT_LIN_ GATE	This parameter has no meaning to this MC
7–5 -	Reserved.
4–0 CASLAT_LIN	Sets the CAS latency linear value. Note: The memory CAS latency is programmed via the mode registers. This parameter is used by the MC and should be programmed to match the memory CAS latency. Not all linear values will be supported for the memory devices being used. Refer to the specification for the memory devices being used. [Bit 0] = 1/2 cycle increment. This bit should be cleared to b0 if the memory system is LPDDR1. This bit is only applicable when the MC is programmed for the following memory systems: LPDDR1 (dram_class = b0001) b0 = No 1/2 cycle increment b1 = Additional 1/2 cycle increment on the CAS latency. [Bits 4:1] = CAS latency linear value. b0000 = Reserved b0001 = 1 cycle b0010 = 2 cycles

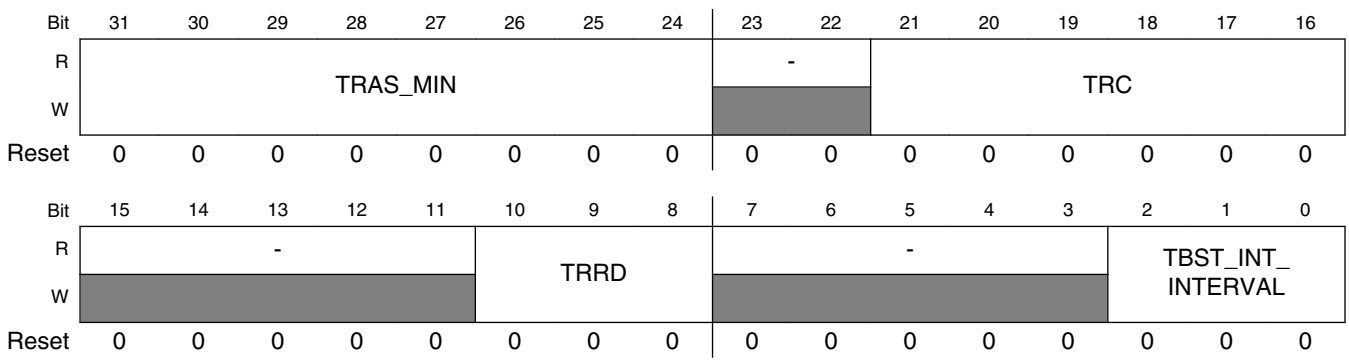
Table continues on the next page...

DRAM_CTL06 field descriptions (continued)

Field	Description
	b0011 = 3 cycles b0100 = 4 cycles b0101 = 5 cycles b0110 = 6 cycles b0111 = 7 cycles Additional values of caslat_lin are supported, in full-cycle increments up to 15 cycles. All other settings Reserved.

20.11.8 DRAM CTL Register 07 (DRAM_CTL07)

Address: DRAM_CTL07 is 1400_0000h base + 1Ch offset = 1400_001Ch



DRAM_CTL07 field descriptions

Field	Description
31–24 TRAS_MIN	Defines the DRAM minimum row activate time, in cycles. Since the counter is initially loaded with the value tras_min-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. In general, it is suggested that this parameter be programmed with a value of at least 0x3.
23–22 -	Reserved.
21–16 TRC	Defines the DRAM period between active commands for the same bank, in cycles. Note: Since the counter is initially loaded with the value trc-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
15–11 -	Reserved.
10–8 TRRD	Defines the DRAM activate to activate delay for different banks, in cycles. Note: Since the counter is initially loaded with the value trrd-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.

Table continues on the next page...

DRAM_CTL07 field descriptions (continued)

Field	Description
7–3 -	Reserved.
2–0 TBST_INT_ INTERVAL	Defines the memory burst interrupt interval. This parameter is only relevant if the burst has not completed. This value is loaded into a counter when a memory burst is issued and another command may only interrupt the current memory burst when this counter value hits 0. If the counter value hits 0 and the memory burst has not completed, the counter will be reset with the tbst_int_interval value. If a command is in progress and the memory burst has not completed, another command may only be issued on cycles after the parameter tccd value cycles have elapsed since the last CAS command and this counter value hits 0. For example with a DDR memory, if the memory burst length (the bstlen parameter) is 8, tccd is 2, tbst_int_interval is 2 and a CAS command was issued on cycle 0, another CAS command could interrupt the current memory burst on cycle 2. After cycle 3, the current memory burst will complete and this parameter would not be relevant. If instead the tbst_int_interval was 1 for the same system, then the command could interrupt on cycles 2 or 3. Note: Since the counter is initially loaded with the value tbst_int_interval-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.

20.11.9 DRAM CTL Register 08 (DRAM_CTL08)

Address: DRAM_CTL08 is 1400_0000h base + 20h offset = 1400_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-			TMRD					-					TRTP			-			TRP					-				TWTR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL08 field descriptions

Field	Description
31–29 -	Reserved.
28–24 TMRD	Defines the minimum number of cycles required between two mode register write commands. This is the time required to complete the write operation to the mode register. There are two parameters that control the timing after a mode register write: tmod and tmrd. This parameter (tmrd) is typically the shorter of the two timing delays. The MC will wait this delay after a mode register command before sending the next mode register command. The MC does not need to wait for the longer tmod timer to expire since termination is not used for mode register commands. Note1: Since the counter is initially loaded with the value tmrd-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum. Note2: For LPDDR2 memories, this parameter should be set to the tmrw value from the memory specification.
23–19 -	Reserved.

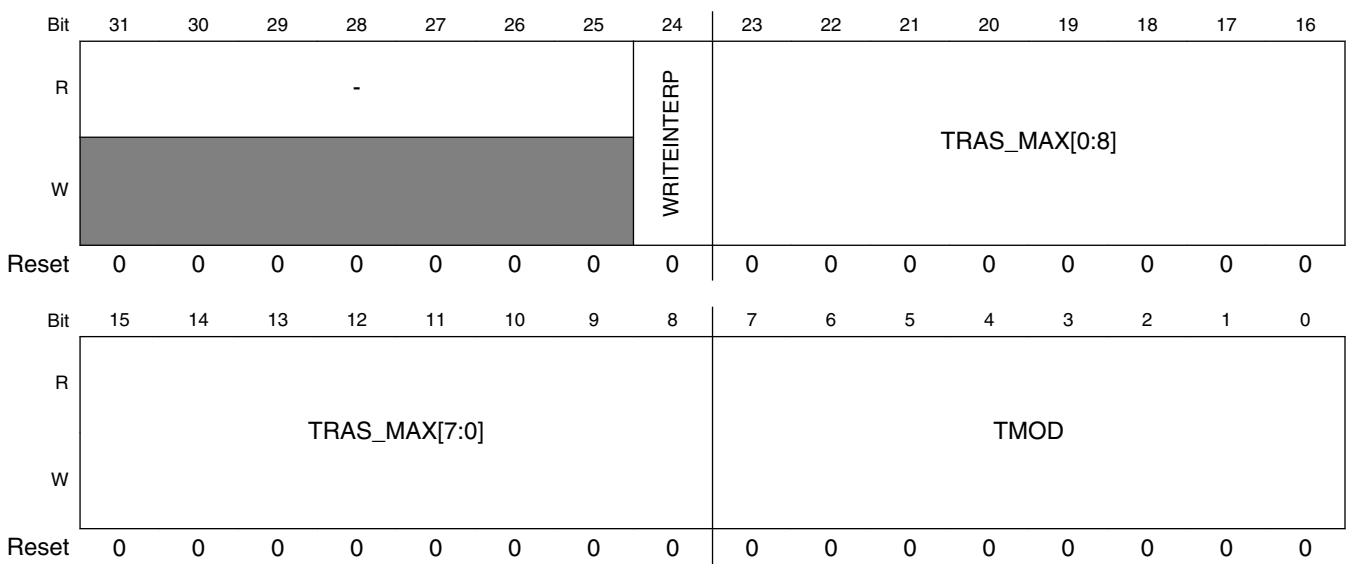
Table continues on the next page...

DRAM_CTL08 field descriptions (continued)

Field	Description
18–16 TRTP	Defines the DRAM tRTP (read to precharge time) parameter, in cycles. Note: Since the counter is initially loaded with the value trtp-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
15–13 -	Reserved.
12–8 TRP	Defines the DRAM pre-charge command time, in cycles. Note: Since the counter is initially loaded with the value trp-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
7–4 -	Reserved.
3–0 TWTR	Sets the number of cycles needed to switch from a write to a read operation. Note: Since the counter is initially loaded with the value twtr-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.

20.11.10 DRAM CTL Register 09 (DRAM_CTL09)

Address: DRAM_CTL09 is 1400_0000h base + 24h offset = 1400_0024h



DRAM_CTL09 field descriptions

Field	Description
31–25 -	Reserved.

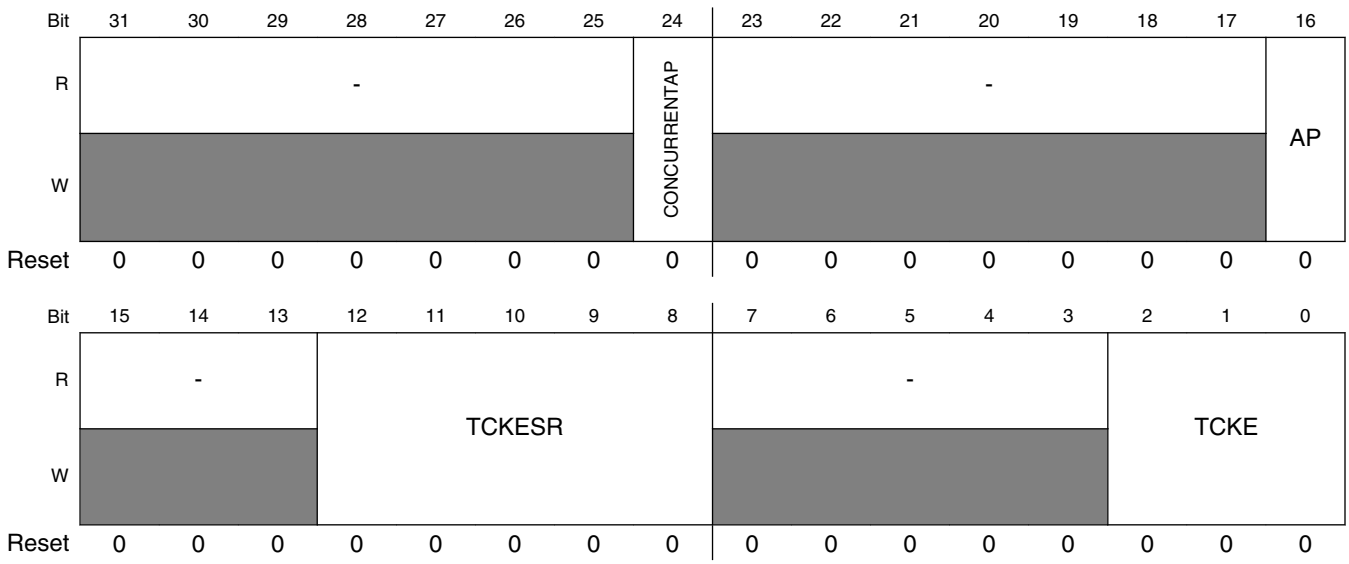
Table continues on the next page...

DRAM_CTL09 field descriptions (continued)

Field	Description
24 WRITEINTERP	Defines whether the memory controller can interrupt a write burst with a read command. Some memory devices do not allow this functionality. For DDR2 memories: This parameter must be cleared to b0. For LPDDR1 memories: Consult the memory specification for the setting for this parameter. For LPDDR2 memories: This parameter must be cleared to b0. For all memory types: b0 = The device does not support read commands interrupting write commands. b1 = The device does support read commands interrupting write commands.
23–8 TRAS_MAX	Defines the DRAM maximum row active time, in cycles.The user should set this value to the timing parameter in the memory specification. In practical use, the MC will use the value in this parameter with adjustments made for write recovery time, memory burst length and any internal delays if required. Note: Since the counter is initially loaded with the value tras_max-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
7–0 TMOD	Defines the number of cycles of wait time after a mode register write to any nonmode register write command.There are two parameters that control the timing after a mode register write: tmod and tmrd. This parameter (tmod) is typically the longer timing delay (defined by the memory specification) and controls the spacing of a mode register command to any other memory command. The MC will wait this delay after a mode register command before sending any other command to memory since the controller has no way of knowing whether or not termination (ODT) will be used. Note1: Since the counter is initially loaded with the value tmod-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum. Note2: For LPDDR2 memories, this parameter should be set to the tmrw value from the memory specification.

20.11.11 DRAM CTL Register 10 (DRAM_CTL10)

Address: DRAM_CTL10 is 1400_0000h base + 28h offset = 1400_0028h



DRAM_CTL10 field descriptions

Field	Description
31–25 -	Reserved.
24 CONCURRENTAP	Enables concurrent auto pre-charge. Some DRAM devices do not allow one bank to be auto pre-charged while another bank is reading or writing. The JEDEC standard allows concurrent auto precharge. The user should set this parameter if the DRAM device supports this feature. Note: This parameter can only be set if the device supports concurrent AP for all transactions. Some memories only support concurrent AP for writes. In this case, these memories require that reads to the same CS cannot be issued before the tWR delay has expired. If the memory devices being used do not support concurrent AP for all transactions, concurrentap must be set to b0, disabling concurrentap for all transfers. Note: It is highly recommended that user set CONCURRENTAP = 0 for this MC. b0 = Concurrent auto pre-charge disabled. b1 = Concurrent auto pre-charge enabled.
23–17 -	Reserved.
16 AP	Enables auto pre-charge mode for DRAM devices. Note: User must set AP = 0 for this MC.
15–13 -	Reserved.
12–8 TCKESR	Minimum CKE low pulse width during a self-refresh.

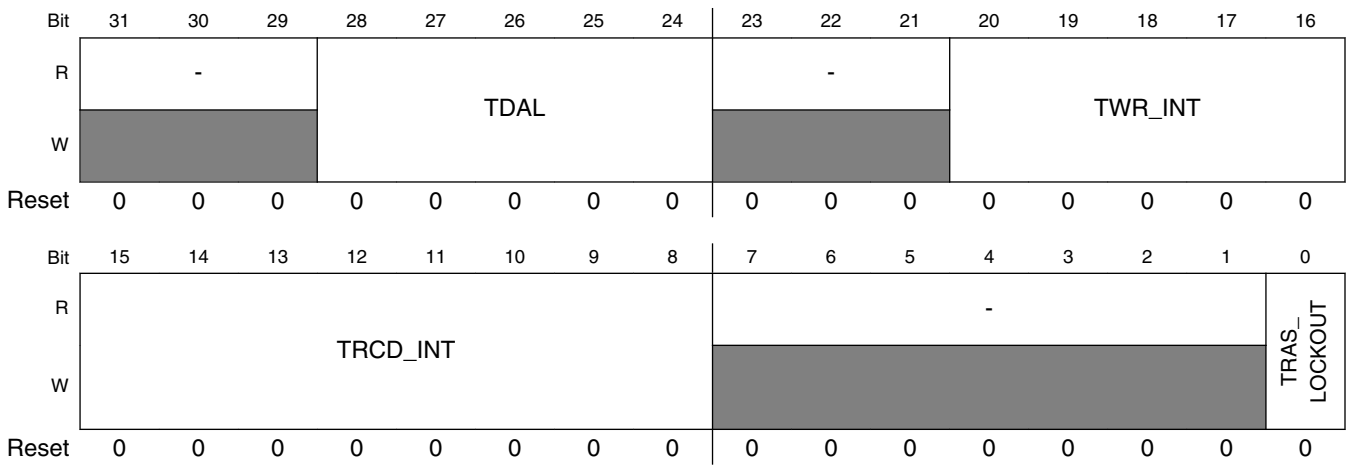
Table continues on the next page...

DRAM_CTL10 field descriptions (continued)

Field	Description
7–3 -	Reserved.
2–0 TCKE	Minimum CKE pulse width.

20.11.12 DRAM CTL Register 11 (DRAM_CTL11)

Address: DRAM_CTL11 is 1400_0000h base + 2Ch offset = 1400_002Ch



DRAM_CTL11 field descriptions

Field	Description
31–29 -	Reserved.
28–24 TDAL	DRAM TDAL parameter in cycles.
23–21 -	Reserved.
20–16 TWR_INT	Defines the DRAM write recovery time, in cycles. Note: Since the counter is initially loaded with the value twr_int-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
15–8 TRCD_INT	Defines the DRAM RAS to CAS delay, in cycles. Note: Since the counter is initially loaded with the value trcd_int-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
7–1 -	Reserved.

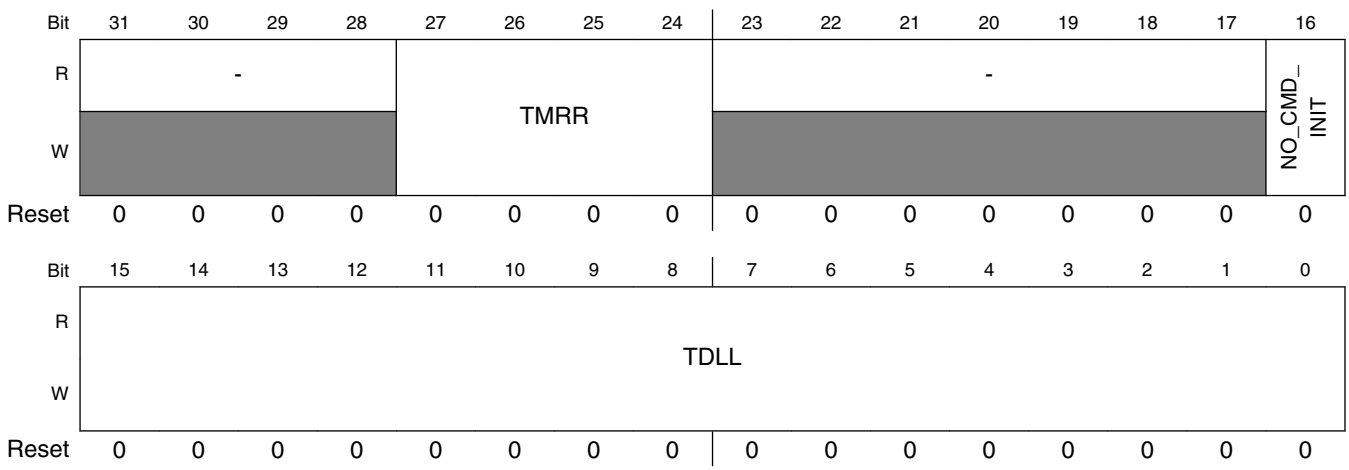
Table continues on the next page...

DRAM_CTL11 field descriptions (continued)

Field	Description
0 TRAS_ LOCKOUT	Defines the tRAS lockout setting for the DRAM device. tRAS lockout allows the memory controller to execute auto precharge commands before the tras_min parameter has expired. Note: This parameter may only be applicable to DDR1 memories. Please refer to the memory specification. 'b0 = tRAS lockout not supported by memory device. b1 = tRAS lockout supported by memory device.

20.11.13 DRAM CTL Register 12 (DRAM_CTL12)

Address: DRAM_CTL12 is 1400_0000h base + 30h offset = 1400_0030h



DRAM_CTL12 field descriptions

Field	Description
31–28 -	Reserved.
27–24 TMRR	Defines the DRAM tMRR term, in cycles. Note: Since the counter is initially loaded with the value tmrr-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
23–17 -	Reserved.
16 NO_CMD_INIT	Disables DRAM commands until DLL initialization is complete and tdll has expired. This parameter is only applicable when the MC is programmed for the following memory systems: DDR2 (dram_class = b0100) LPDDR1 (dram_class = b0001)

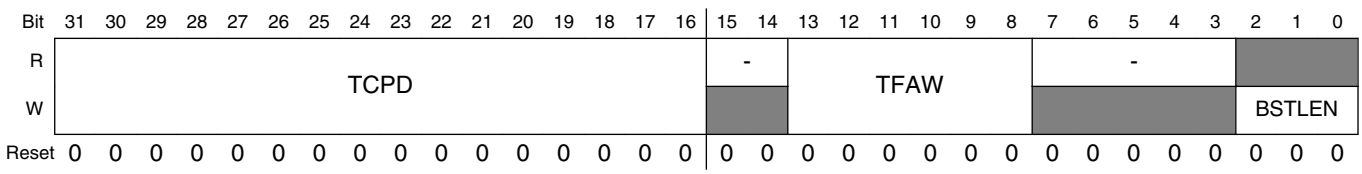
Table continues on the next page...

DRAM_CTL12 field descriptions (continued)

Field	Description
	b0 = Issue only REF and PRE commands during DLL initialization of the DRAM devices. If PRE commands are issued before DLL initialization is complete, the command will be executed immediately, and then the DLL initialization will continue. b1 = Do not issue any type of command during DLL initialization of the DRAM devices. If any other commands are issued during the initialization time, they will be held off until DLL initialization is complete.
15–0 TDLL	DRAM TDLL parameter in cycles.

20.11.14 DRAM CTL Register 13 (DRAM_CTL13)

Address: DRAM_CTL13 is 1400_0000h base + 34h offset = 1400_0034h

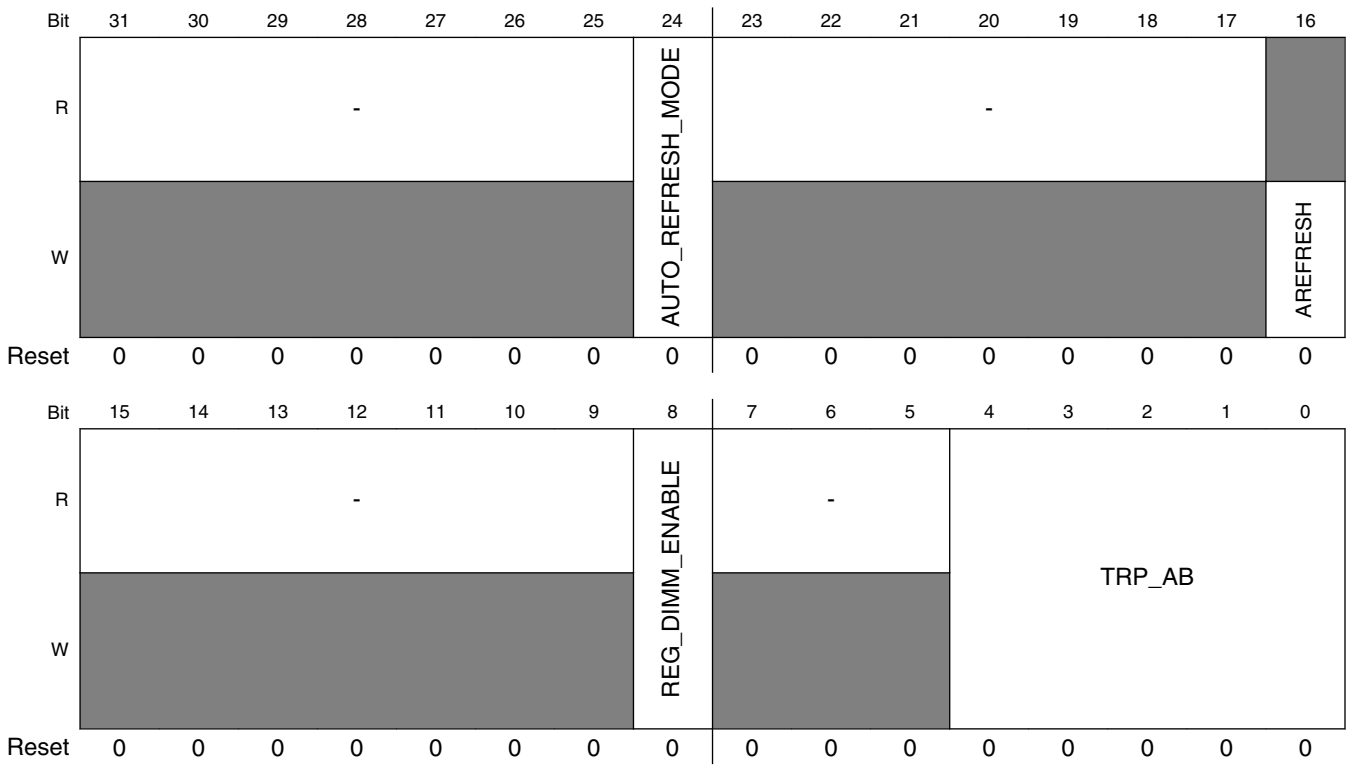


DRAM_CTL13 field descriptions

Field	Description
31–16 TCPD	DRAM TCPD parameter in cycles.
15–14 -	Reserved.
13–8 TFAW	DRAM TFAW parameter in cycles.
7–3 -	Reserved.
2–0 BSTLEN	Defines the memory burst length encoding for the MC. The mode is programmed in the dram_class parameter. b001 = 2 memory words - only applicable if the reduc parameter is set to b0 for operation in full datapath mode. Applicable for these memory systems: LPDDR1 (dram_class = b0001) b010 = 4 memory words. Applicable for these memory systems: LPDDR1 (dram_class = b0001) DDR2 (dram_class = b0100) LPDDR2 (dram_class = b0101) b011 = 8 memory words. Applicable for these memory systems: LPDDR1 (dram_class = b0001) LPDDR2 (dram_class = b0101) All other settings are Reserved

20.11.15 DRAM CTL Register 14 (DRAM_CTL14)

Address: DRAM_CTL14 is 1400_0000h base + 38h offset = 1400_0038h



DRAM_CTL14 field descriptions

Field	Description
31–25 -	Reserved.
24 AUTO_ REFRESH_ MODE	Sets the mode for when the automatic refresh will occur. If the auto_refresh_mode parameter is set and a refresh is required to memory, the memory controller will delay this refresh until the end of the current transaction (if the transaction is fully contained inside a single page), or until the current transaction hits the end of the current page. b0 = Issue refresh on the next memory burst boundary, even if the current command is not complete. b1 = Issue refresh on the next command boundary.
23–17 -	Reserved.
16 AREFRESH	Initiates an automatic refresh to the DRAM devices based on the setting of the auto_refresh_mode parameter. If there are any open banks when this parameter is set, the MC will automatically close these banks before issuing the auto-refresh command. This parameter will always read back as 0x0. For more information on the refresh per chip select feature, refer to the Other Databahn Features Chapter. b0 = No action b1 = Issue refresh to the DRAM devices

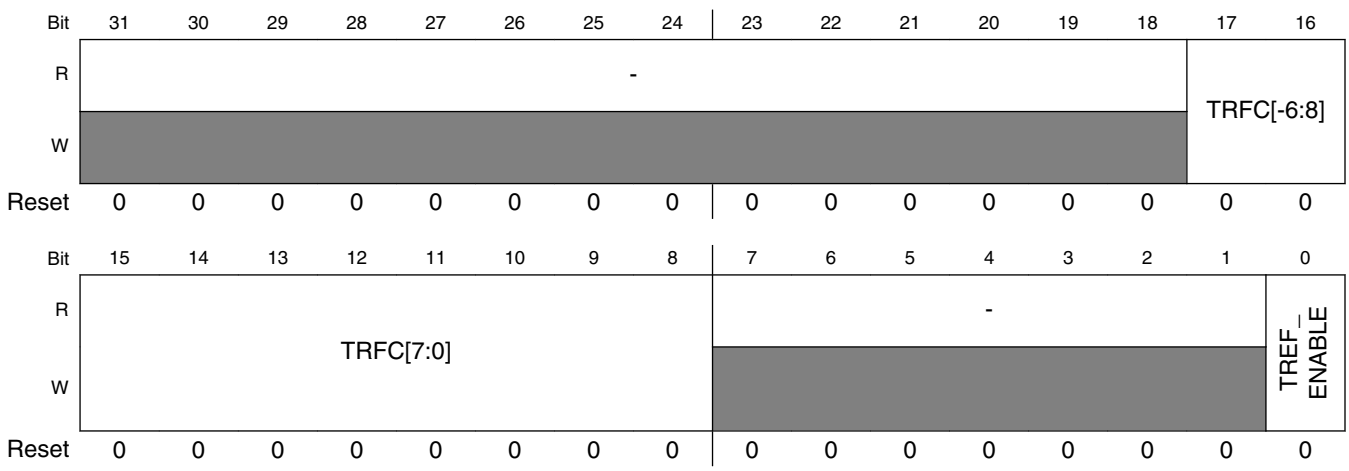
Table continues on the next page...

DRAM_CTL14 field descriptions (continued)

Field	Description
15–9 -	Reserved.
8 REG_DIMM_ENABLE	Enables registered DIMM operations to control the address and command pipeline of the memory controller. b0 = Normal operation b1 = Enable registered DIMM operation
7–5 -	Reserved.
4–0 TRP_AB	Defines the DRAM TRP time for all banks, in cycles. Note: Since the counter is initially loaded with the value trp_ab-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.

20.11.16 DRAM CTL Register 15 (DRAM_CTL15)

Address: DRAM_CTL15 is 1400_0000h base + 3Ch offset = 1400_003Ch



DRAM_CTL15 field descriptions

Field	Description
31–18 -	Reserved.
17–8 TRFC	Defines the DRAM refresh command time, in cycles. Note: Since the counter is initially loaded with the value trfc-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum. For more information on the refresh per chip select feature, refer to the Other Databahn Features Chapter.

Table continues on the next page...

DRAM_CTL15 field descriptions (continued)

Field	Description
7–1 -	Reserved.
0 TREF_ENABLE	Enables refresh commands. If command refresh mode is configured, then refresh commands will be automatically issued based on the tref parameter value and any refresh commands sent through the command interface or the register interface. Refreshes will still occur even if the DRAM devices have been placed in power-down state by the assertion of the power_down parameter. Note: Even with this parameter cleared to b0, some refresh commands may still be issued to memory. This only controls the automatic refreshes issued every tref cycles. b0 = Refresh commands disabled. b1 = Refresh commands enabled.

20.11.17 DRAM CTL Register 16 (DRAM_CTL16)

Address: DRAM_CTL16 is 1400_0000h base + 40h offset = 1400_0040h

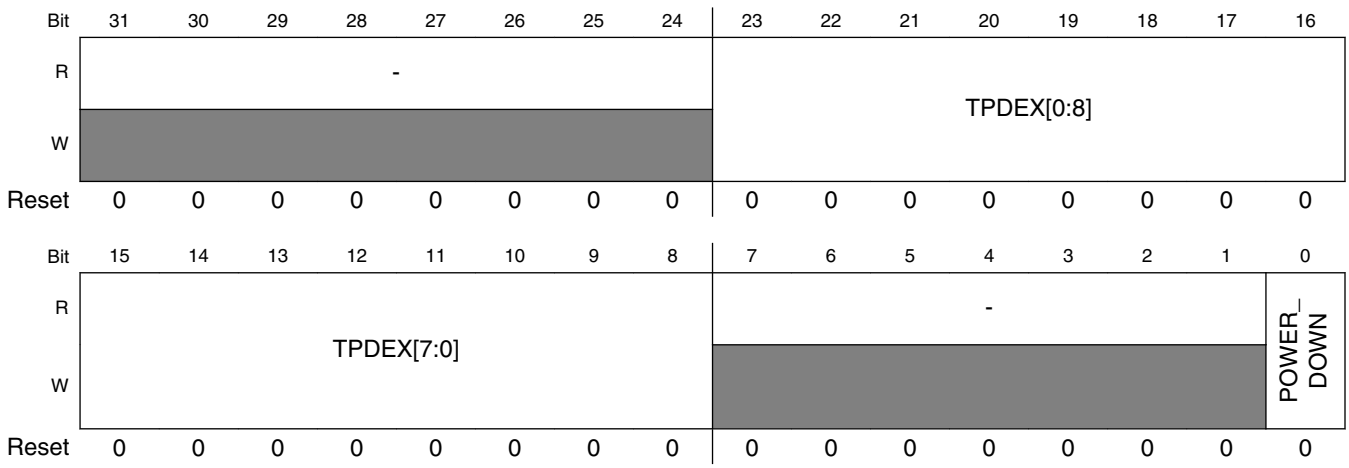
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-		TREF_INTERVAL														TREF															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL16 field descriptions

Field	Description
31–30 -	Reserved.
29–16 TREF_ INTERVAL	Sets the time delay, in clock cycles, between when refresh commands are issued to different chip selects. If tref_interval is cleared to b000, then the refresh per chip select logic will be disabled and all chip selects will be refreshed simultaneously. The tref_interval parameter value must be less than the trfc parameter value. For more information on the refresh per chip select feature, refer to the Other Databahn Features Chapter.
15–0 TREF	Defines the DRAM cycles between refresh commands. This parameter sets the average interval between refreshes. For more information on this parameter, refer to Section 13.3, Refresh Per Command Timing. For more information on the refresh per chip select feature, refer to the Section 13.5, Refresh Per Chip Select. Note: Since the counter is initially loaded with the value tref-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.

20.11.18 DRAM CTL Register 17 (DRAM_CTL17)

Address: DRAM_CTL17 is 1400_0000h base + 44h offset = 1400_0044h



DRAM_CTL17 field descriptions

Field	Description
31–24 -	Reserved.
23–8 TPDEX	Defines the DRAM power-down exit command period, in cycles. Note: Since the counter is initially loaded with the value tpdex-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
7–1 -	Reserved.
0 POWER_DOWN	When this parameter is set to b1, the memory controller will complete processing of the current memory burst for the current transaction (if any), issue a pre-charge all command and then disable the clock enable signal to the DRAM devices. Any subsequent commands in the command queue will be suspended until this parameter is cleared to b0. Note: This parameter should only be used when the low power logic is NOT enabled (lowpower_control = 0x00). Note: If the tref_enable parameter is set to b1, the memory controller will continue to issue refresh commands to the memory devices even when this parameter is set to b1. b0 = Enable full power state b1 = Disable the clock enable and power down the memory controller.

20.11.19 DRAM CTL Register 18 (DRAM_CTL18)

Address: DRAM_CTL18 is 1400_0000h base + 48h offset = 1400_0048h

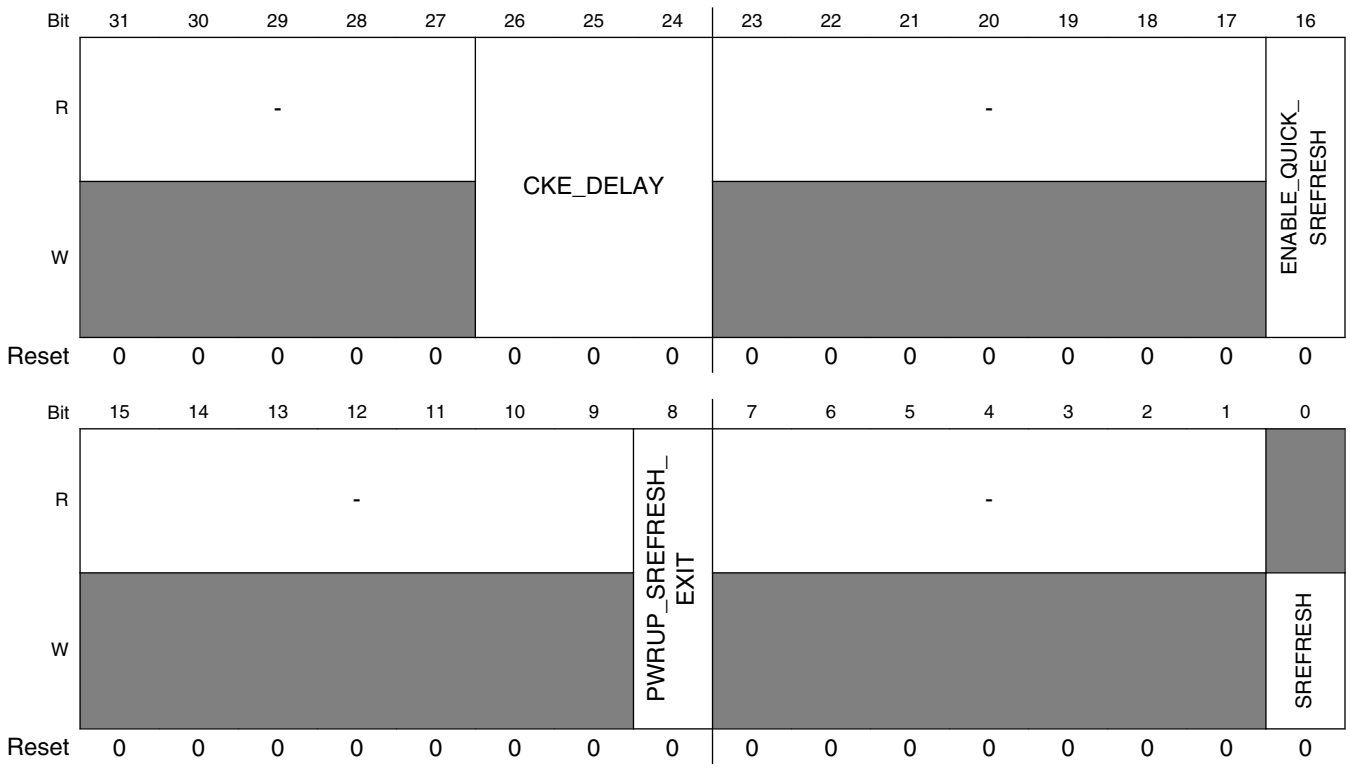
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TXSNR																TXSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL18 field descriptions

Field	Description
31–16 TXSNR	Defines the DRAM time from a selfrefresh exit to a command that does not require the memory DLL to be locked. Note: Since the counter is initially loaded with the value txsnr-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.
15–0 TXSR	Defines the DRAM time from a selfrefresh exit to an active command that requires the memory DLL to be locked. For DDR2 memories: Set this parameter to the difference between the txsrd and trcd values in the memory specification. Note: Since the counter is initially loaded with the value txsr-1, if this parameter is cleared to b0, the counter will underflow on load. This will result in setting the counter to its maximum value. Therefore, the user should set this parameter to b1 as a minimum.

20.11.20 DRAM CTL Register 19 (DRAM_CTL19)

Address: DRAM_CTL19 is 1400_0000h base + 4Ch offset = 1400_004Ch



DRAM_CTL19 field descriptions

Field	Description
31–27 -	Reserved.
26–24 CKE_DELAY	Sets the number of additional cycles of delay to include in the CKE signal cke_status for status reporting. The default delay is 0 cycles.
23–17 -	Reserved.
16 ENABLE_ QUICK_ SREFRESH	When this bit is set to b1, the memory initialization sequence may be interrupted and the memory may enter self-refresh mode. This is used to place the memory devices into self-refresh mode when a power loss is detected during the initialization process. Setting this parameter does not initiate a self-refresh entry during initialization. Rather, if the srefresh parameter is set to b1 during initialization, the self-refresh entry will only occur if this parameter is also set to b1. b0 = Continue memory initialization. b1 = Interrupt memory initialization and enter self-refresh mode.
15–9 -	Reserved.

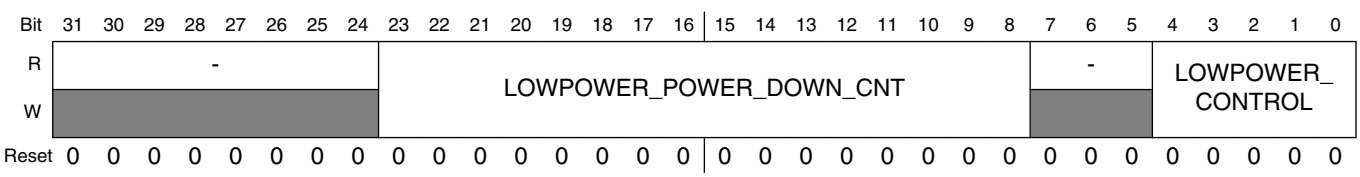
Table continues on the next page...

DRAM_CTL19 field descriptions (continued)

Field	Description
8 PWRUP_ SREFRESH_ EXIT	Allows controller to exit power-down mode by executing a self-refresh exit instead of the full memory initialization. This parameter provides a means to skip full initialization when the DRAM devices are in a known self-refresh state. b0 = Disabled b1 = Enabled
7–1 -	Reserved.
0 SREFRESH	When this parameter is set to b1, the DRAM device(s) will be placed in selfrefresh mode. To do so, the current memory burst for the current transaction (if any) will complete, all banks will be closed, the self-refresh command will be issued to the DRAM, and the clock enable signal will be de-asserted. The system will remain in self-refresh mode until this parameter is cleared to b0. The DRAM devices will return to normal operating mode after the self-refresh exit time (the txsr parameter) of the device and any DLL initialization time for the DRAM is reached. The memory controller will resume processing of the commands from the interruption point. Note1: This parameter should only be used when the low power logic is NOT enabled (lowpower_control = 0x00). Note2: If this parameter is asserted during the initialization sequence, the memory controller will only issue the self-refresh if such behavior is enabled in the enable_quick_srefresh parameter. This parameter will be updated with an assertion of the srefresh_enter signal, regardless of the behavior on the register interface. To disable self-refresh again after a srefresh_enter signal assertion, the user will need to clear the parameter to b0. b0 = Disable self-refresh mode b1 = Initiate self-refresh of the DRAM devices.

20.11.21 DRAM CTL Register 20 (DRAM_CTL20)

Address: DRAM_CTL20 is 1400_0000h base + 50h offset = 1400_0050h



DRAM_CTL20 field descriptions

Field	Description
31–24 -	Reserved.
23–8 LOWPOWER_ POWER_ DOWN_CNT	Counts the number of idle cycles before memory power-down or power-down with memory clock gating low power mode. This parameter must be programmed to a non-zero value for proper operation.

Table continues on the next page...

DRAM_CTL20 field descriptions (continued)

Field	Description
7–5 -	Reserved.
4–0 LOWPOWER_ CONTROL	Enables the individual low power modes of the device. Bit [4] = Controls memory power-down mode (Mode 1). Bit [3] = Controls memory power-down with memory clock gating mode (Mode 2). Bit [2] = Controls memory self-refresh mode (Mode 3). Bit [1] = Controls memory self-refresh with memory clock gating mode (Mode 4). Bit [0] = Controls memory self-refresh with memory and controller clock gating mode (Mode 5). Note: Unless this parameter is cleared to 0x00, the srefresh and power_down parameters should NOT be used. For all bits: b0 = Disabled b1 = Enabled

20.11.22 DRAM CTL Register 21 (DRAM_CTL21)

Address: DRAM_CTL21 is 1400_0000h base + 54h offset = 1400_0054h

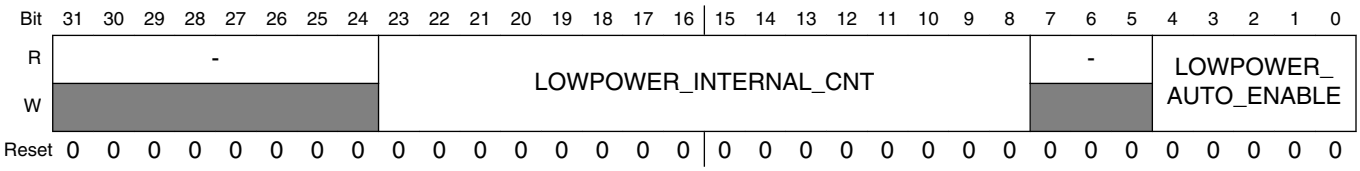
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOWPOWER_EXTERNAL_CNT																LOWPOWER_SELF_REFRESH_CNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL21 field descriptions

Field	Description
31–16 LOWPOWER_ EXTERNAL_ CNT	Counts the number of idle cycles before memory self-refresh with memory clock gating low power mode. This parameter must be programmed to a non-zero value for proper operation.
15–0 LOWPOWER_ SELF_ REFRESH_CNT	Counts the number of cycles to the next memory self-refresh low power mode. This parameter must be programmed to a non-zero value for proper operation.

20.11.23 DRAM CTL Register 22 (DRAM_CTL22)

Address: DRAM_CTL22 is 1400_0000h base + 58h offset = 1400_0058h



DRAM_CTL22 field descriptions

Field	Description
31–24 -	Reserved.
23–8 LOWPOWER_ INTERNAL_CNT	Counts the number of idle cycles before memory self-refresh with memory and controller clock gating low power mode. This parameter must be programmed to a non-zero value for proper operation.
7–5 -	Reserved.
4–0 LOWPOWER_ AUTO_ENABLE	Enables automatic entry into the low power modes of the memory controller. Bit [4] = Controls memory power-down mode (Mode 1). Bit [3] = Controls memory power-down with memory clock gating mode (Mode 2). Bit [2] = Controls memory self-refresh mode (Mode 3). Bit [1] = Controls memory self-refresh with memory clock gating mode (Mode 4). Bit [0] = Controls memory self-refresh with memory and controller clock gating mode (Mode 5). Note1: It is not possible to enter Mode 5 manually. Setting bit [0] of lowpower_control with bit [0] of this parameter cleared will not result in any change. Note2: The user should not use both automatic and manual modes for the various low power modes. All modes should be entered automatically or all entered manually. For all bits: b0 = Automatic entry into this mode is disabled. The user may enter this mode manually by setting the associated lowpower_control bit. b1 = Automatic entry into this mode is enabled. The mode will be entered automatically when the proper counters expire, and only if the associated lowpower_control bit is set.

20.11.24 DRAM CTL Register 23 (DRAM_CTL23)

Address: DRAM_CTL23 is 1400_0000h base + 5Ch offset = 1400_005Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-				CKSRE				-						LOWPOWER_REFRESH_ENABLE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

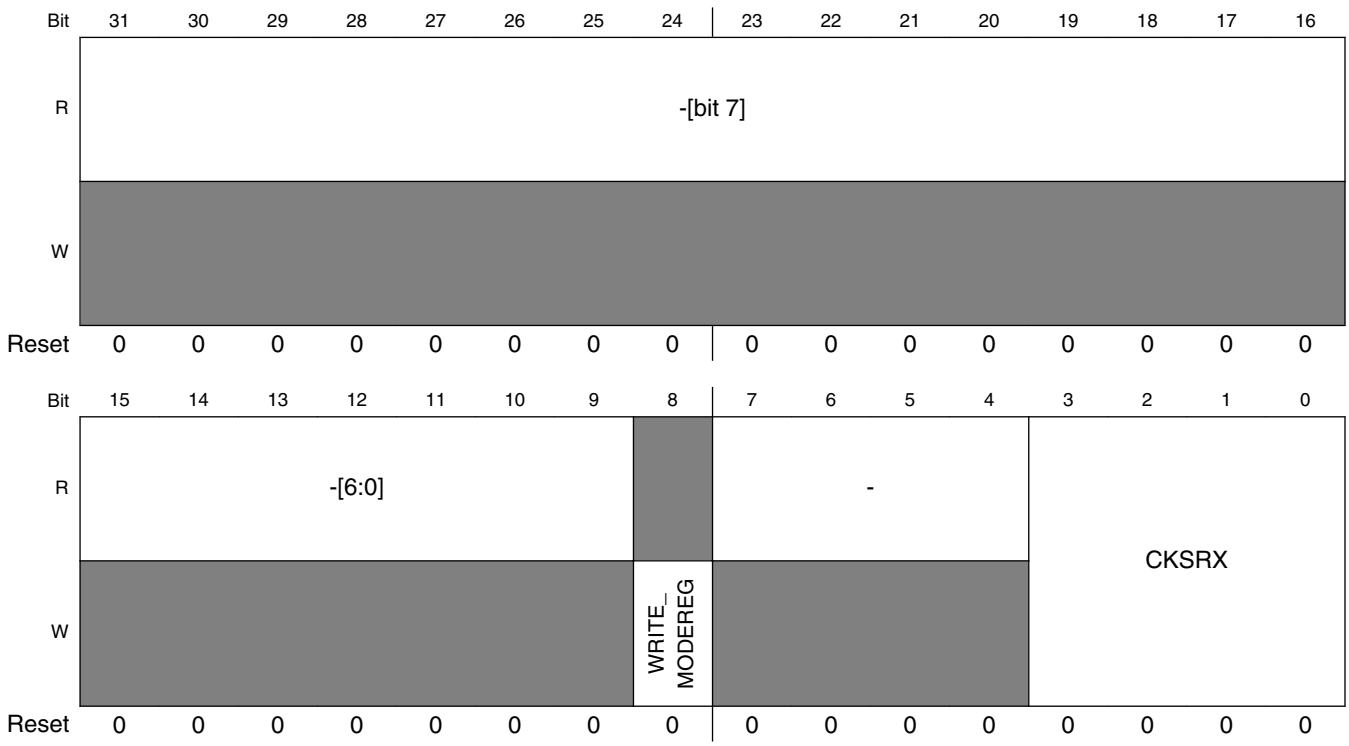
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOWPOWER_REFRESH_HOLD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL23 field descriptions

Field	Description
31–28 -	Reserved.
27–24 CKSRE	Sets the number of cycles to hold the clock stable after entering self-refresh mode. The clock will run for a minimum of cksre cycles after CKE falls.
23–18 -	Reserved.
17–16 LOWPOWER_REFRESH_ENABLE	Sets whether refreshes will occur while the memory controller is in one of the power-down modes. There is one bit for each chip select. Note that refreshes will not occur while in any of the self-refresh modes. Note: This parameter is active low. b0 = Refreshes still occur b1 = Refreshes do not occur
15–0 LOWPOWER_REFRESH_HOLD	Sets the number of cycles that the MC will wait before attempting to re-lock the DLL when using the Memory Self-Refresh with Memory and Controller Clock Gating low power mode. This counter will only be used in this mode, which is the deepest low power mode. When this counter expires, the DLL will be un-gated for at least 16 cycles during which the DLL will attempt to re-lock. After 16 cycles have elapsed and the DLL has locked, then the DLL controller clock will be gated again and the counter will reset to this value. If the DLL requires more than 16 cycles to re-lock, then the un-gated time will be longer.

20.11.25 DRAM CTL Register 24 (DRAM_CTL24)

Address: DRAM_CTL24 is 1400_0000h base + 60h offset = 1400_0060h

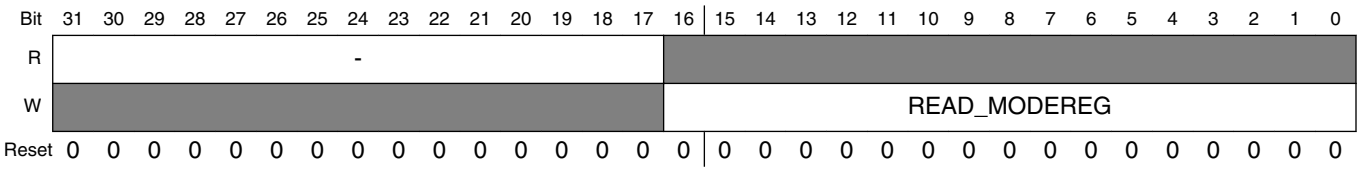


DRAM_CTL24 field descriptions

Field	Description
31–9 -	Reserved.
8 WRITE_ MODEREG	Writes mode register information into the memory devices. The user should program the appropriate mrY_data_X parameters with valid information based on the memory type being used. All of the mode registers that are relevant for the memory type specified in the dram_class parameter will be written on each write_modereg setting. This parameter will always read back as b0. The mode registers are automatically written at initialization of the memory controller. There is no need to initiate a mode register write after setting the start parameter in the memory controller unless some value in these registers needs to be changed after initialization. Note: This parameter may not be changed when the memory is in power-down mode (when the CKE input is deasserted).
7–4 -	Reserved.
3–0 CKSRX	Sets the number of cycles to hold the clock stable before exiting self-refresh mode. The clock will run for a minimum of cksrx cycles before CKE rises.

20.11.26 DRAM CTL Register 25 (DRAM_CTL25)

Address: DRAM_CTL25 is 1400_0000h base + 64h offset = 1400_0064h

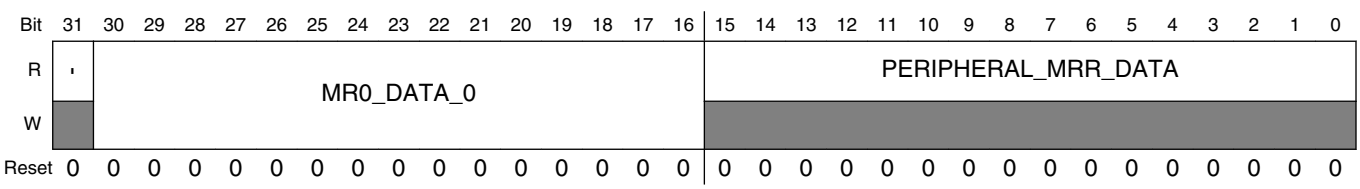


DRAM_CTL25 field descriptions

Field	Description
31–17 -	Reserved.
16–0 READ_ MODEREG	Read from the mode register MRx. Bits [7:0] = Defines the mode register number being requested (MRx). Bits [15:8] = Defines the chip select for which to read the command. Bit [16] = Trigger the mode register read. b0 = No action b1 = Read the mode register specified

20.11.27 DRAM CTL Register 26 (DRAM_CTL26)

Address: DRAM_CTL26 is 1400_0000h base + 68h offset = 1400_0068h



DRAM_CTL26 field descriptions

Field	Description
31 -	Reserved.
30–16 MR0_DATA_0	Holds the memory mode register 0 data for chip select 0 written during memory initialization or when the write_modereg parameter is asserted. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this MC: For DDR2 memories: (MR) The memory controller does not support interleaving and therefore the A3 bit should be cleared to b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. For a memory burst length of 4, the bits A2:A0 should be set to b010.

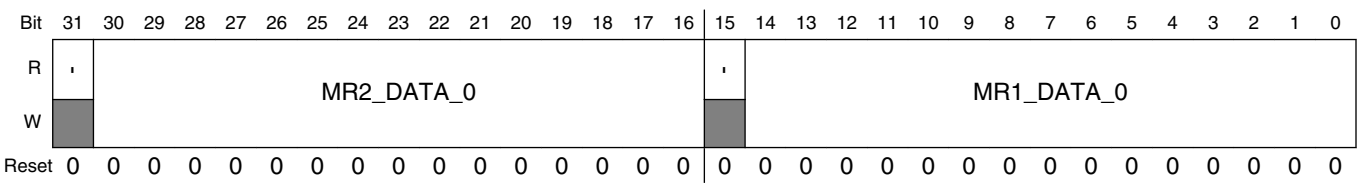
Table continues on the next page...

DRAM_CTL26 field descriptions (continued)

Field	Description
	For LPDDR1 memories: (MR) The memory controller does not support interleaving and therefore the A3 bit should be cleared to b0. For a memory burst length of 4, the bits A2:A0 should be set to b010. For LPDDR2 memories: (MR0) For this memory type, the fields of MR0 are all read-only, therefore this parameter has no meaning during initialization, or when the write_modereg parameter is set to b1. To gather the contents of this memory register, the user must set the read_modereg parameter appropriately and then read the peripheral_mrr_data parameter.
15–0 PERIPHERAL_MRR_DATA	Data and chip returned from the mode register read issued as a result of setting the parameter read_modereg to b1. The contents of this parameter are only valid when the peripheral mode read complete bit (bit 6) in the int_status parameter is asserted. Bits [15:8] = MRR chip information Bits [7:0] = Mode register data

20.11.28 DRAM CTL Register 27 (DRAM_CTL27)

Address: DRAM_CTL27 is 1400_0000h base + 6Ch offset = 1400_006Ch



DRAM_CTL27 field descriptions

Field	Description
31 -	Reserved.
30–16 MR2_DATA_0	Holds the memory mode register 2 data for chip select 0 written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this MC: For DDR2 memories: This parameter correlates to the extended memory mode register 2 (EMR2). For LPDDR1 memories: This parameter correlates to the extended mode register (EMR). For LPDDR2 memories: This parameter correlates to memory mode register 2 (MR2). This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is asserted.
15 -	Reserved.
14–0 MR1_DATA_0	Holds the memory mode register 1 data for chip select 0 written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this MC: For DDR2 memories: This parameter correlates to the extended memory mode register 1 (EMR1). The memory controller does not support additive latency and therefore the A5:A3 bits should be cleared to b000.

Table continues on the next page...

DRAM_CTL27 field descriptions (continued)

Field	Description
	For LPDDR1 memories: This parameter has no meaning for this memory type. For LPDDR2 memories: This parameter correlates to memory mode register 1 (MR1). The memory controller does not support interleaving and therefore the OP3 bit should be cleared to b0. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is asserted.

20.11.29 DRAM CTL Register 28 (DRAM_CTL28)

Address: DRAM_CTL28 is 1400_0000h base + 70h offset = 1400_0070h

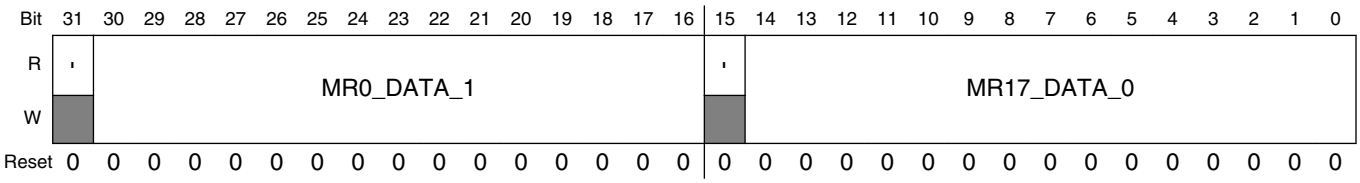
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	'	MR16_DATA_0															'	MR3_DATA_0														
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL28 field descriptions

Field	Description
31 -	Reserved.
30–16 MR16_DATA_0	Holds the memory mode register 16 data for chip select 0 written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this MC: For DDR2 memories: This parameter has no meaning for this memory type. For LPDDR1 memories: This parameter has no meaning for this memory type. For LPDDR2 memories: This parameter correlates to the memory mode register 16 (MR16). The fields of MR16 will be programmed into the DRAM at initialization or when the write_modereg parameter is asserted.
15 -	Reserved.
14–0 MR3_DATA_0	Holds the memory mode register 3 data for chip select 0 written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this MC: For DDR2 memories: This parameter correlates to the extended memory mode register 3 (EMR3). For LPDDR1 memories: This parameter has no meaning for this memory type. For LPDDR2 memories: This parameter correlates to memory mode register 3 (MR3). This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is asserted.

20.11.30 DRAM CTL Register 29 (DRAM_CTL29)

Address: DRAM_CTL29 is 1400_0000h base + 74h offset = 1400_0074h

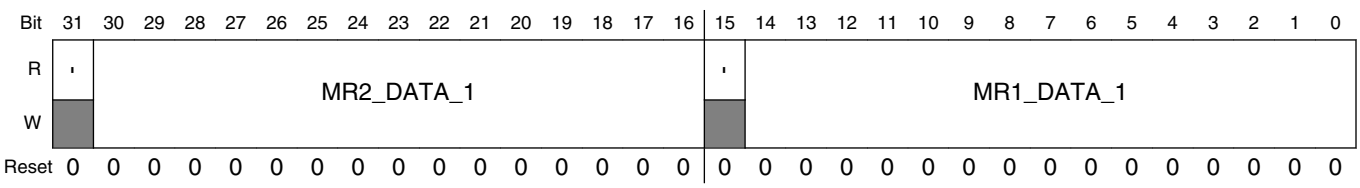


DRAM_CTL29 field descriptions

Field	Description
31 -	Reserved.
30–16 MR0_DATA_1	MRS data to program to memory mode register 0 for chip select 1. Refer to MR0_DATA_0 for detailed description.
15 -	Reserved.
14–0 MR17_DATA_0	Holds the memory mode register 17 data for chip select 0 written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this MC: For DDR2 memories: This parameter has no meaning for this memory type. For LPDDR1 memories: This parameter has no meaning for this memory type. For LPDDR2 memories: This parameter correlates to the memory mode register 17 (MR17) for LPDDR2-S4 memory devices. If the lpddr2_s4 parameter is set to b1, the fields of MR17 will be programmed into the DRAM at initialization or when the write_modereg parameter is asserted.

20.11.31 DRAM CTL Register 30 (DRAM_CTL30)

Address: DRAM_CTL30 is 1400_0000h base + 78h offset = 1400_0078h



DRAM_CTL30 field descriptions

Field	Description
31 -	Reserved.

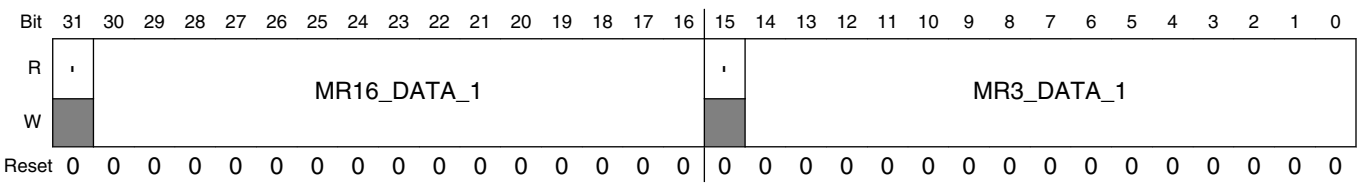
Table continues on the next page...

DRAM_CTL30 field descriptions (continued)

Field	Description
30–16 MR2_DATA_1	Data to program into memory mode register 2 for chip select 1. Refer to MR2_DATA_0 for detailed description.
15 -	Reserved.
14–0 MR1_DATA_1	Data to program into memory mode register 1 for chip select 1. Refer to MR1_DATA_0 for detailed description.

20.11.32 DRAM CTL Register 31 (DRAM_CTL31)

Address: DRAM_CTL31 is 1400_0000h base + 7Ch offset = 1400_007Ch

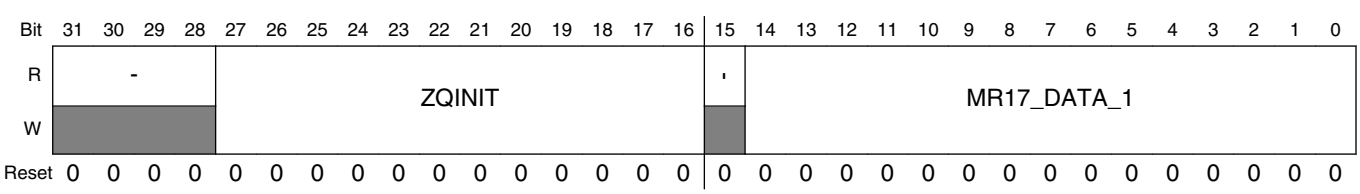


DRAM_CTL31 field descriptions

Field	Description
31 -	Reserved.
30–16 MR16_DATA_1	Data to program into memory mode register 16 for chip select 1. Refer to MR16_DATA_0 for detailed description.
15 -	Reserved.
14–0 MR3_DATA_1	Data to program into memory mode register 3 for chip select 1. Refer to MR3_DATA_0 for detailed description.

20.11.33 DRAM CTL Register 32 (DRAM_CTL32)

Address: DRAM_CTL32 is 1400_0000h base + 80h offset = 1400_0080h

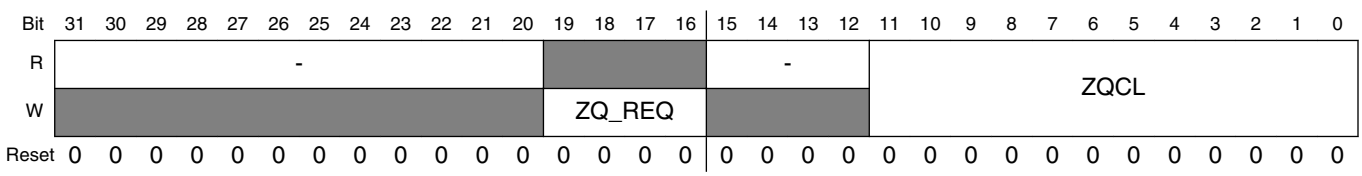


DRAM_CTL32 field descriptions

Field	Description
31–28 -	Reserved.
27–16 ZQINIT	Specifies the duration of wait time, in cycles, required for the memory devices to complete a ZQ command during initialization. Note: This parameter should be set to the ZQCL time. Internally, the MC will set the time required for a ZQCL to 1/2 of this setting. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101)
15 -	Reserved.
14–0 MR17_DATA_1	Data to program into memory mode register 17 for chip select 1. Refer to MR17_DATA_0 for detailed description.

20.11.34 DRAM CTL Register 33 (DRAM_CTL33)

Address: DRAM_CTL33 is 1400_0000h base + 84h offset = 1400_0084h



DRAM_CTL33 field descriptions

Field	Description
31–20 -	Reserved.
19–16 ZQ_REQ	Triggers a user-requested ZQ operation. This parameter is write-only and any writes will be cleared once the process is initiated. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101) Bit [3] = Triggers a ZQ reset to all chip selects in the system. Bit [2] = Triggers a ZQ initialization to all chip selects in the system. Bit [1] = Triggers a long ZQ calibration to all chip selects in the system. Bit [0] = Triggers a short ZQ calibration. If the zqcs_rotate parameter is set to b1, only one chip select will be calibrated (in round-robin fashion). If the zqcs_rotate parameter is cleared to b0, all chip selects defined in the parameter cs_map will be calibrated.
15–12 -	Reserved.

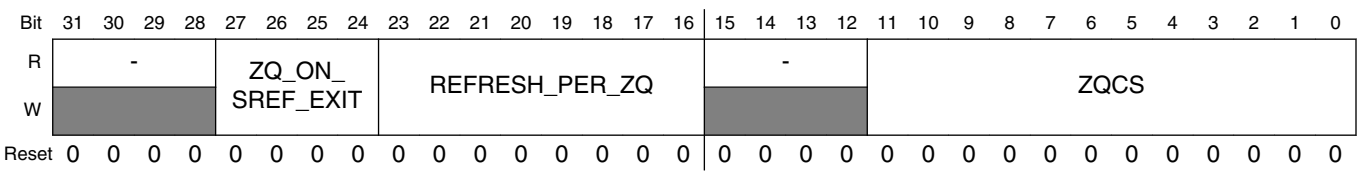
Table continues on the next page...

DRAM_CTL33 field descriptions (continued)

Field	Description
11–0 ZQCL	Specifies the duration of wait time, in cycles, required for the memory devices to complete a long ZQ calibration command (ZQCL). This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101) Note: This parameter must be set to 1/2 of the time set in the zqinit parameter.

20.11.35 DRAM CTL Register 34 (DRAM_CTL34)

Address: DRAM_CTL34 is 1400_0000h base + 88h offset = 1400_0088h



DRAM_CTL34 field descriptions

Field	Description
31–28 -	Reserved.
27–24 ZQ_ON_SREF_ EXIT	Issues a ZQ command when exiting selfrefresh mode. This is a one-hot parameter which determines the type of operation performed on self-refresh exit. Note: It is recommended to set this parameter to execute a ZQCL on selfrefresh exit. By setting this parameter this way, a ZQCL will be issued before any other commands are permitted to execute. If this parameter is set to any other setting, the system must manage self-refresh exit and ZQ calibration. If using the ZQCS option, It advises that the rotate function should be disabled (the parameter zqcs_rotate should be cleared to b0) to calibrate all chip selects. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101) Bit [3] = Triggers a ZQ reset to all chip selects in the system on self-refresh exit. Bit [2] = Triggers a ZQ initialization to all chip selects in the system on selfrefresh exit. Bit [1] = Triggers a long ZQ calibration to all chip selects in the system on selfrefresh exit. Bit [0] = Triggers a short ZQ calibration. If the zqcs_rotate parameter is set to b1, only one chip select will be calibrated (in round-robin fashion). If the zqcs_rotate parameter is cleared to b0, all chip selects defined in the parameter cs_map will be calibrated.
23–16 REFRESH_ PER_ZQ	Sets the maximum number of refreshes allowed between automatic ZQCS commands. Setting this parameter to 0x0 will disable automatic ZQCS commands. When using the rotate option (the parameter zqcs_rotate is set to b1, this parameter should be programmed to the total number of cycles desired between ZQCS commands divided by the number of chip selects in the system. This parameter is only applicable when the MC is programmed for the following memory system:

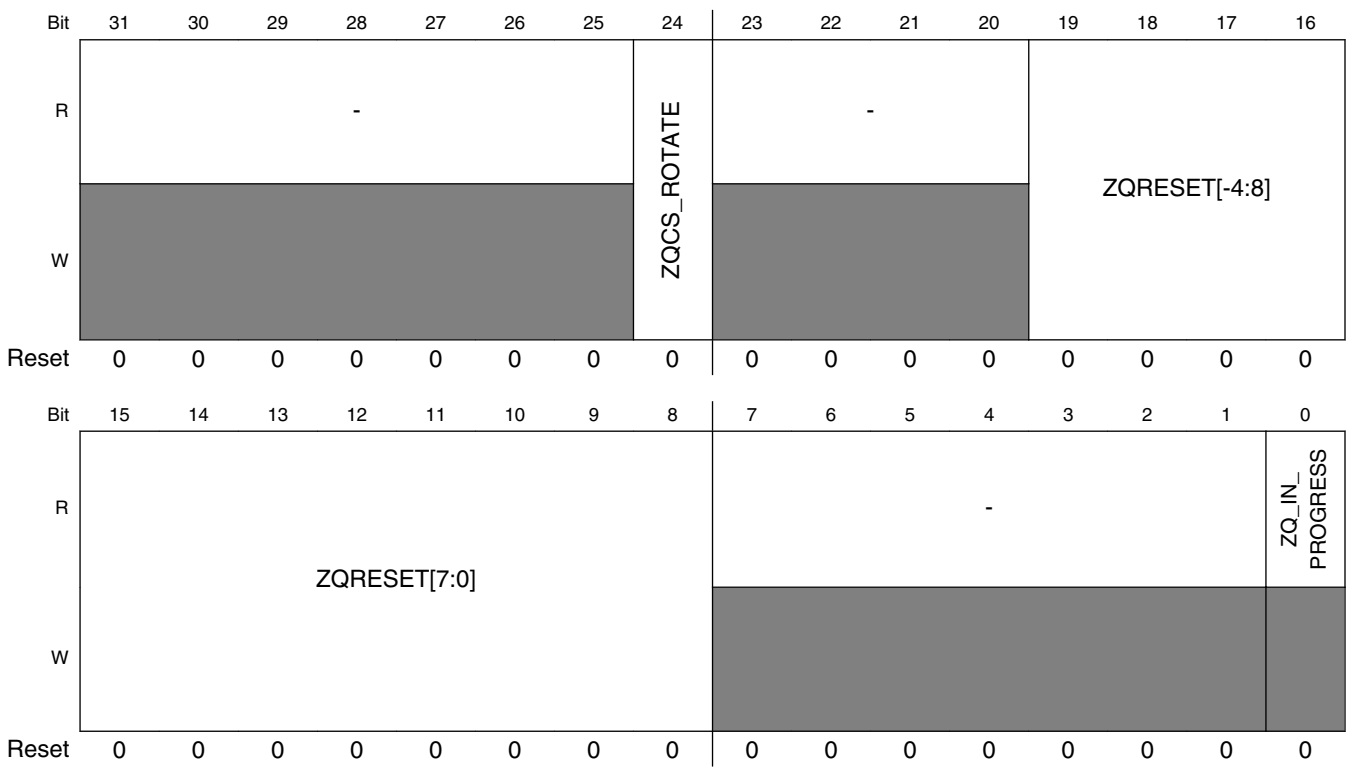
Table continues on the next page...

DRAM_CTL34 field descriptions (continued)

Field	Description
	LPDDR2 (dram_class = b0101)
15–12 -	Reserved.
11–0 ZQCS	Specifies the duration of wait time, in cycles, required for the memory devices to complete a short ZQ calibration command (ZQCS). If the zqcs_rotate parameter will be used, the user should program this parameter accordingly. If each memory device must receive a ZQCS command every Y cycles, and there are a total of X chip selects in the system, then the zqcs parameter should be programmed to Y/X. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101)

20.11.36 DRAM CTL Register 35 (DRAM_CTL35)

Address: DRAM_CTL35 is 1400_0000h base + 8Ch offset = 1400_008Ch



DRAM_CTL35 field descriptions

Field	Description
31–25 -	Reserved.

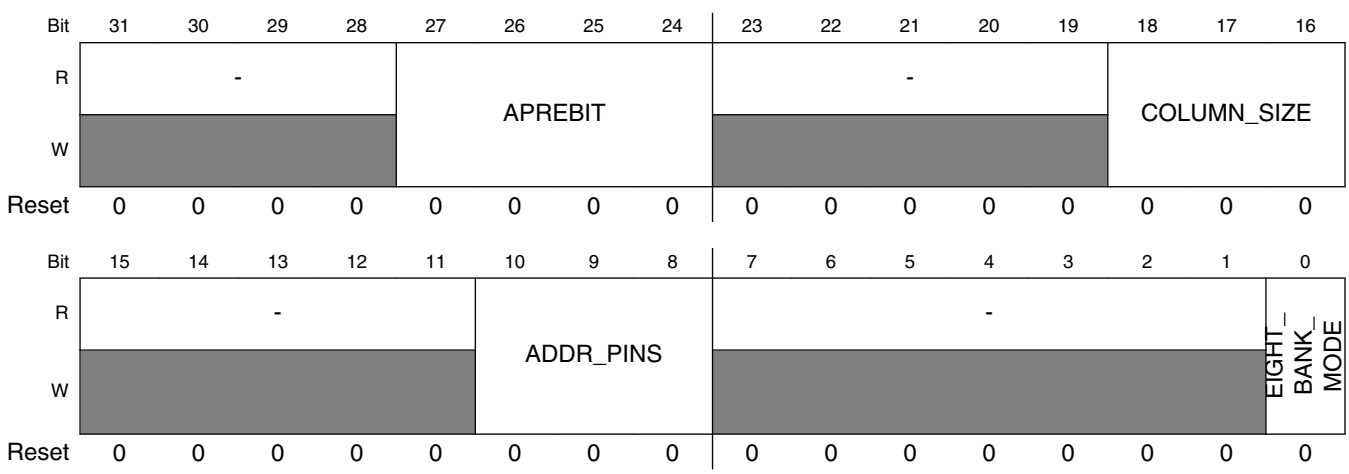
Table continues on the next page...

DRAM_CTL35 field descriptions (continued)

Field	Description
24 ZQCS_ROTATE	Defines the behavior of short ZQ calibrations. If this parameter is set to b1, a ZQCS request will only be issued to one chip select on each request, regardless of the source of the request. The MC will maintain a counter such that the chip selects are calibrated in a fixed rotating sequence. This option can be used to minimize the overhead of periodic ZQ calibrations. If the total time to calibrate all chip selects is acceptable offline time, the user may disable the zqcs_rotate parameter. Note: If this parameter is set to b1, the user should program the refresh_per_zq parameter to the total time allowed between ZQCS commands divided by the number of chip selects. If this parameter is cleared to b0, the user should program the refresh_per_zq parameter to the time between ZQCS commands for all chip selects. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101) b0 = Calibrate all chip selects on each ZQCS request. b1 = Calibrate only one chip select on the next ZQCS request.
23–20 -	Reserved.
19–8 ZQRESET	Specifies the duration of wait time, in cycles, required for the memory devices to complete a ZQRESET command. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101)
7–1 -	Reserved.
0 ZQ_IN_PROGRESS	Indicates that a ZQ command is currently in progress. If a ZQ command is requested while this parameter is set to b1, the new ZQ request will be ignored. This parameter is only applicable when the MC is programmed for the following memory system: LPDDR2 (dram_class = b0101)

20.11.37 DRAM CTL Register 36 (DRAM_CTL36)

Address: DRAM_CTL36 is 1400_0000h base + 90h offset = 1400_0090h

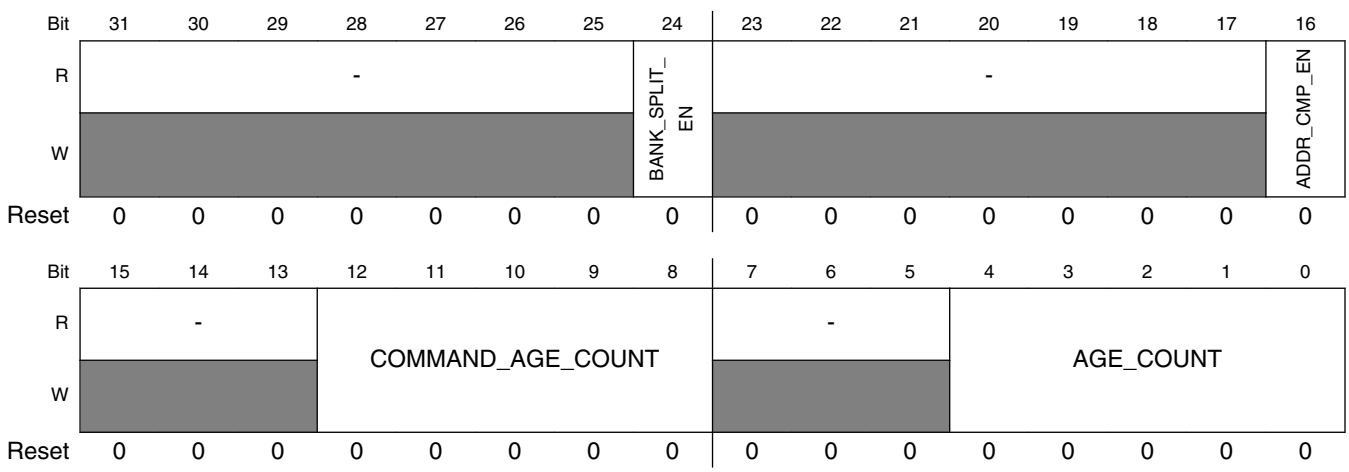


DRAM_CTL36 field descriptions

Field	Description
31–28 -	Reserved.
27–24 APREBIT	Defines the location of the auto precharge bit in the DRAM address in decimal encoding.
23–19 -	Reserved.
18–16 COLUMN_SIZE	Shows the difference between the maximum column width available (10) and the actual number of column pins being used. The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this parameter. For details, refer to the Databahn Address Mapping Chapter.
15–11 -	Reserved.
10–8 ADDR_PINS	Defines the difference between the maximum number of address pins configured (15) and the actual number of pins being used. Note: For LPDDR2 memories, this parameter holds the difference between the maximum number of row pins configured and the actual number of row pins being used for chip select X. The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this parameter. For details, refer to the Databahn Address Mapping Chapter.
7–1 -	Reserved.
0 EIGHT_BANK_MODE	Indicates that the memory devices have eight banks. b0 = Memory devices have 4 banks. b1 = Memory devices have 8 banks.

20.11.38 DRAM CTL Register 37 (DRAM_CTL37)

Address: DRAM_CTL37 is 1400_0000h base + 94h offset = 1400_0094h

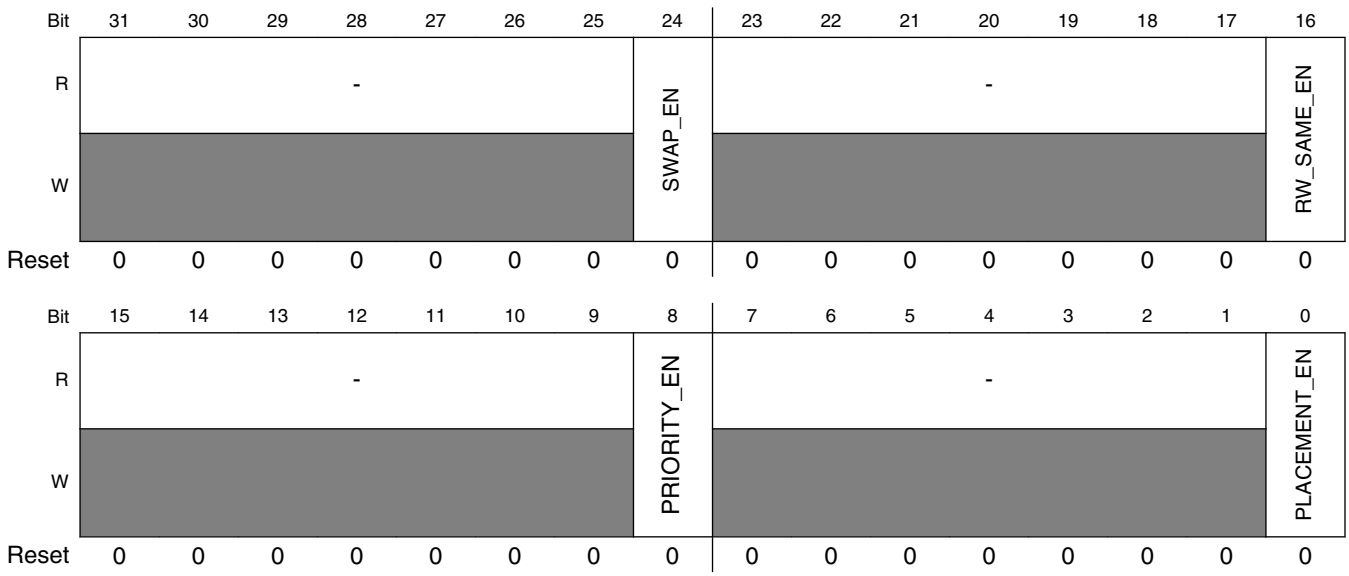


DRAM_CTL37 field descriptions

Field	Description
31–25 -	Reserved.
24 BANK_SPLIT_EN	Enables bank splitting as a condition when using the placement logic to fill the command queue. b0 = Disabled b1 = Enabled
23–17 -	Reserved.
16 ADDR_CMP_EN	Enables address collision/data coherency detection as a condition when using the placement logic to fill the command queue. b0 = Disabled b1 = Enabled
15–13 -	Reserved.
12–8 COMMAND_AGE_COUNT	Holds the initial value of the command aging counters associated with each command in the command queue. When using the placement logic to fill the command queue, the command aging counters decrement one each time the master aging-rate counter counts down the number of cycles in the age_count parameter. This parameter is only applicable when priority is enabled as a placement factor (the priority_en parameter is set to b1).
7–5 -	Reserved.
4–0 AGE_COUNT	Holds the initial value of the master aging-rate counter. When using the placement logic to fill the command queue, the command aging counters will be decremented one each time the master aging-rate counter counts down age_count cycles. This parameter is only applicable when priority is enabled as a placement factor (the priority_en parameter is set to b1).

20.11.39 DRAM CTL Register 38 (DRAM_CTL38)

Address: DRAM_CTL38 is 1400_0000h base + 98h offset = 1400_0098h



DRAM_CTL38 field descriptions

Field	Description
31–25 -	Reserved.
24 SWAP_EN	Enables swapping of the active command for a new higher-priority command when using the placement logic. b0 = Disabled b1 = Enabled
23–17 -	Reserved.
16 RW_SAME_EN	Enables read/write grouping as a condition when using the placement logic to fill the command queue. b0 = Disabled b1 = Enabled
15–9 -	Reserved.
8 PRIORITY_EN	Enables priority as a condition when using the placement logic to fill the command queue. b0 = Disabled b1 = Enabled
7–1 -	Reserved.
0 PLACEMENT_EN	Enables using the placement logic to fill the command queue. b0 = Placement logic is disabled. The command queue is a straight FIFO.

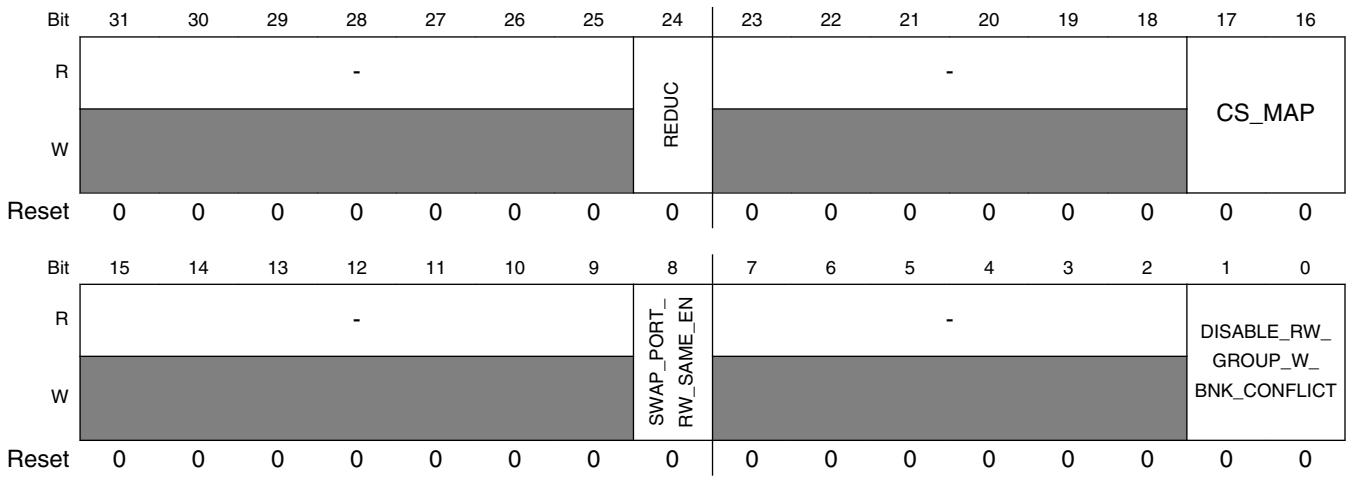
Table continues on the next page...

DRAM_CTL38 field descriptions (continued)

Field	Description
	b1 = Placement logic is enabled. The command queue will be filled according to the placement logic factors.

20.11.40 DRAM CTL Register 39 (DRAM_CTL39)

Address: DRAM_CTL39 is 1400_0000h base + 9Ch offset = 1400_009Ch



DRAM_CTL39 field descriptions

Field	Description
31–25 -	Reserved.
24 REDUC	Allows the MC to be used with memory devices with a smaller datapath (16-bit). When enabled, certain bits of the DFI data bus are unused since the memory data bus is half the width of the initial configured size. The use of bits is detailed in Section 13.7, Data Byte Disable and the Half Datapath Option in the Other Databahn Features Chapter. Note1: The entire user datapath is used regardless of this setting. Note2: It is not possible to run with a burst length of 2 when operating in half datapath mode. b0 = Standard operation using full memory bus. b1 = Memory datapath width is half of the maximum size. Irrelevant bits of the data_byte_disable bus will be driven to b1.
23–18 -	Reserved.
17–16 CS_MAP	Sets the mask that determines which chip select pins are active, with each bit representing a different chip select. The user address chip select field will be mapped into the active chip selects indicated by this parameter in ascending order from lowest to highest. This allows the memory controller to map the entire contiguous user address into any group of chip selects. Bit [0] of this parameter corresponds to chip select [0], bit [1] corresponds to chip select [1], etc. Note1: The number of chip selects, the number of bits set to b1 in this parameter, must be a power of 2 (1, 2, 4 or 8).

Table continues on the next page...

DRAM_CTL39 field descriptions (continued)

Field	Description
	Note2: The bits set must be contiguous.
15–9 -	Reserved.
8 SWAP_PORT_RW_SAME_EN	Note: User must set SWAP_PORT_RW_SAME_EN = 0 for this MC.
7–2 -	Reserved.
1–0 DISABLE_RW_GROUP_W_BNK_CONFLICT	Enables read/write grouping as a condition when using the placement logic to fill the command queue. For more information on this feature, refer to Section 8.1.6, Read/Write Grouping in the Simple Command Queue Example Chapter. Bit [0] = Prohibits placement into a command queue entry immediately before or immediately after a command with a bank conflict. b0 = Allowed b1 = Prohibited Bit [1] = Prohibits placement into a command queue entry 2 entries away (before or after) from a command with a bank conflict. b0 = Allowed b1 = Prohibited Note: It is not meaningful to set bit [1] of this parameter without setting bit [0].

20.11.41 DRAM CTL Register 40 (DRAM_CTL40)

Address: DRAM_CTL40 is 1400_0000h base + A0h offset = 1400_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-							FAST_WRITE	-							LPDDR2_S4
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

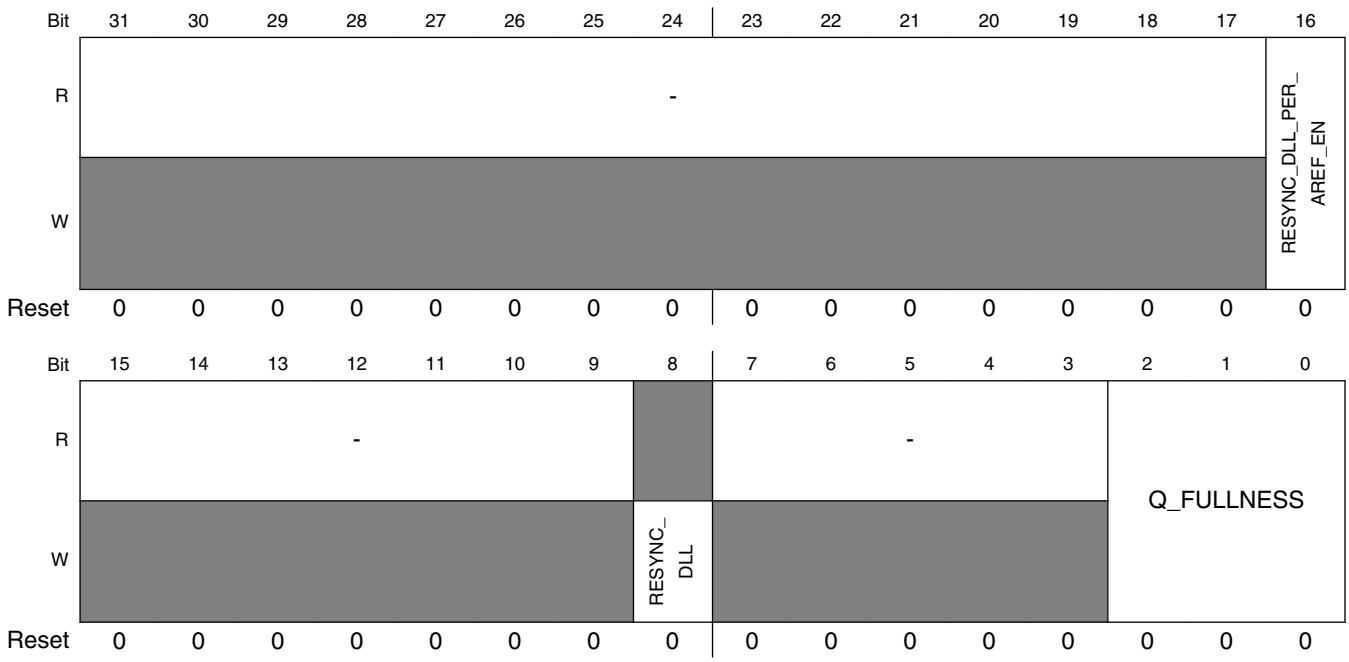
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-							WRDATALAT_REDUC_EN	-							CMDLAT_REDUC_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL40 field descriptions

Field	Description
31–25 -	Reserved.
24 FAST_WRITE	Controls when the write commands are issued to the DRAM devices. b0 = The memory controller will issue a write command to the DRAM devices when it has received enough data for one memory burst. In this mode, write data can be sent in any cycle relative to the write command. This mode also allows for multi-word write command data to arrive in non-sequential cycles. b1 = The memory controller will issue a write command to the DRAM devices after the first word of the write data is received by the memory controller. The first word can be sent at any time relative to the write command. In this mode, multi-word write command data must be available to the memory controller in sequential cycles.
23–17 -	Reserved.
16 LPDDR2_S4	Indicates the type of LPDDR2 device being used. The user must manually set this bit for LPDDR2-S4 devices. This parameter is only applicable when the MC is programmed for the LPDDR2 (dram_class = b0101). b0 = LPDDR2-S2 or LPDDR-NVM device b1 = LPDDR2-S4 device
15–9 -	Reserved.
8 WRDATALAT_ REDUC_EN	Enables data path latency reduction for the I/O cells of the PHY. b0 = Disable b1 = Enable
7–1 -	Reserved.
0 CMDLAT_ REDUC_EN	Enables reducing the latency of the command path in the MC by one clock cycle. b0 = Disabled b1 = Enabled

20.11.42 DRAM CTL Register 41 (DRAM_CTL41)

Address: DRAM_CTL41 is 1400_0000h base + A4h offset = 1400_00A4h

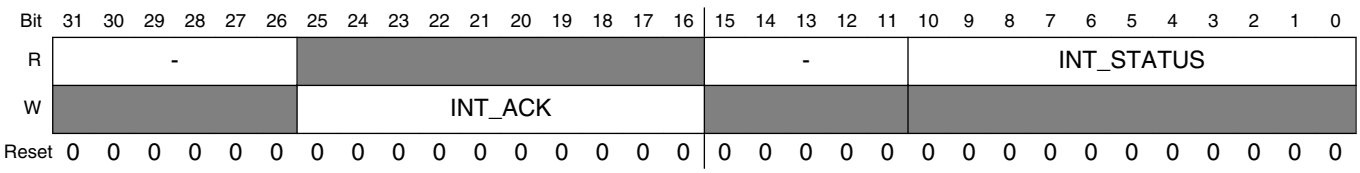


DRAM_CTL41 field descriptions

Field	Description
31–17 -	Reserved.
16 RESYNC_DLL_PER_AREF_EN	Enables an automatic re-synchronization of the DLL after every refresh.
15–9 -	Reserved.
8 RESYNC_DLL	Initiates a re-synchronization of the DLL. This parameter is write-only. Note: For proper operation, the user must issue at least one read or write command to memory after reset prior to setting this parameter.
7–3 -	Reserved.
2–0 Q_FULLNESS	Defines quantity of data that will be considered full for the command queue. When this value is reached, the q_almost_full parameter will be set.

20.11.43 DRAM CTL Register 42 (DRAM_CTL42)

Address: DRAM_CTL42 is 1400_0000h base + A8h offset = 1400_00A8h

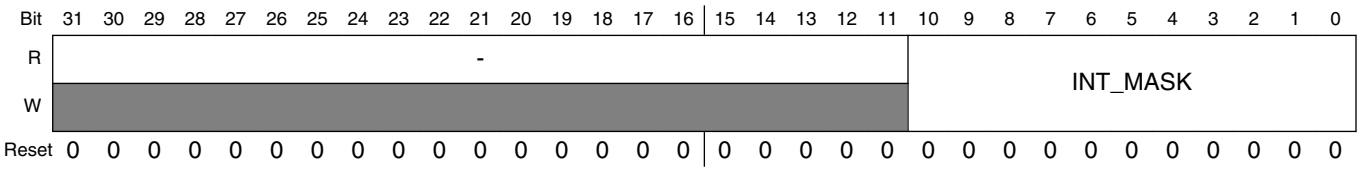


DRAM_CTL42 field descriptions

Field	Description
31–26 -	Reserved.
25–16 INT_ACK	Controls the clearing of the int_status parameter. If any of the int_ack bits are set to b1, the corresponding bit in the int_status parameter will be cleared to b0. Any int_ack bits cleared to b0 will not alter the corresponding bit in the int_status parameter. This parameter will always read back as 0x0.
15–11 -	Reserved.
10–0 INT_STATUS	Shows the status of all possible interrupts generated by the memory controller. The MSB is the result of a logical OR of all the lower bits. READ-ONLY. Note: Backwards compatibility is available for register parameters across configurations. However, even with this compatibility, the individual bits, their meaning and the size of the int_status parameter may change. The int_status bits correspond to these interrupts: Bit [10] = Logical OR of all lower bits. Bit [9] = User-initiated DLL resync is finished. Bit [8] = dfi_init_complete state change detected. Bit [7] = Indicates that a register interface mode register write has finished and that another register interface mode register write may be issued. Bit [6] = Indicates that a peripheral register interface mode register read has finished and that the mode register data and chip can be found in the param_peripheral_mrr_data register. Bit [5] = ODT enabled and CAS Latency 3 programmed error detected. This is an unsupported programming option. Bit [4] = DRAM initialization complete. Bit [3] = Error was found with command data channel in a port. Bit [2] = Error was found with command channel in a port. Bit [1] = Multiple accesses outside the defined PHYSICAL memory space detected. Bit [0] = A single access outside the defined PHYSICAL memory space detected.

20.11.44 DRAM CTL Register 43 (DRAM_CTL43)

Address: DRAM_CTL43 is 1400_0000h base + ACh offset = 1400_00ACh

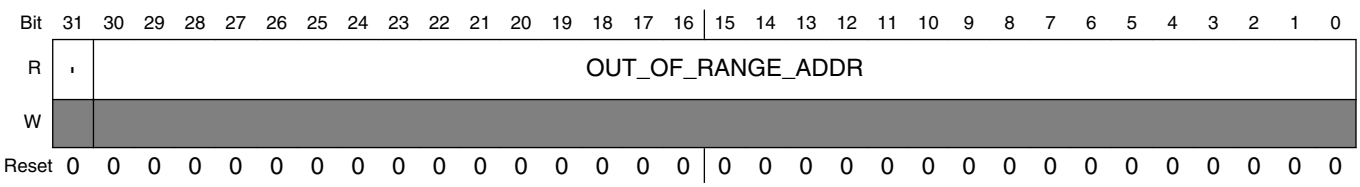


DRAM_CTL43 field descriptions

Field	Description
31–11 -	Reserved.
10–0 INT_MASK	Active-high mask bits that control the value of the memory controller_int signal on the ASIC interface. Unless the user has suppressed interrupt reporting (by setting the most significant bit of this parameter to b1), all lower bits of the int_mask parameter will be inverted and logically ANDed with the corresponding bits of the int_status parameter and the result is reported on the controller_int signal.

20.11.45 DRAM CTL Register 44 (DRAM_CTL44)

Address: DRAM_CTL44 is 1400_0000h base + B0h offset = 1400_00B0h

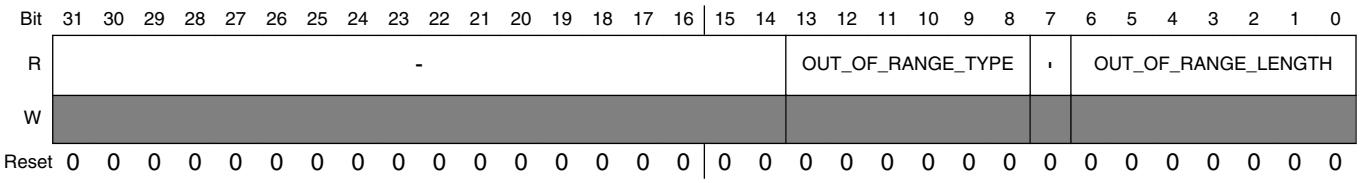


DRAM_CTL44 field descriptions

Field	Description
31 -	Reserved.
30–0 OUT_OF_ RANGE_ADDR	Holds the address of the command that caused an out-of-range interrupt request to the memory devices. READ-ONLY. For more information, refer to the Other Databahn Features Chapter.

20.11.46 DRAM CTL Register 45 (DRAM_CTL45)

Address: DRAM_CTL45 is 1400_0000h base + B4h offset = 1400_00B4h

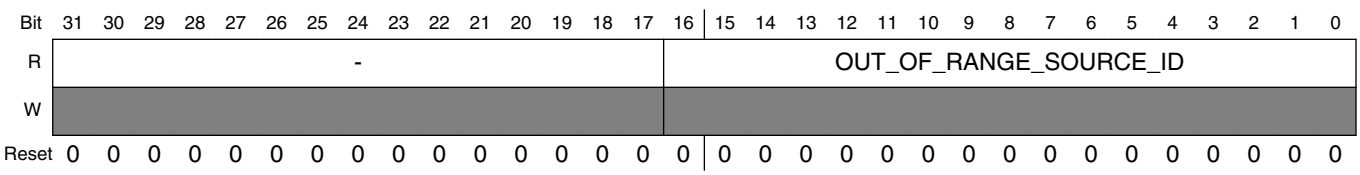


DRAM_CTL45 field descriptions

Field	Description
31–14 -	Reserved.
13–8 OUT_OF_ RANGE_TYPE	Holds the type of command that caused an out-of-range interrupt request to the memory devices. READ-ONLY. For more information, refer to the Other Databahn Features Chapter.
7 -	Reserved.
6–0 OUT_OF_ RANGE_ LENGTH	Holds the length of the command that caused an out-of-range interrupt request to the memory devices. READ-ONLY. For more information, refer to the Other Databahn Features Chapter.

20.11.47 DRAM CTL Register 46 (DRAM_CTL46)

Address: DRAM_CTL46 is 1400_0000h base + B8h offset = 1400_00B8h

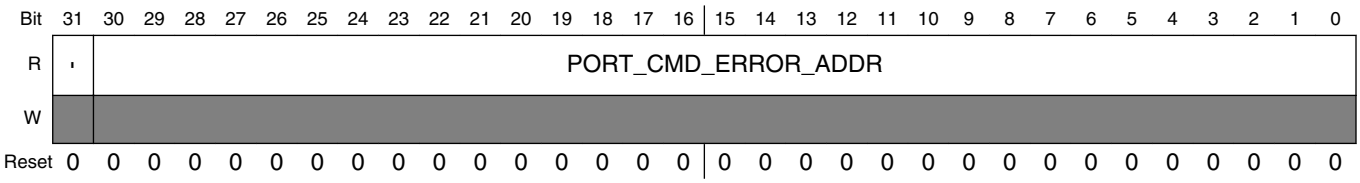


DRAM_CTL46 field descriptions

Field	Description
31–17 -	Reserved.
16–0 OUT_OF_ RANGE_ SOURCE_ID	Holds the Source ID of the command that caused an out-of-range interrupt request to the memory devices. READ-ONLY. For more information, refer to the Other Databahn Features Chapter.

20.11.48 DRAM CTL Register 47 (DRAM_CTL47)

Address: DRAM_CTL47 is 1400_0000h base + BCh offset = 1400_00BCh

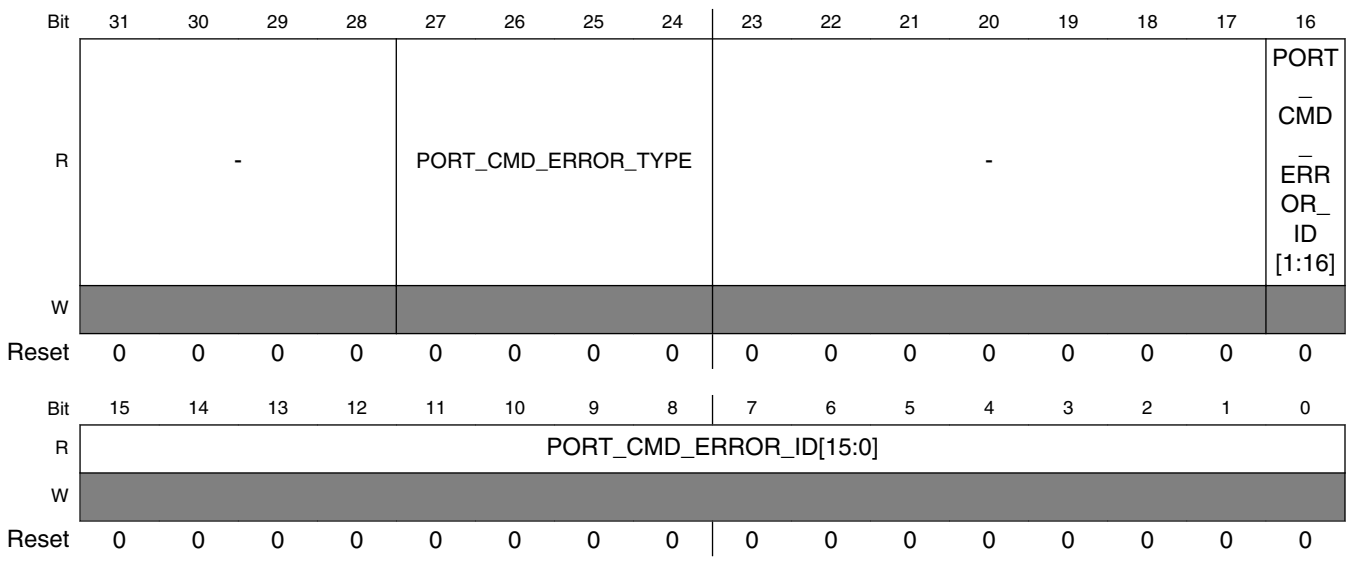


DRAM_CTL47 field descriptions

Field	Description
31 -	Reserved.
30–0 PORT_CMD_ ERROR_ADDR	Holds the address of the command that caused a port command error condition. READ-ONLY.

20.11.49 DRAM CTL Register 48 (DRAM_CTL48)

Address: DRAM_CTL48 is 1400_0000h base + C0h offset = 1400_00C0h



DRAM_CTL48 field descriptions

Field	Description
31–28 -	Reserved.

Table continues on the next page...

DRAM_CTL48 field descriptions (continued)

Field	Description
27–24 PORT_CMD_ ERROR_TYPE	Defines the type of error and the access type that caused the port command error condition. If multiple bits are set to b1, then multiple errors were found. READ-ONLY. Bit [3] = Narrow transfer requested for a requestor from port Y whose axiY_en_size_lt_width_instr parameter is clear. Bit [2] = Reserved Bit [1] = Reserved Bit [0] = Reserved
23–17 -	Reserved.
16–0 PORT_CMD_ ERROR_ID	Holds the source ID of the command that caused a port command error condition. For AXI ports, the source ID is comprised of the Port ID and the Requestor ID, where the Requestor ID is the axiY_AWID for write commands or the axiY_ARID for read commands. READ-ONLY.

20.11.50 DRAM CTL Register 49 (DRAM_CTL49)

Address: DRAM_CTL49 is 1400_0000h base + C4h offset = 1400_00C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-					PORT_DATA_ERROR_TYPE			-							PORT_DATA_ERROR_ID[1:16]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PORT_DATA_ERROR_ID[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL49 field descriptions

Field	Description
31–27 -	Reserved.
26–24 PORT_DATA_ ERROR_TYPE	Defines the type of error and the access type that caused the port data error condition. If multiple bits are set to b1, then multiple errors were found. READ-ONLY. Bit [2] = Reserved Bit [1] = Reserved

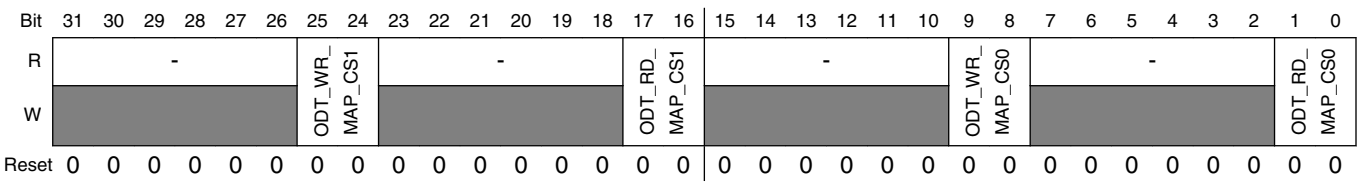
Table continues on the next page...

DRAM_CTL49 field descriptions (continued)

Field	Description
	Bit [0] = Reserved
23–17 -	Reserved.
16–0 PORT_DATA_ERROR_ID	Holds the source ID of the command that caused a port data error condition. For AXI ports, the source ID is comprised of the Port ID and the Requestor ID, where the Requestor ID is the axiY_BID for write response errors, the axiY_RID for read data errors, or the axiY_WID for write data errors. READ-ONLY.

20.11.51 DRAM CTL Register 50 (DRAM_CTL50)

Address: DRAM_CTL50 is 1400_0000h base + C8h offset = 1400_00C8h



DRAM_CTL50 field descriptions

Field	Description
31–26 -	Reserved.
25–24 ODT_WR_MAP_CS1	Sets up which (if any) chip(s) will have their ODT termination active while a write occurs on chip select 1. Refer to ODT_WR_MAP_CS0 for detailed description.
23–18 -	Reserved.
17–16 ODT_RD_MAP_CS1	Sets up which chip(s) will have their ODT termination active while a read occurs on chip select 1. Refer to ODT_RD_MAP_CS0 for detailed description.
15–10 -	Reserved.
9–8 ODT_WR_MAP_CS0	Sets up which (if any) chip(s) will have their ODT termination active while a write occurs on chip select 0. Example: If the system consists of 2 chip selects and odt_wr_map_cs0 is set to b10, then when CS0 is performing a write, CS1 will have active ODT termination. And if odt_wr_map_cs0 was set to b01, then instead CS0 would be active. Bit [1] = CS1 will have active ODT termination during a write. Bit [0] = CS0 will have active ODT termination during a write.
7–2 -	Reserved.

Table continues on the next page...

DRAM_CTL50 field descriptions (continued)

Field	Description
1–0 ODT_RD_MAP_CS0	Sets up which chip(s) will have their ODT termination active while a read occurs on chip select 0. Example: The system consists of 2 chip selects and odt_rd_map_cs0 is set to b10, then when CS0 is performing a read, CS1 will have active ODT termination. And if odt_rd_map_cs0 was set to b01, then instead CS0 would be active. Bit [1] = CS1 will have active ODT termination during a read. Bit [0] = CS0 will have active ODT termination during a read.

20.11.52 DRAM CTL Register 51 (DRAM_CTL51)

Address: DRAM_CTL51 is 1400_0000h base + CCh offset = 1400_00CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-				ADD_ODT_CLK_SAMETYPE_DIFFCS				-			ADD_ODT_CLK_DIFFTYPE_DIFFCS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-				ADD_ODT_CLK_W2R_SAMECS				-				ADD_ODT_CLK_R2W_SAMECS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL51 field descriptions

Field	Description
31–28 -	Reserved.
27–24 ADD_ODT_CLK_SAMETYPE_DIFFCS	Defines the number of additional clocks of delay to insert between commands of the same type (read to read, write to write) to different chip selects to meet ODT timing requirements.
23–21 -	Reserved.
20–16 ADD_ODT_CLK_DIFFTYPE_DIFFCS	Defines the number of additional clocks of delay to insert between commands of different types (read to write, write to read) to different chip selects to meet ODT timing requirements.
15–12 -	Reserved.

Table continues on the next page...

DRAM_CTL51 field descriptions (continued)

Field	Description
11–8 ADD_ODT_ CLK_W2R_ SAMECS	Defines the number of additional clocks of delay to insert after a write command before a read command to the same chip select to meet ODT timing requirements.
7–4 -	Reserved.
3–0 ADD_ODT_ CLK_R2W_ SAMECS	Defines the number of additional clocks of delay to insert after a read command before a write command to the same chip select to meet ODT timing requirements.

20.11.53 DRAM CTL Register 52 (DRAM_CTL52)

Address: DRAM_CTL52 is 1400_0000h base + D0h offset = 1400_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-					W2W_DIFFCS_DLY			-					W2R_DIFFCS_DLY			-					R2W_DIFFCS_DLY			-					R2R_DIFFCS_DLY		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL52 field descriptions

Field	Description
31–27 -	Reserved.
26–24 W2W_DIFFCS_ DLY	Defines the number of additional clocks of delay to insert from a write command to one chip select to a write command to a different chip select. The minimum allowable programming of this parameter is 1 cycle.
23–19 -	Reserved.
18–16 W2R_DIFFCS_ DLY	Defines the number of additional clocks of delay to insert from a write command to one chip select to a read command to a different chip select. The minimum allowable programming of this parameter is 1 cycle.
15–11 -	Reserved.
10–8 R2W_DIFFCS_ DLY	Defines the number of additional clocks of delay to insert from a read command to one chip select to a write command to a different chip select. The minimum allowable programming of this parameter is 1 cycle.
7–3 -	Reserved.
2–0 R2R_DIFFCS_ DLY	Defines the number of additional clocks of delay to insert from a read command to one chip select to a read command to a different chip select. The minimum allowable programming of this parameter is 1 cycle.

20.11.54 DRAM CTL Register 53 (DRAM_CTL53)

Address: DRAM_CTL53 is 1400_0000h base + D4h offset = 1400_00D4h

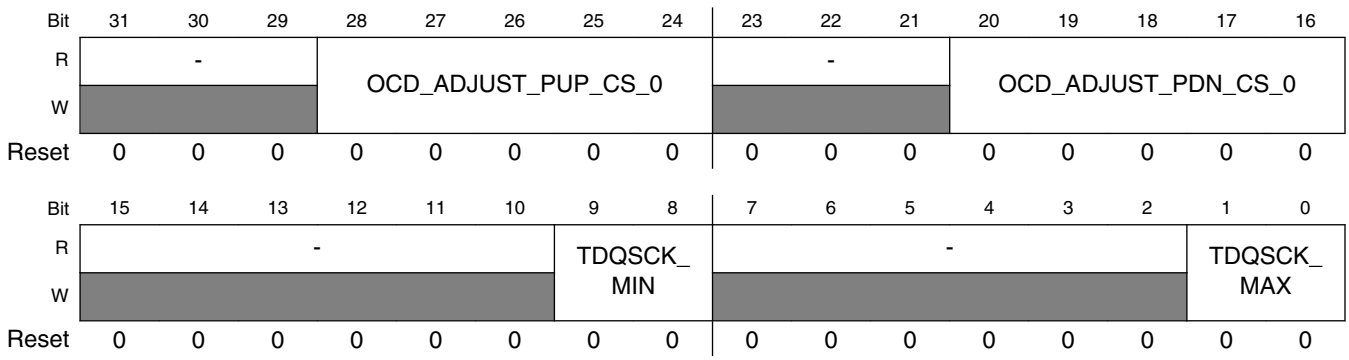
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-					W2W_SAMECS_DLY			-					W2R_SAMECS_DLY			-					R2W_SAMECS_DLY			-					R2R_SAMECS_DLY		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL53 field descriptions

Field	Description
31–27 -	Reserved.
26–24 W2W_SAMECS_DLY	Defines the number of additional clocks of delay to insert between two write commands to the same chip select.
23–19 -	Reserved.
18–16 W2R_SAMECS_DLY	Defines the number of additional clocks of delay to insert from a write command to a read command to the same chip select. The minimum allowable programming of this parameter is 1 cycle.
15–11 -	Reserved.
10–8 R2W_SAMECS_DLY	Defines the number of additional clocks of delay to insert from a read command to a write command to the same chip select. The minimum allowable programming of this parameter is 1 cycle.
7–3 -	Reserved.
2–0 R2R_SAMECS_DLY	Defines the number of additional clocks of delay to insert between two read commands to the same chip select.

20.11.55 DRAM CTL Register 54 (DRAM_CTL54)

Address: DRAM_CTL54 is 1400_0000h base + D8h offset = 1400_00D8h

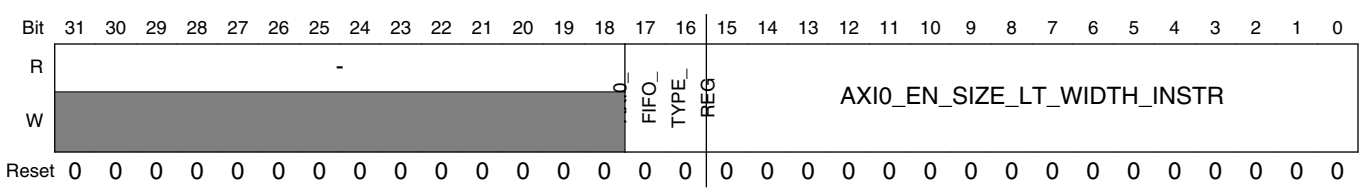


DRAM_CTL54 field descriptions

Field	Description
31–29 -	Reserved.
28–24 OCD_ADJUST_PUP_CS_0	This parameter is reserved for future use and should be programmed to 0x0.
23–21 -	Reserved.
20–16 OCD_ADJUST_PDN_CS_0	This parameter is reserved for future use and should be programmed to 0x0.
15–10 -	Reserved.
9–8 TDQSCK_MIN	Additional delay needed for tDQSCK
7–2 -	Reserved.
1–0 TDQSCK_MAX	Additional delay needed for tDQSCK

20.11.56 DRAM CTL Register 55 (DRAM_CTL55)

Address: DRAM_CTL55 is 1400_0000h base + DCh offset = 1400_00DCh

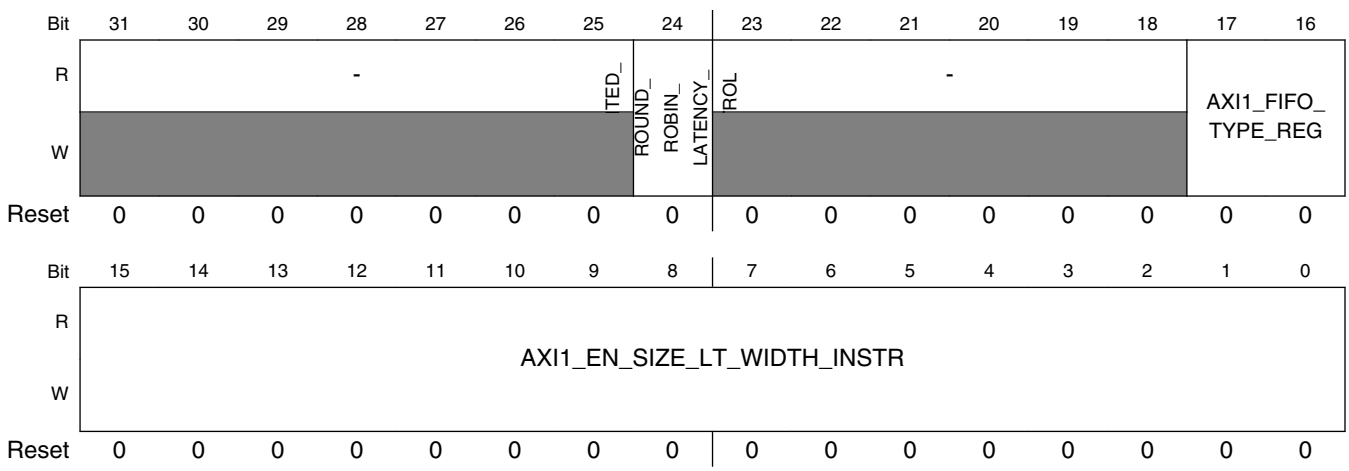


DRAM_CTL55 field descriptions

Field	Description
31–18 -	Reserved.
17–16 AXI0_FIFO_ TYPE_REG	Sets the relativity of the clock domains between AXI port Y and the MC core clock. b00 = Asynchronous b01 = Reserved b10 = Reserved b11 = Synchronous
15–0 AXI0_EN_SIZE_ LT_WIDTH_ INSTR	Allows the port to accept size less than width transactions on AXI port Y from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port Y will be able to send size less than width transactions. Note: User must set AXI0_EN_SIZE_LT_WIDTH_INSTR = 0xFFFF for this MC. b0 = Requestor Z of port Y may only issue size equal to width instructions. b1 = Requestor Z of port Y can issue size equal to width or size less than width instructions.

20.11.57 DRAM CTL Register 56 (DRAM_CTL56)

Address: DRAM_CTL56 is 1400_0000h base + E0h offset = 1400_00E0h



DRAM_CTL56 field descriptions

Field	Description
31–25 -	Reserved.
24 WEIGHTED_ ROUND_ ROBIN_	Controls the weighted round-robin latency option. b0 = Counters only count when their port has a command waiting to be processed. b1 = Counters are always running.

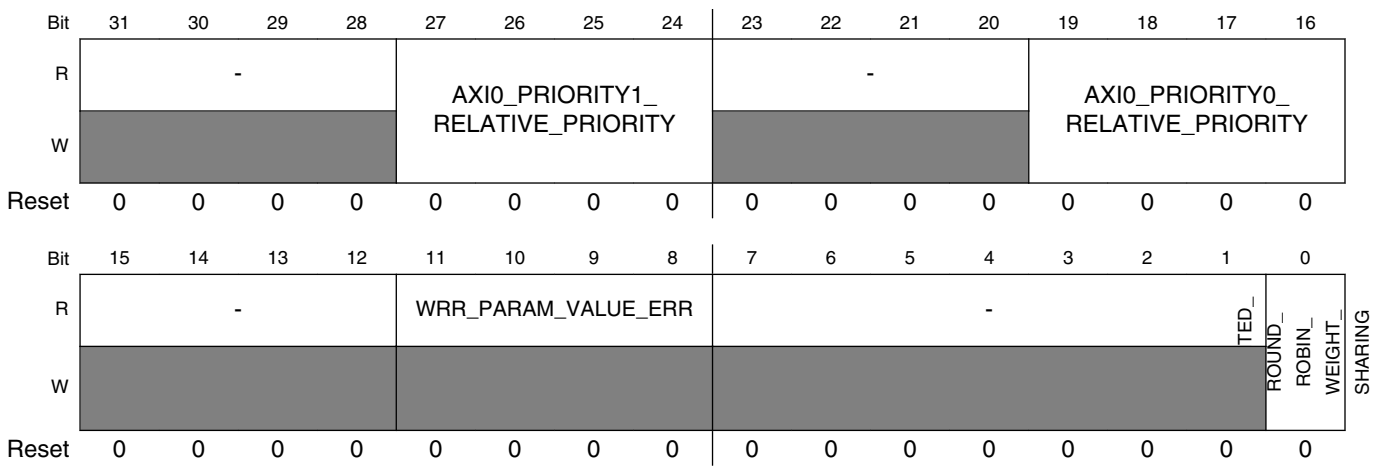
Table continues on the next page...

DRAM_CTL56 field descriptions (continued)

Field	Description
LATENCY_ CONTROL	
23–18 -	Reserved.
17–16 AXI1_FIFO_ TYPE_REG	AXI1 port is tied-off in this MC.
15–0 AXI1_EN_SIZE_ LT_WIDTH_ INSTR	AXI1 port is tied-off in this MC.

20.11.58 DRAM CTL Register 57 (DRAM_CTL57)

Address: DRAM_CTL57 is 1400_0000h base + E4h offset = 1400_00E4h



DRAM_CTL57 field descriptions

Field	Description
31–28 -	Reserved.
27–24 AXI0_ PRIORITY1_ RELATIVE_ PRIORITY	Holds the relative priority of the AXI port 0 for priority 1 commands in weighted round robin arbitration.
23–20 -	Reserved.
19–16 AXI0_ PRIORITY0_	Holds the relative priority of the AXI port 0 for priority 0 commands in weighted round robin arbitration.

Table continues on the next page...

DRAM_CTL57 field descriptions (continued)

Field	Description
RELATIVE_ PRIORITY	
15–12 -	Reserved.
11–8 WRR_PARAM_ VALUE_ERR	Shows the weighted round-robin arbitration errors/warnings. READ-ONLY. Bit [3] = The port ordering parameter values for paired ports is not sequential. Bit [2] = The relative priority values for any of the ports paired through the weighted_round_robin_weight_sharing parameter are not identical. Bit [1] = Any of the relative priority parameters have been programmed with a zero value. Bit [0] = The port ordering parameters do not all contain unique values.
7–1 -	Reserved.
0 WEIGHTED_ ROUND_ ROBIN_ WEIGHT_ SHARING	Indicates that the port pair is tied together in arbitration decisions in weighted round-robin arbitration. Bit [0] represents ports 0 and 1, bit [1] represents ports 2 and 3, etc. Each bit setting is as follows: 'b0 = The represented ports are treated independently in arbitration. b1 = The represented ports are tied together for arbitration.

20.11.59 DRAM CTL Register 58 (DRAM_CTL58)

Address: DRAM_CTL58 is 1400_0000h base + E8h offset = 1400_00E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-				AXI0_PRIORITY5_ RELATIVE_PRIORITY				-				AXI0_PRIORITY4_ RELATIVE_PRIORITY			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-				AXI0_PRIORITY3_ RELATIVE_PRIORITY				-				AXI0_PRIORITY2_ RELATIVE_PRIORITY			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL58 field descriptions

Field	Description
31–28 -	Reserved.
27–24 AXI0_ PRIORITY5_ RELATIVE_ PRIORITY	Holds the relative priority of the AXI port 0 for priority 5 commands in weighted round robin arbitration.

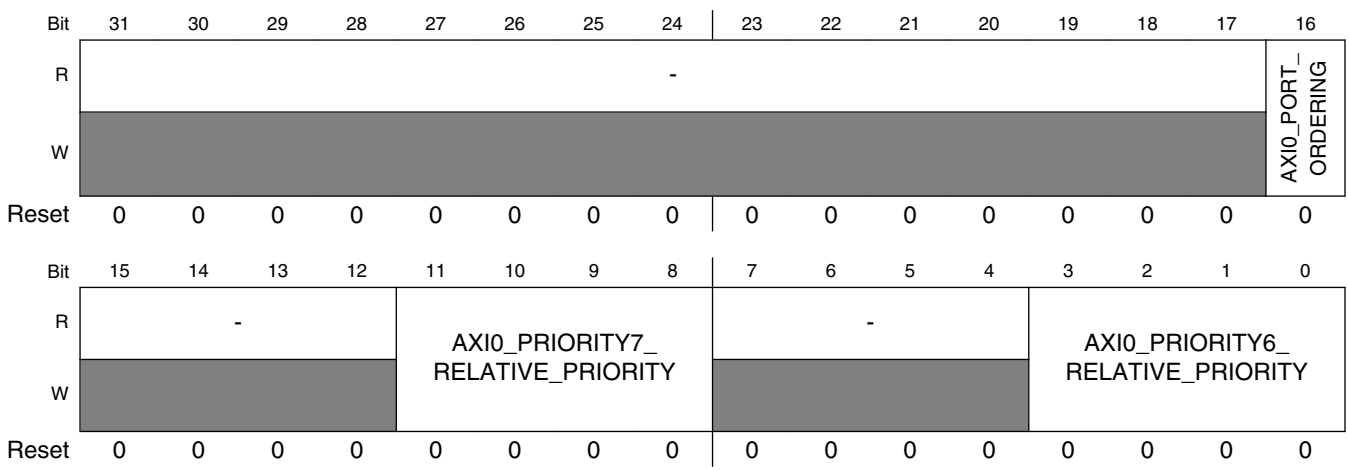
Table continues on the next page...

DRAM_CTL58 field descriptions (continued)

Field	Description
23–20 -	Reserved.
19–16 AXI0_ PRIORITY4_ RELATIVE_ PRIORITY	Holds the relative priority of the AXI port 0 for priority 4 commands in weighted round robin arbitration.
15–12 -	Reserved.
11–8 AXI0_ PRIORITY3_ RELATIVE_ PRIORITY	Holds the relative priority of the AXI port 0 for priority 3 commands in weighted round robin arbitration.
7–4 -	Reserved.
3–0 AXI0_ PRIORITY2_ RELATIVE_ PRIORITY	Holds the relative priority of the AXI port 0 for priority 2 commands in weighted round robin arbitration.

20.11.60 DRAM CTL Register 59 (DRAM_CTL59)

Address: DRAM_CTL59 is 1400_0000h base + ECh offset = 1400_00ECh



DRAM_CTL59 field descriptions

Field	Description
31–17 -	Reserved.

Table continues on the next page...

DRAM_CTL59 field descriptions (continued)

Field	Description
16 AXI0_PORT_ ORDERING	Used in weighted round-robin arbitration to modify the order than the ports are scanned when multiple commands are at the same priority level and have the same relative priorities.
15–12 -	Reserved.
11–8 AXI0_ PRIORITY7_ RELATIVE_ PRIORITY	Holds the relative priority of the AXI port 0 for priority 7 commands in weighted round robin arbitration.
7–4 -	Reserved.
3–0 AXI0_ PRIORITY6_ RELATIVE_ PRIORITY	Holds the relative priority of the AXI port 0 for priority 6 commands in weighted round robin arbitration.

20.11.61 DRAM CTL Register 60 (DRAM_CTL60)

Address: DRAM_CTL60 is 1400_0000h base + F0h offset = 1400_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-				AXI1_PRIORITY1_ RELATIVE_PRIORITY				-				AXI1_PRIORITY0_ RELATIVE_PRIORITY			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								AXI0_PRIORITY_RELAX							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL60 field descriptions

Field	Description
31–28 -	Reserved.
27–24 AXI1_ PRIORITY1_ RELATIVE_ PRIORITY	AXI1 port is tied-off in this MC.
23–20 -	Reserved.

Table continues on the next page...

DRAM_CTL60 field descriptions (continued)

Field	Description
19–16 AXI1_ PRIORITY0_ RELATIVE_ PRIORITY	AXI1 port is tied-off in this MC.
15–10 -	Reserved.
9–0 AXI0_ PRIORITY_ RELAX	Holds the counter value for AXI port Y at which the priority relax condition is triggered in weighted round robin arbitration.

20.11.62 DRAM CTL Register 61 (DRAM_CTL61)

Address: DRAM_CTL61 is 1400_0000h base + F4h offset = 1400_00F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-				AXI1_PRIORITY5_ RELATIVE_PRIORITY				-				AXI1_PRIORITY4_ RELATIVE_PRIORITY			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-				AXI1_PRIORITY3_ RELATIVE_PRIORITY				-				AXI1_PRIORITY2_ RELATIVE_PRIORITY			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL61 field descriptions

Field	Description
31–28 -	Reserved.
27–24 AXI1_ PRIORITY5_ RELATIVE_ PRIORITY	AXI1 port is tied-off in this MC.
23–20 -	Reserved.
19–16 AXI1_ PRIORITY4_ RELATIVE_ PRIORITY	AXI1 port is tied-off in this MC.

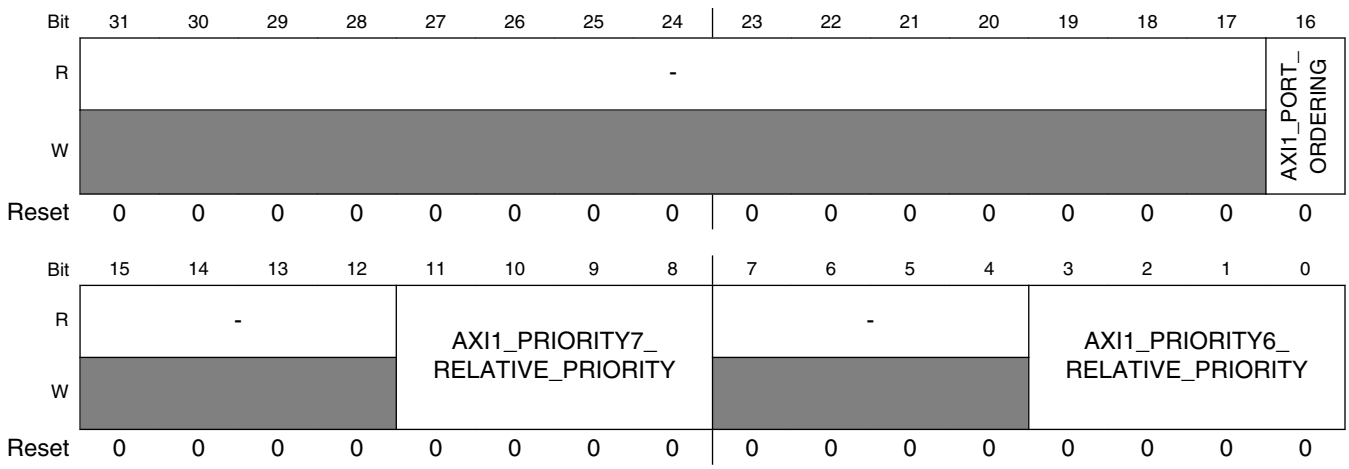
Table continues on the next page...

DRAM_CTL61 field descriptions (continued)

Field	Description
15–12 -	Reserved.
11–8 AXI1_ PRIORITY3_ RELATIVE_ PRIORITY	AXI1 port is tied-off in this MC.
7–4 -	Reserved.
3–0 AXI1_ PRIORITY2_ RELATIVE_ PRIORITY	AXI1 port is tied-off in this MC.

20.11.63 DRAM CTL Register 62 (DRAM_CTL62)

Address: DRAM_CTL62 is 1400_0000h base + F8h offset = 1400_00F8h



DRAM_CTL62 field descriptions

Field	Description
31–17 -	Reserved.
16 AXI1_PORT_ ORDERING	AXI1 port is tied-off in this MC.
15–12 -	Reserved.

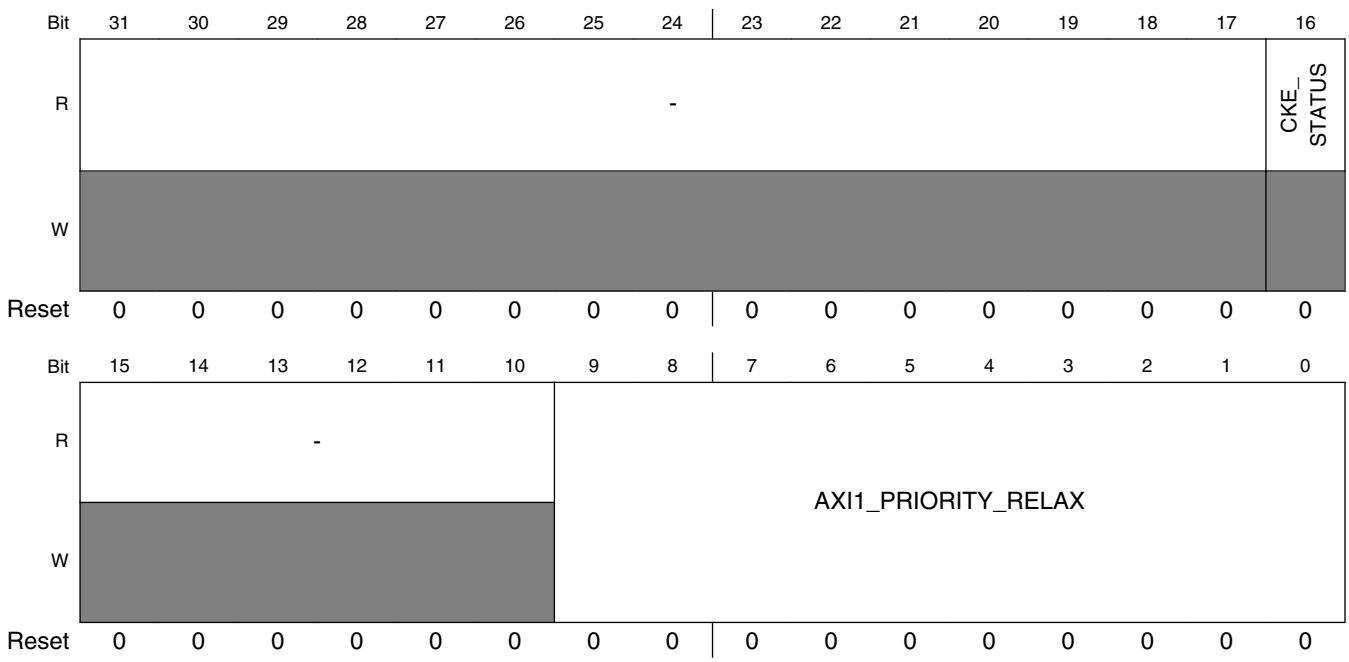
Table continues on the next page...

DRAM_CTL62 field descriptions (continued)

Field	Description
11–8 AXI1_ PRIORITY7_ RELATIVE_ PRIORITY	AXI1 port is tied-off in this MC.
7–4 -	Reserved.
3–0 AXI1_ PRIORITY6_ RELATIVE_ PRIORITY	AXI1 port is tied-off in this MC.

20.11.64 DRAM CTL Register 63 (DRAM_CTL63)

Address: DRAM_CTL63 is 1400_0000h base + FCh offset = 1400_00FCh



DRAM_CTL63 field descriptions

Field	Description
31–17 -	Reserved.
16 CKE_STATUS	Provides the value of the cke_status signal in a parameter. This signal is the value of the control_cke signal inside the MC, delayed by the number of clocks specified in the cke_delay parameter. READ-ONLY.

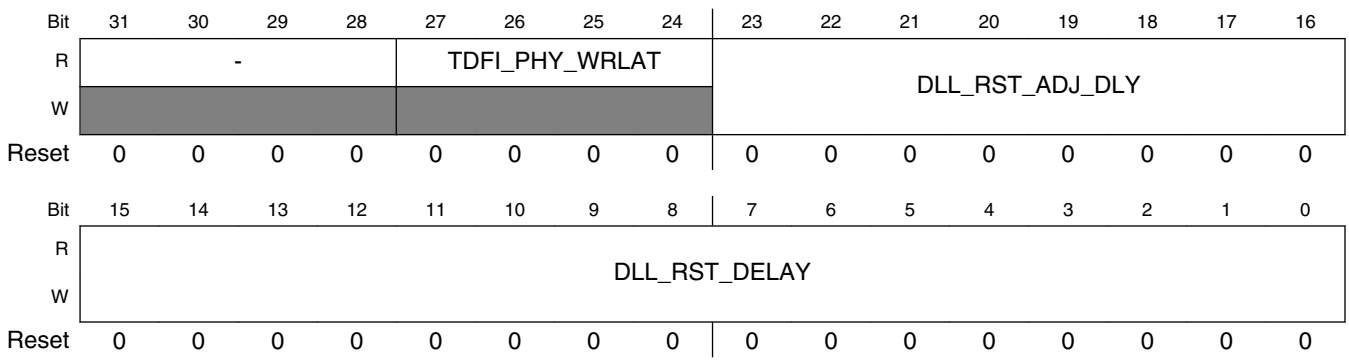
Table continues on the next page...

DRAM_CTL63 field descriptions (continued)

Field	Description
15–10 -	Reserved.
9–0 AXI1_ PRIORITY_ RELAX	AXI1 port is tied-off in this MC.

20.11.65 DRAM CTL Register 64 (DRAM_CTL64)

Address: DRAM_CTL64 is 1400_0000h base + 100h offset = 1400_0100h



DRAM_CTL64 field descriptions

Field	Description
31–28 -	Reserved.
27–24 TDFI_PHY_ WRLAT	Holds the calculated DFI tPHY_WRLAT timing parameter. READ-ONLY
23–16 DLL_RST_ADJ_ DLY	Specifies the minimum number of cycles after the master delay value is programmed before the DLL reset may be asserted.
15–0 DLL_RST_ DELAY	Sets the number of cycles that the reset must be held asserted for the DLL.

20.11.66 DRAM CTL Register 65 (DRAM_CTL65)

Address: DRAM_CTL65 is 1400_0000h base + 104h offset = 1400_0104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-			TDFI_RDDATA_EN_BASE					-			TDFI_RDDATA_EN					-			TDFI_PHY_RDLAT					-				TDFI_PHY_WRLAT_BASE			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL65 field descriptions

Field	Description
31–29 -	Reserved.
28–24 TDFI_RDDATA_EN_BASE	Sets DFI base value for the tRDDATA_EN timing parameter.
23–21 -	Reserved.
20–16 TDFI_RDDATA_EN	Holds the calculated DFI tRDDATA_EN timing parameter. READ-ONLY
15–13 -	Reserved.
12–8 TDFI_PHY_RDLAT	Holds the tPHY_RDLAT timing parameter.
7–4 -	Reserved.
3–0 TDFI_PHY_WRLAT_BASE	Sets DFI base value for the tPHY_WRLAT timing parameter.

20.11.67 DRAM CTL Register 66 (DRAM_CTL66)

Address: DRAM_CTL66 is 1400_0000h base + 108h offset = 1400_0108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TDFI_CTRLUPD_MAX															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-				TDFI_CTRLUPD_MIN				-						DRAM_CLK_DISABLE	
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

DRAM_CTL66 field descriptions

Field	Description
31–16 TDFI_ CTRLUPD_MAX	Defines the maximum number of DFI clock cycles that the dfi_ctrlupd_req signal can assert.
15–12 -	Reserved.
11–8 TDFI_ CTRLUPD_MIN	Holds the minimum number of DFI clock cycles that the dfi_ctrlupd_req signal must be asserted. READ-ONLY
7–2 -	Reserved.
1–0 DRAM_CLK_ DISABLE	Sets value for the DFI output signal dfi_dram_clk_disable. Bit [0] controls CS0, Bit [1] controls CS1. For each bit: b0 = Memory clock/s should be active. b1 = Memory clock/s should be disabled.

20.11.68 DRAM CTL Register 67 (DRAM_CTL67)

Address: DRAM_CTL67 is 1400_0000h base + 10Ch offset = 1400_010Ch

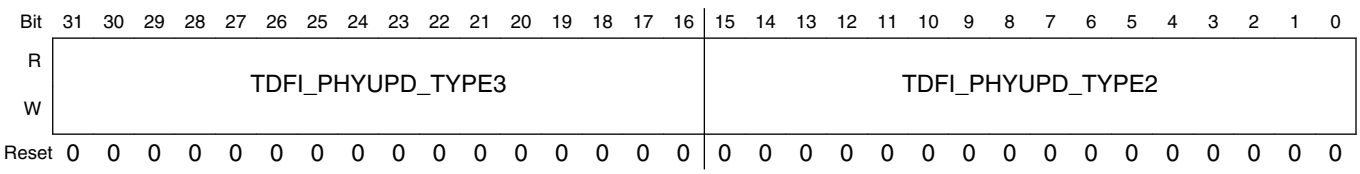
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TDFI_PHYUPD_TYPE1																TDFI_PHYUPD_TYPE0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL67 field descriptions

Field	Description
31–16 TDFI_PHYUPD_TYPE1	Defines the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type1.
15–0 TDFI_PHYUPD_TYPE0	Defines the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type0.

20.11.69 DRAM CTL Register 68 (DRAM_CTL68)

Address: DRAM_CTL68 is 1400_0000h base + 110h offset = 1400_0110h

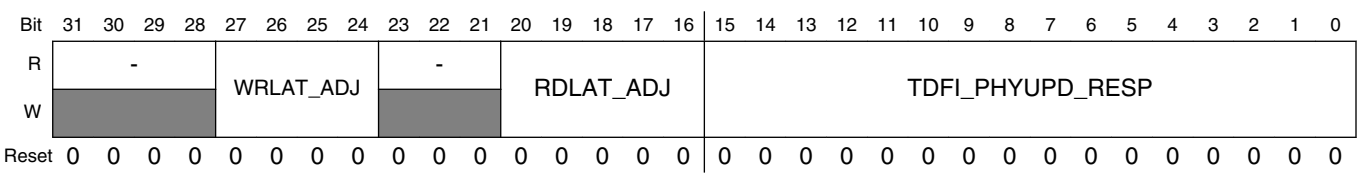


DRAM_CTL68 field descriptions

Field	Description
31–16 TDFI_PHYUPD_TYPE3	Defines the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type3.
15–0 TDFI_PHYUPD_TYPE2	Defines the maximum number of DFI clock cycles that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type2.

20.11.70 DRAM CTL Register 69 (DRAM_CTL69)

Address: DRAM_CTL69 is 1400_0000h base + 114h offset = 1400_0114h



DRAM_CTL69 field descriptions

Field	Description
31–28 -	Reserved.

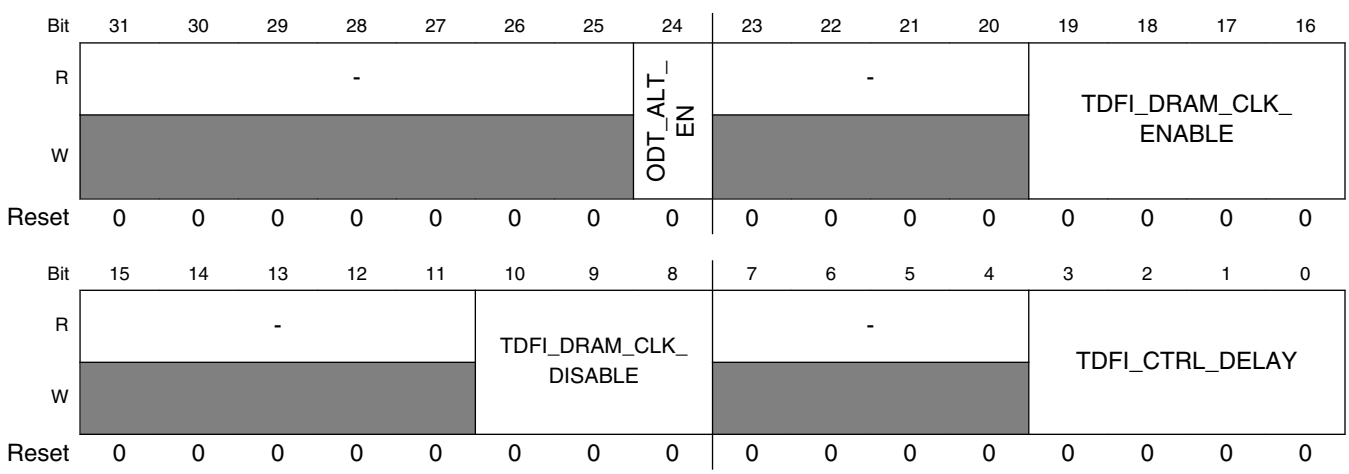
Table continues on the next page...

DRAM_CTL69 field descriptions (continued)

Field	Description
27–24 WRLAT_ADJ	Adjusts the relative timing between DFI write commands and the dfi_wrdata_en signal to conform to PHY timing requirements. Note: For LPDDR2 memory systems (dram_class = b0101), the wrlat_adj value is incremented inside the memory controller by 1 to compensate for TDSS. When this parameter is programmed to 0x0, dfi_wrdata_en will assert on the same cycle as the dfi_address. The sum of the tdfi_phy_wrlat_base parameter and this parameter must be at least 0x3. These two parameters work together to set the actual PHY write latency (tdfi_phy_wrlat). This parameter only affects the DFI.
23–21 -	Reserved.
20–16 RDLAT_ADJ	Adjusts the relative timing between DFI read commands and the dfi_rddata_en signal to conform to PHY timing requirements. When this parameter is programmed to 0x0, dfi_rddata_en will assert one cycle after the dfi_address. The sum of the tdfi_rddata_en_base parameter and this parameter must be at least 0x4. These two parameters work together to set the actual PHY read latency (tdfi_rddata_en). This parameter only affects the DFI.
15–0 TDFI_PHYUPD_RESP	Defines the maximum number of DFI clock cycles after the assertion of the dfi_phyupd_req signal to the assertion of the dfi_phyupd_ack signal.

20.11.71 DRAM CTL Register 70 (DRAM_CTL70)

Address: DRAM_CTL70 is 1400_0000h base + 118h offset = 1400_0118h



DRAM_CTL70 field descriptions

Field	Description
31–25 -	Reserved.

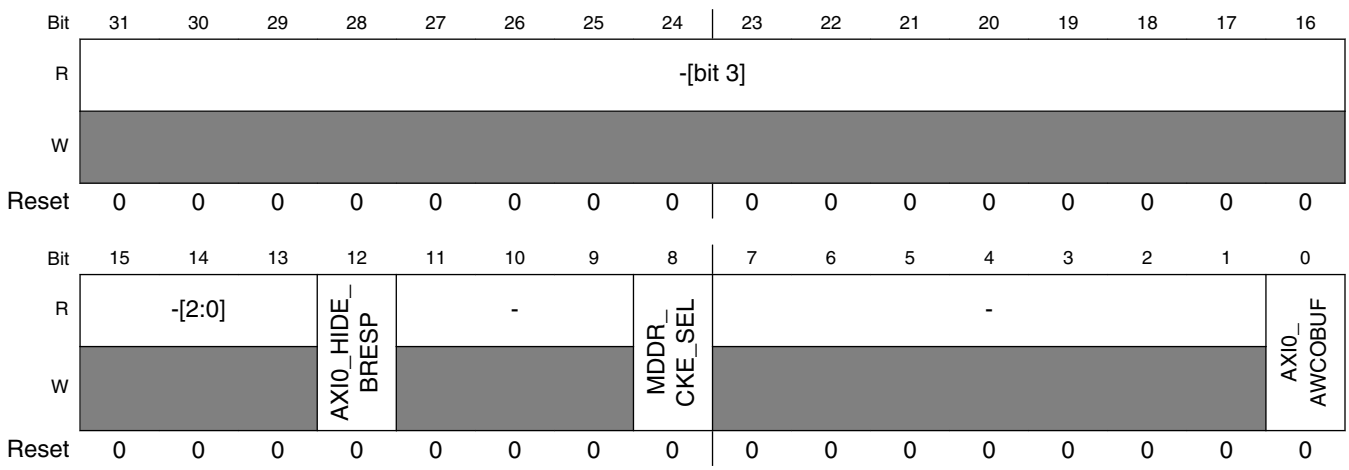
Table continues on the next page...

DRAM_CTL70 field descriptions (continued)

Field	Description
24 ODT_ALT_EN	User must set ODT_ALT_EN=0 for this MC. b0 = ODT support with CAS latency 3 is not supported. b1 = ODT support with CAS latency 3 is supported.
23–20 -	Reserved.
19–16 TDFI_DRAM_ CLK_ENABLE	Delay from DFI clock enable to memory clock enable.
15–11 -	Reserved.
10–8 TDFI_DRAM_ CLK_DISABLE	Delay from DFI clock disable to memory clock disable.
7–4 -	Reserved.
3–0 TDFI_CTRL_ DELAY	Delay from DFI command to memory command.

20.11.72 DRAM CTL Register 71 (DRAM_CTL71)

Address: DRAM_CTL71 is 1400_0000h base + 11Ch offset = 1400_011Ch



DRAM_CTL71 field descriptions

Field	Description
31–13 -	Reserved.

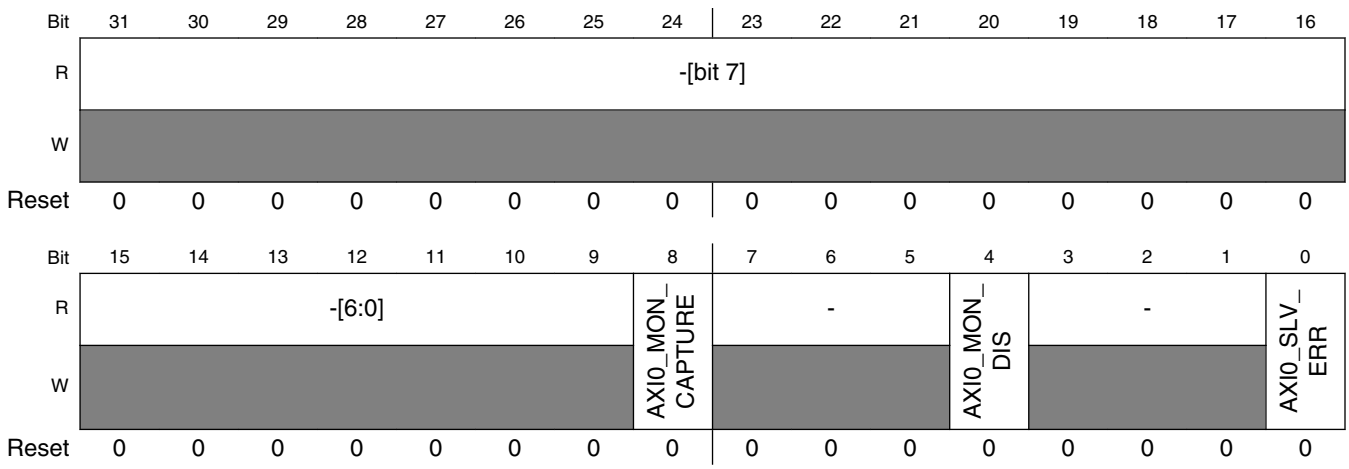
Table continues on the next page...

DRAM_CTL71 field descriptions (continued)

Field	Description
12 AXI0_HIDE_ BRESP	Hide extra latency of axi response.
11–9 -	Reserved.
8 MDDR_CKE_ SEL	Select the CKE init status for MDDR.
7–1 -	Reserved.
0 AXI0_ AWCOBUF	Coherent bufferable write command.

20.11.73 DRAM CTL Register 72 (DRAM_CTL72)

Address: DRAM_CTL72 is 1400_0000h base + 120h offset = 1400_0120h



DRAM_CTL72 field descriptions

Field	Description
31–9 -	Reserved.
8 AXI0_MON_ CAPTURE	Test purpose: capture debug data.
7–5 -	Reserved.
4 AXI0_MON_DIS	Test purpose: disable the axi0 monitor.

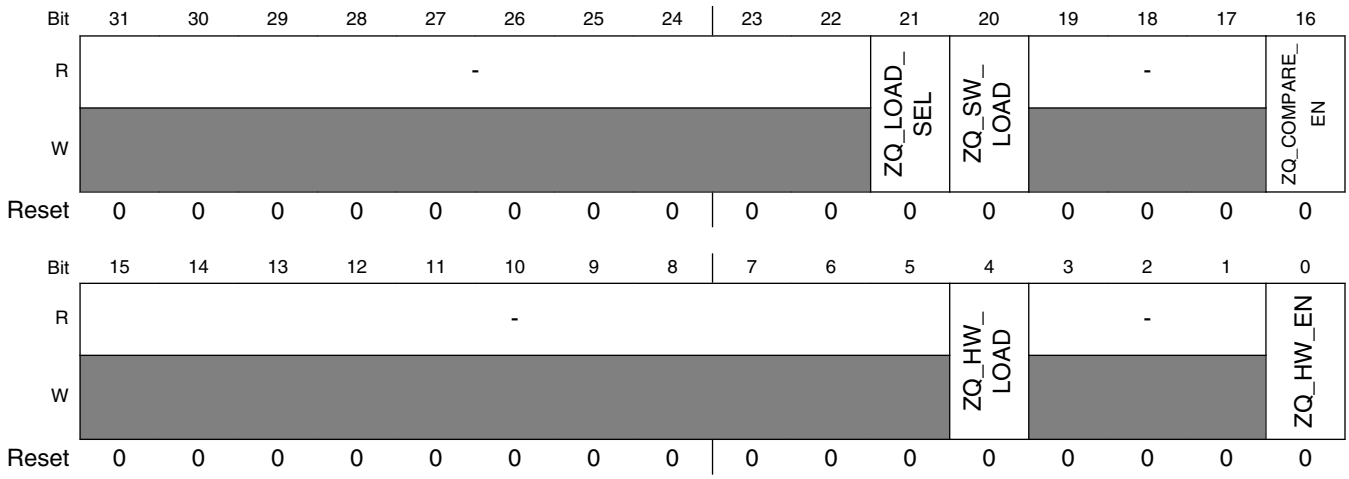
Table continues on the next page...

DRAM_CTL72 field descriptions (continued)

Field	Description
3–1 -	Reserved.
0 AXI0_SLV_ERR	Test purpose: force slave error on axi0.

20.11.74 DRAM CTL Register 73 (DRAM_CTL73)

Address: DRAM_CTL73 is 1400_0000h base + 124h offset = 1400_0124h



DRAM_CTL73 field descriptions

Field	Description
31–22 -	Reserved.
21 ZQ_LOAD_SEL	Select the mode of on-chip ZQ loading. b0 = Select hardware controlled ZQ loading mode b1 = Select software controlled ZQ loading mode
20 ZQ_SW_LOAD	When ZQ_LOAD_SEL=1, set this bit high loading ZQ values into the on-chip ZQ buffers.
19–17 -	Reserved.
16 ZQ_COMPARE_EN	Enable the on-chip ZQ comparator. It needs at least 300ns for the ZQ comparator to complete a comparsion. Please clear this bit before starting a new ZQ comparsion process. b0 = The on-chip ZQ comparator disabled b1 = The on-chip ZQ comparator enabled
15–5 -	Reserved.

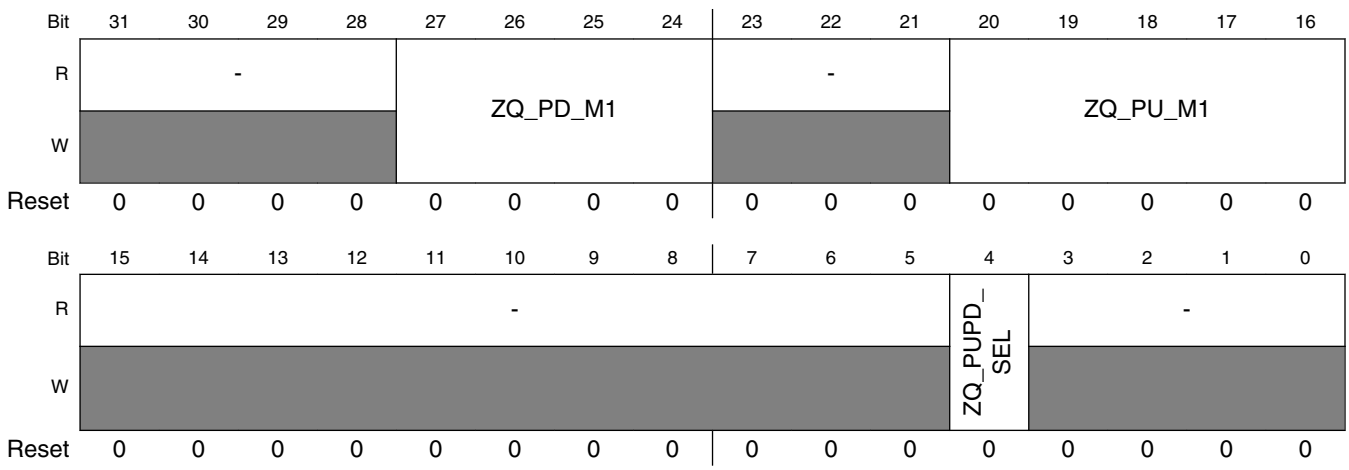
Table continues on the next page...

DRAM_CTL73 field descriptions (continued)

Field	Description
4 ZQ_HW_LOAD	When ZQ_LOAD_SEL=0 and ZQ_HW_EN=1, hardware start the ZQ loading routine on the posedge of ZQ_HW_LOAD.
3–1 -	Reserved.
0 ZQ_HW_EN	Enable the on-chip ZQ hardware controlled loading. The ZQ_PU/PD value would be loaded into the on-chip ZQ buffer on the next auto-refresh event.

20.11.75 DRAM CTL Register 74 (DRAM_CTL74)

Address: DRAM_CTL74 is 1400_0000h base + 128h offset = 1400_0128h



DRAM_CTL74 field descriptions

Field	Description
31–28 -	Reserved.
27–24 ZQ_PD_M1	The PD minus 1 value for on-chip ZQ. ZQ_PD_M1 = (ZQ_PD==0) ? 0 : (ZQ_PD - 1)
23–21 -	Reserved.
20–16 ZQ_PU_M1	The PU minus 1 value for on-chip ZQ. ZQ_PU_M1 = (ZQ_PU==0) ? 0 : (ZQ_PU - 1)
15–5 -	Reserved.
4 ZQ_PUPD_SEL	Select the PU or PD to be calibrated or loading buffer. b0 = Select PU calibration/loading mode b1 = Select PD calibration/loading mode

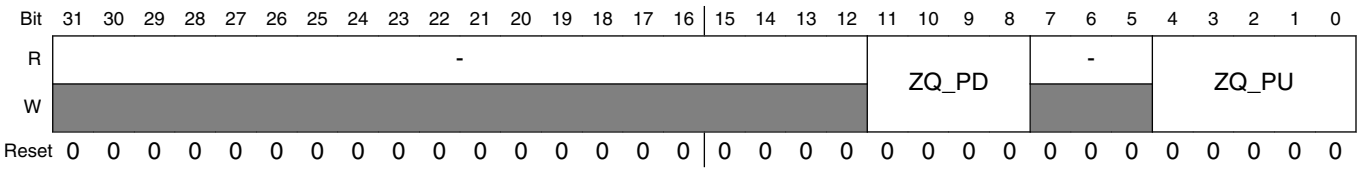
Table continues on the next page...

DRAM_CTL74 field descriptions (continued)

Field	Description
3–0 -	Reserved.

20.11.76 DRAM CTL Register 75 (DRAM_CTL75)

Address: DRAM_CTL75 is 1400_0000h base + 12Ch offset = 1400_012Ch

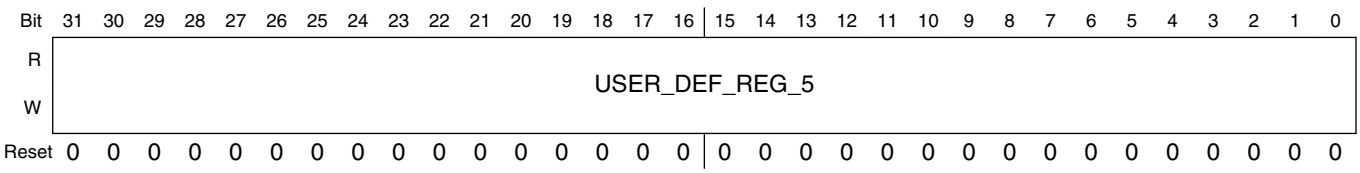


DRAM_CTL75 field descriptions

Field	Description
31–12 -	Reserved.
11–8 ZQ_PD	The PD value for on-chip ZQ.
7–5 -	Reserved.
4–0 ZQ_PU	The PU value for on-chip ZQ.

20.11.77 DRAM CTL Register 76 (DRAM_CTL76)

Address: DRAM_CTL76 is 1400_0000h base + 130h offset = 1400_0130h



DRAM_CTL76 field descriptions

Field	Description
31–0 USER_DEF_ REG_5	User-defined output register 5.

20.11.78 DRAM CTL Register 77 (DRAM_CTL77)

Address: DRAM_CTL77 is 1400_0000h base + 134h offset = 1400_0134h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_6																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL77 field descriptions

Field	Description
31–0 USER_DEF_ REG_6	User-defined output register 6.

20.11.79 DRAM CTL Register 78 (DRAM_CTL78)

Address: DRAM_CTL78 is 1400_0000h base + 138h offset = 1400_0138h

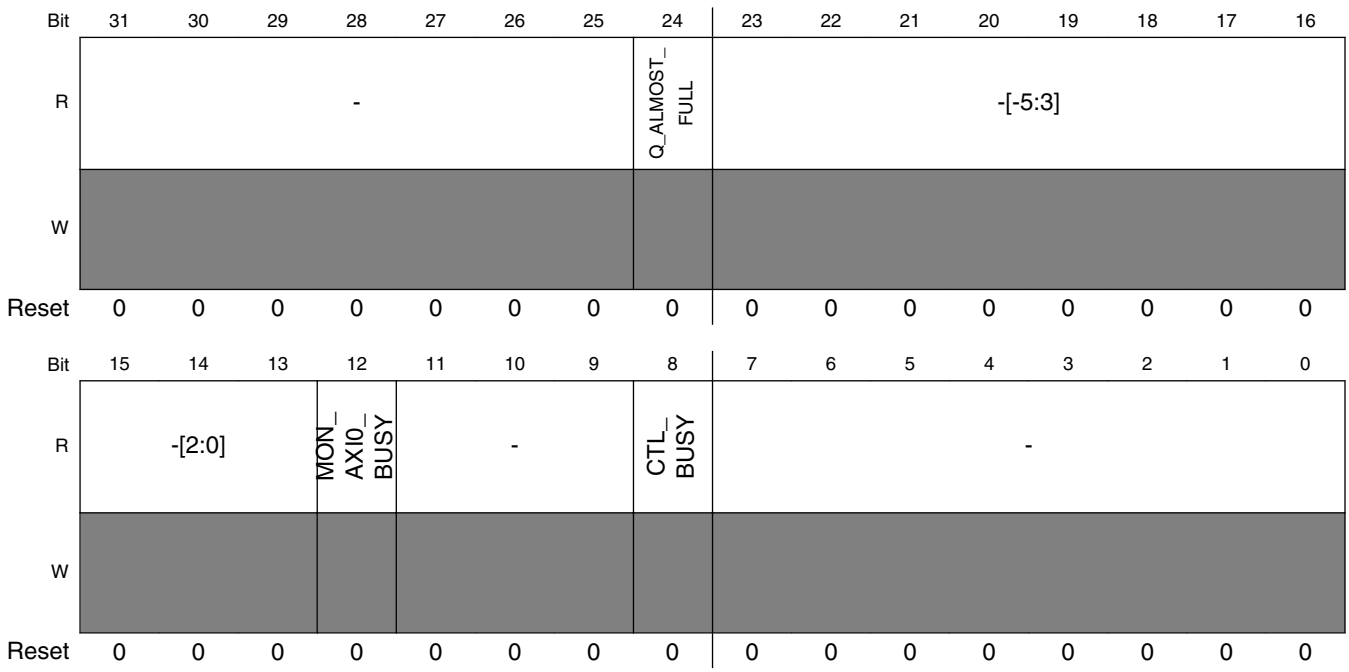
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_7																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL78 field descriptions

Field	Description
31–0 USER_DEF_ REG_7	User-defined output register 7.

20.11.80 DRAM CTL Register 79 (DRAM_CTL79)

Address: DRAM_CTL79 is 1400_0000h base + 13Ch offset = 1400_013Ch



DRAM_CTL79 field descriptions

Field	Description
31–25 -	Reserved.
24 Q_ALMOST_FULL	Indicate the command queue is almost full. READ-ONLY Refer to parameter Q_FULLNESS
23–13 -	Reserved.
12 MON_AXI0_BUSY	AXI0 port monitor busy. READ-ONLY
11–9 -	Reserved.
8 CTL_BUSY	Controller busy. READ-ONLY
7–0 -	Reserved.

20.11.81 DRAM CTL Register 80 (DRAM_CTL80)

Address: DRAM_CTL80 is 1400_0000h base + 140h offset = 1400_0140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	USER_DEF_REG_RO_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL80 field descriptions

Field	Description
31–0 USER_DEF_ REG_RO_1	User-defined input register 1. READ-ONLY

20.11.82 DRAM CTL Register 81 (DRAM_CTL81)

Address: DRAM_CTL81 is 1400_0000h base + 144h offset = 1400_0144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AXI0_MON_DBG0																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL81 field descriptions

Field	Description
31–0 AXI0_MON_ DBG0	Test purpose: debug data 0.

20.11.83 DRAM CTL Register 82 (DRAM_CTL82)

Address: DRAM_CTL82 is 1400_0000h base + 148h offset = 1400_0148h

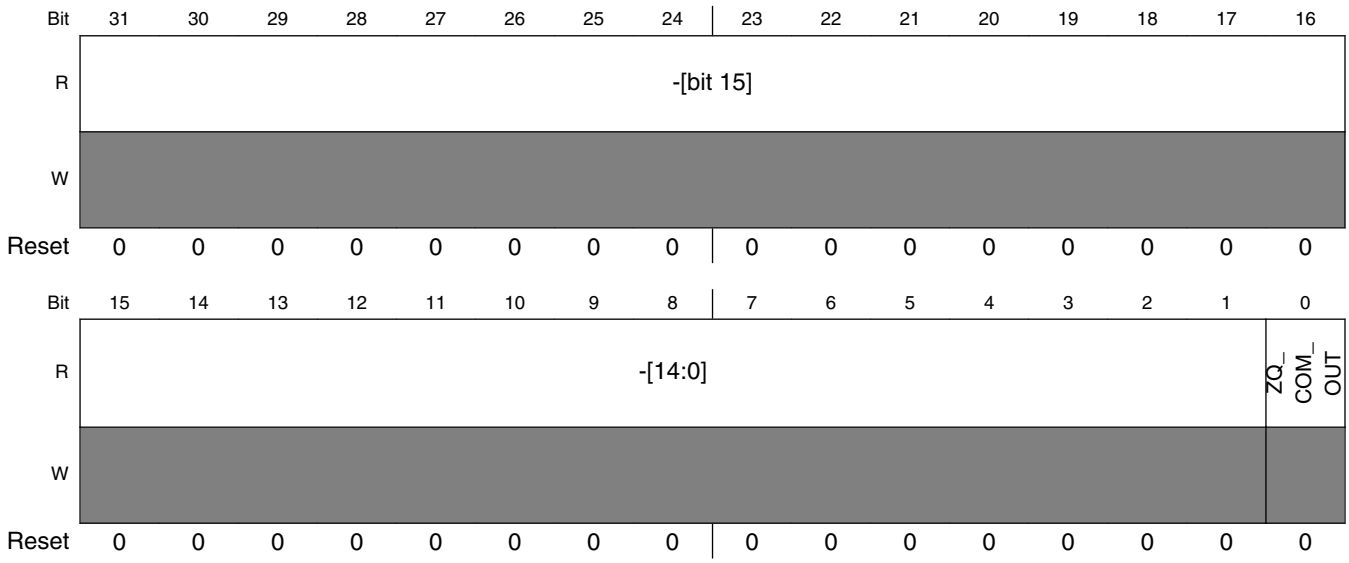
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AXI0_MON_DBG1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_CTL82 field descriptions

Field	Description
31–0 AXI0_MON_ DBG1	Test purpose: debug data 1.

20.11.84 DRAM CTL Register 83 (DRAM_CTL83)

Address: DRAM_CTL83 is 1400_0000h base + 14Ch offset = 1400_014Ch

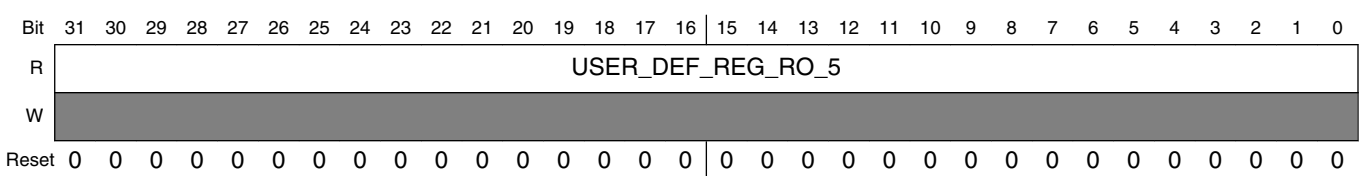


DRAM_CTL83 field descriptions

Field	Description
31–1 -	Reserved.
0 ZQ_COM_OUT	ZQ comparator output. READ-ONLY

20.11.85 DRAM CTL Register 84 (DRAM_CTL84)

Address: DRAM_CTL84 is 1400_0000h base + 150h offset = 1400_0150h

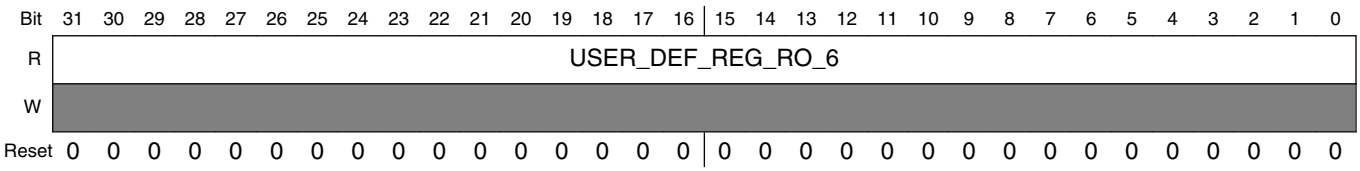


DRAM_CTL84 field descriptions

Field	Description
31–0 USER_DEF_ REG_RO_5	User-defined input register 5. READ-ONLY

20.11.86 DRAM CTL Register 85 (DRAM_CTL85)

Address: DRAM_CTL85 is 1400_0000h base + 154h offset = 1400_0154h

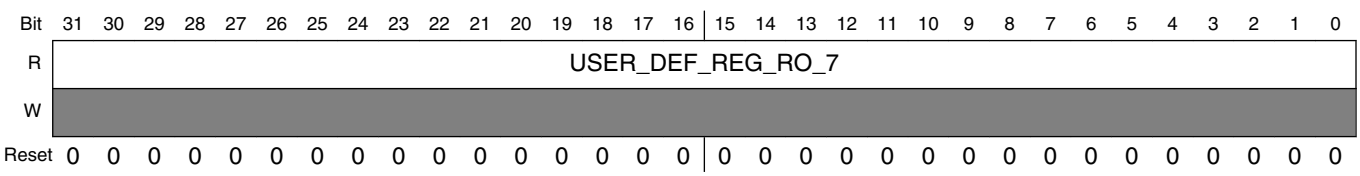


DRAM_CTL85 field descriptions

Field	Description
31–0 USER_DEF_ REG_RO_6	User-defined input register 6. READ-ONLY

20.11.87 DRAM CTL Register 86 (DRAM_CTL86)

Address: DRAM_CTL86 is 1400_0000h base + 158h offset = 1400_0158h

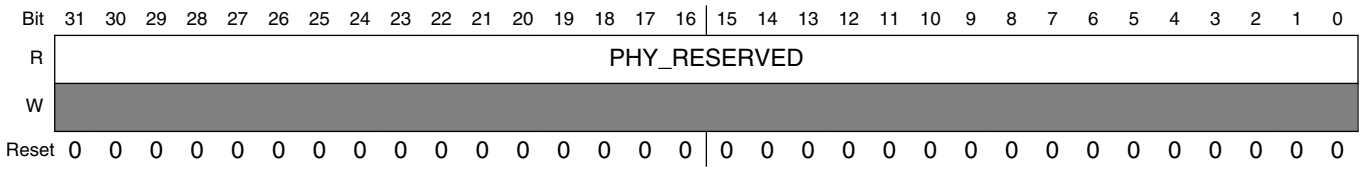


DRAM_CTL86 field descriptions

Field	Description
31–0 USER_DEF_ REG_RO_7	User-defined input register 7. READ-ONLY

20.11.88 DRAM PHY Register 00 (DRAM_PHY00)

Address: DRAM_PHY00 is 1400_0000h base + 200h offset = 1400_0200h

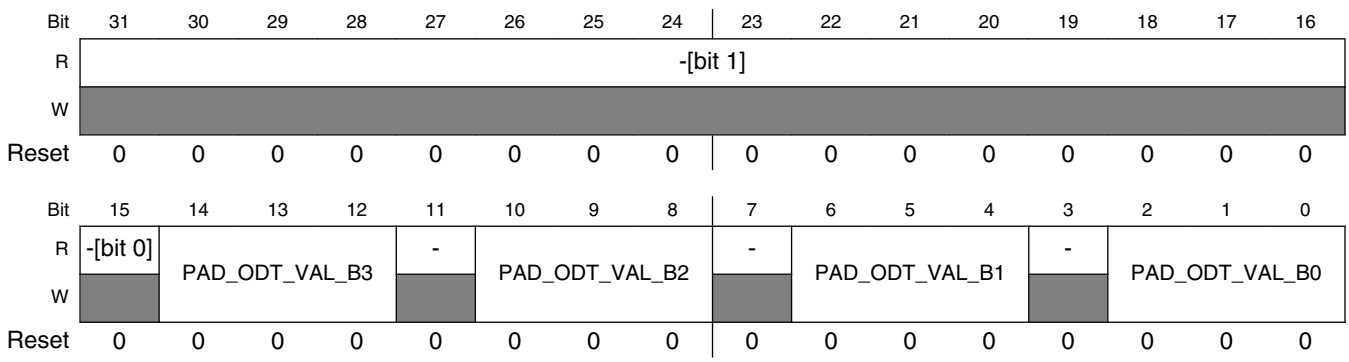


DRAM_PHY00 field descriptions

Field	Description
31–0 Reserved	This field is reserved. This parameter has no meaning for this PHY.

20.11.89 DRAM PHY Register 01 (DRAM_PHY01)

Address: DRAM_PHY01 is 1400_0000h base + 204h offset = 1400_0204h



DRAM_PHY01 field descriptions

Field	Description
31–15 -	Reserved.
14–12 PAD_ODT_ VAL_B3	DDR IO ODT settings for ddr data byte 3. Refer to PAD_ODT_VAL_B0 for detailed description.
11 -	Reserved.
10–8 PAD_ODT_ VAL_B2	DDR IO ODT settings for ddr data byte 2. Refer to PAD_ODT_VAL_B0 for detailed description.
7 -	Reserved.

Table continues on the next page...

DRAM_PHY01 field descriptions (continued)

Field	Description
6–4 PAD_ODT_ VAL_B1	DDR IO ODT settings for ddr data byte 1. Refer to PAD_ODT_VAL_B0 for detailed description.
3 -	Reserved.
2–0 PAD_ODT_ VAL_B0	DDR IO ODT settings for ddr data byte 0. Configure on-chip DDR IO in DDR2 ODT mode (ddr_sel=b00, refer to the IOMUXC chapter for ddr_sel definition): b000 = ODT disabled b001 = 150 Ohm ODT b010 = 75 Ohm ODT b011 = 50 Ohm ODT Note: It defines the ODT function of on-chip ddr IO, not the ODT function of external ddr devices. The ODT function of external ddr devices is controlled by the ODT pin.

20.11.90 DRAM PHY Register 02 (DRAM_PHY02)

Controls the PHY parameters for data slice 0 Alias - PHY_CTRL_REG_0_B0

Address: DRAM_PHY02 is 1400_0000h base + 208h offset = 1400_0208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DM_TSEL_EN	DQ_TSEL_EN	DQS_RATIO	ECHO_GATE_EN	-	RD_DLY_SEL			-			PAD_OE_POLARITY	-			ENABLE_HALF_CAS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DQS_OE_START				DQS_OE_END				-	DATA_OE_START			-	DATA_OE_END		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY02 field descriptions

Field	Description
31 DM_TSEL_EN	Enables on-chip pad ODT for the DM pads. b0 = Disabled b1 = Enabled

Table continues on the next page...

DRAM_PHY02 field descriptions (continued)

Field	Description
30 DQ_TSEL_EN	Enables on-chip pad ODT for the DQ pads. b0 = Disabled b1 = Enabled
29 DQS_RATIO	Controls on-chip pad ODT for the DM pads. b0 = Enabled b1 = Disabled
28 ECHO_GATE_EN	Echo gate control for data slice X. Default 0x0. b0 = Uses the dfi_rddata_en signal to create a gate. b1 = Creates an echo_gate signal.
27 -	Reserved.
26–24 RD_DLY_SEL	Defines the read data delay. Holds the number of cycles to delay the dfi_rddata_en signal prior to enabling the read FIFO. After this delay, the read pointers begin incrementing the read FIFO. Default 0x3. Changes to this field affect phy_ctrl_reg_2 [3:0]. If this field is increased, the value in phy_ctrl_reg_2 [3:0] must be increased by at least the same amount.
23–21 -	Reserved.
20 PAD_OE_POLARITY	Sets the pad output enable polarity. Default 0x0. Note: User must set PAD_OE_POLARITY = 0 for this MC. b0 = OEN pad b1 = OE pad
19–17 -	Reserved.
16 ENABLE_HALF_CAS	Subtracts 1/2 cycle from the DQS gate value programmed into phy_ctrl_reg_1_X [2:0] by 1/2 cycle. Default 0x0. This is used when the gate is being aligned to the first DQS, and then is removed to move the gate back into the center of the preamble. b0 = Do not adjust b1 = Adjust the DQS gate by 1/2 clock forward
15–12 DQS_OE_START	Adjusts the starting point of the DQS pad output enable window. Lower numbers pull the rising edge earlier in time, and larger numbers cause the rising edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. Default 0x2.
11–8 DQS_OE_END	Adjusts the ending point of the DQS pad output enable window. Lower numbers pull the falling edge earlier in time, and larger numbers cause the falling edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. This field must be set to at least the value of bits [14:12]+2 to prevent disabling the pad before the data is completely written. Default 0x7.
7 -	Reserved.
6–4 DATA_OE_START	Adjusts the starting point of the DQ pad output enable window. Lower numbers pull the rising edge earlier in time, and larger numbers cause the rising edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. Default 0x2.
3 -	Reserved.

Table continues on the next page...

DRAM_PHY02 field descriptions (continued)

Field	Description
2–0 DATA_OE_END	Adjusts the ending point of the DQ pad output enable window. Lower numbers pull the falling edge earlier in time, and larger numbers cause the falling edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. This field must be set to at least the value of bits [14:12]+2 to prevent disabling the pad before the data is completely written. Default 0x7.

20.11.91 DRAM PHY Register 03 (DRAM_PHY03)

Controls the PHY parameters for data slice 0 Alias - PHY_CTRL_REG_1_B0

Address: DRAM_PHY03 is 1400_0000h base + 20Ch offset = 1400_020Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TSEL_START				TSEL_END				DQS_TSEL_EN	TSEL_POLARITY	LPBK_ERR_CHECK	LPBK_FAIL_SEL	LPBK_CTRL		LPBK_INTERNAL	LPBK_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-	LPBK_ERR_DELAY			-			GATE_ERR_DELAY			GATE_CLOSE_CFG		-	GATE_CFG		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY03 field descriptions

Field	Description
31–28 TSEL_START	Defines the on-chip pad ODT enable time. Larger values add greater delay to when tsel turns on. Each bit changes the output enable time by a 1/2 cycle resolution.
27–24 TSEL_END	Defines the on-chip pad ODT disable time. Larger values increase the delay to when tsel turns off. Each bit changes the output enable time by a 1/2 cycle resolution.
23 DQS_TSEL_EN	Enables on-chip pad ODT for the DQS pads. b0 = Disabled b1 = Enabled
22 TSEL_ POLARITY	Controls the polarity of the tsel signal for the DQS and DM pads. b0 = Negative Polarity b1 = Positive Polarity
21 LPBK_ERR_ CHECK	Triggers a data return to the memory controller. b0 = No action b1 = Sends loopback data on the dfi_rddata signal.

Table continues on the next page...

DRAM_PHY03 field descriptions (continued)

Field	Description
20 LPBK_FAIL_SEL	Selects data output type for den_phy_obs_reg_0_X [23:8]. b0 = Return the expected data. b1 = Return the actual data.
19–18 LPBK_CTRL	Loopback control b00 = Normal operational mode b01 = lpbk_start; Enables loopback write mode b10 = lpbk_stop; Stop loopback to check the error register b11 = clear; Clear loopback registers
17 LPBK_ INTERNAL	Controls the loopback read multiplexer. b0 = External loopback b1 = Internal loopback
16 LPBK_EN	Controls the internal write multiplexer. b0 = Normal operation b1 = Enable loopback
15 -	Reserved.
14–12 LPBK_ERR_ DELAY	Sets the cycle delay between the LFSR and loopback error check logic. Note that h7 is not a valid selection and will result in a false passing result.
11–9 -	Reserved.
8–6 GATE_ERR_ DELAY	Defines the number of cycles of margin allowed for the final DQS falling edge of a read burst. An error will occur if the final DQS is not found after DQS_FALLING_EDGE_EXP_DLY and before (DQS_FALLING_EDGE_EXP_DLY+ gate_error_delay) cycles. If this field is set to a non-zero value and an error occurs, the PHY will automatically deassert (close) the read DQS gate and log the error in den_phy_obs_reg_0_X [24]. In the event of an error, the user should check the connections between the SOC and DRAM devices. This field can be useful for debugging the I/O during RTL integration. The default value is based on the delay information provided at configuration. A delayed version of the falling edge of dfi_rddata_en is used to determine when the final DQS falling edge is expected. The delay of the falling edge of dfi_rddata_en, in clock cycles is: DQS_FALLING_EDGE_EXP_DLY = (den_phy_ctrl_reg_1_X [2:0] + N*TCK), where N is 1/2 when enable_half_cas is off (den_phy_ctrl_reg_0_X [16] = b0), or 1 when enable_half_cas is on (den_phy_ctrl_reg_0_X [16] = b1) This field is only relevant when mobile support is disabled (den_phy_ctrl_reg_2 [6] = b0). Clearing this field to 0x0 disables error detection.
5–4 GATE_CLOSE_ CFG	Program either or both of these bits to b1 to extend the closing of the DQS gate by 1 cycle. This should be done when the round trip flight time of the DQS (IO_Pad_Delay + (Flight Time * 2) + IO_Input_Pad_Delay) + tDQSCK(max) is greater than 2 times the cycle time. Otherwise use b00.
3 -	Reserved.
2–0 GATE_CFG	Coarse adjust of gate open time. This value is the number of cycles to delay the dfi_rddata_en signal prior to opening the gate in full cycle increments. Decreasing this value pulls the gate earlier in time. This field, along with the den_phy_ctrl_reg_0_X [16] bit should be programmed such that the gate signal lands in the valid DQS gate window.

20.11.92 DRAM PHY Register 04 (DRAM_PHY04)

Address: DRAM_PHY04 is 1400_0000h base + 210h offset = 1400_0210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_CTRL_REG_0_B1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY04 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_0_B1	Controls the PHY parameters for data slice 1. Refer to PHY_CTRL_REG_0_B0

20.11.93 DRAM PHY Register 05 (DRAM_PHY05)

Address: DRAM_PHY05 is 1400_0000h base + 214h offset = 1400_0214h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_CTRL_REG_1_B1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY05 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_1_B1	Controls the PHY parameters for data slice 1. Refer to PHY_CTRL_REG_1_B0

20.11.94 DRAM PHY Register 06 (DRAM_PHY06)

Address: DRAM_PHY06 is 1400_0000h base + 218h offset = 1400_0218h

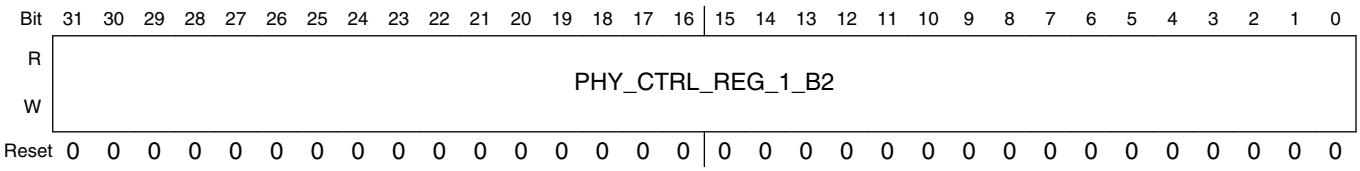
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_CTRL_REG_0_B2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY06 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_0_B2	Controls the PHY parameters for data slice 2. Refer to PHY_CTRL_REG_0_B0

20.11.95 DRAM PHY Register 07 (DRAM_PHY07)

Address: DRAM_PHY07 is 1400_0000h base + 21Ch offset = 1400_021Ch

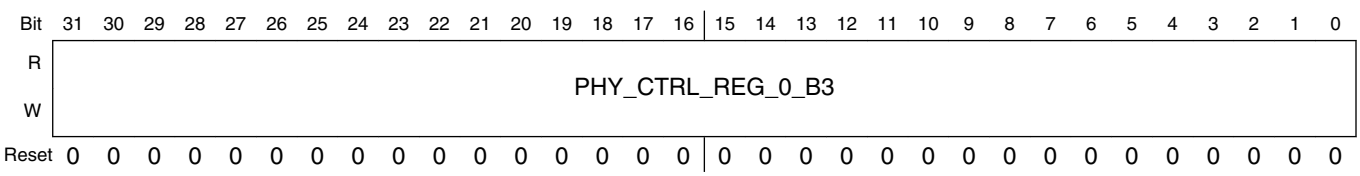


DRAM_PHY07 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_1_B2	Controls the PHY parameters for data slice 2. Refer to PHY_CTRL_REG_1_B0

20.11.96 DRAM PHY Register 08 (DRAM_PHY08)

Address: DRAM_PHY08 is 1400_0000h base + 220h offset = 1400_0220h



DRAM_PHY08 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_0_B3	Controls the PHY parameters for data slice 3. Refer to PHY_CTRL_REG_0_B0

20.11.97 DRAM PHY Register 09 (DRAM_PHY09)

Address: DRAM_PHY09 is 1400_0000h base + 224h offset = 1400_0224h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_CTRL_REG_1_B3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY09 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_1_B3	Controls the PHY parameters for data slice 3. Refer to PHY_CTRL_REG_1_B0

20.11.98 DRAM PHY Register 10 (DRAM_PHY10)

Address: DRAM_PHY10 is 1400_0000h base + 228h offset = 1400_0228h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_CTRL_REG_0_CA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY10 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_0_CA	Controls the PHY parameters for data slice CA. Refer to PHY_CTRL_REG_0_B0

20.11.99 DRAM PHY Register 11 (DRAM_PHY11)

Address: DRAM_PHY11 is 1400_0000h base + 22Ch offset = 1400_022Ch

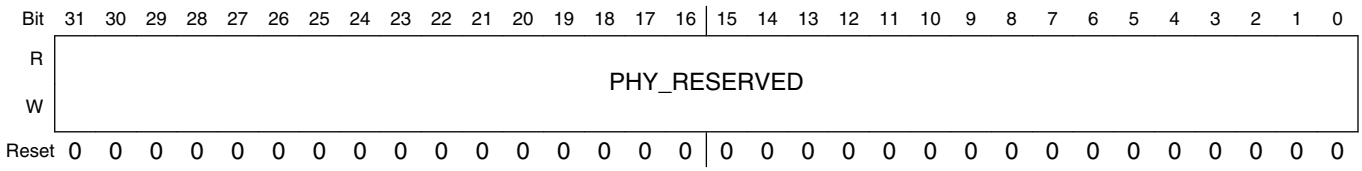
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_CTRL_REG_1_CA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY11 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_1_CA	Controls the PHY parameters for data slice CA. Refer to PHY_CTRL_REG_1_B0

20.11.100 DRAM PHY Register 12 (DRAM_PHY12)

Address: DRAM_PHY12 is 1400_0000h base + 230h offset = 1400_0230h

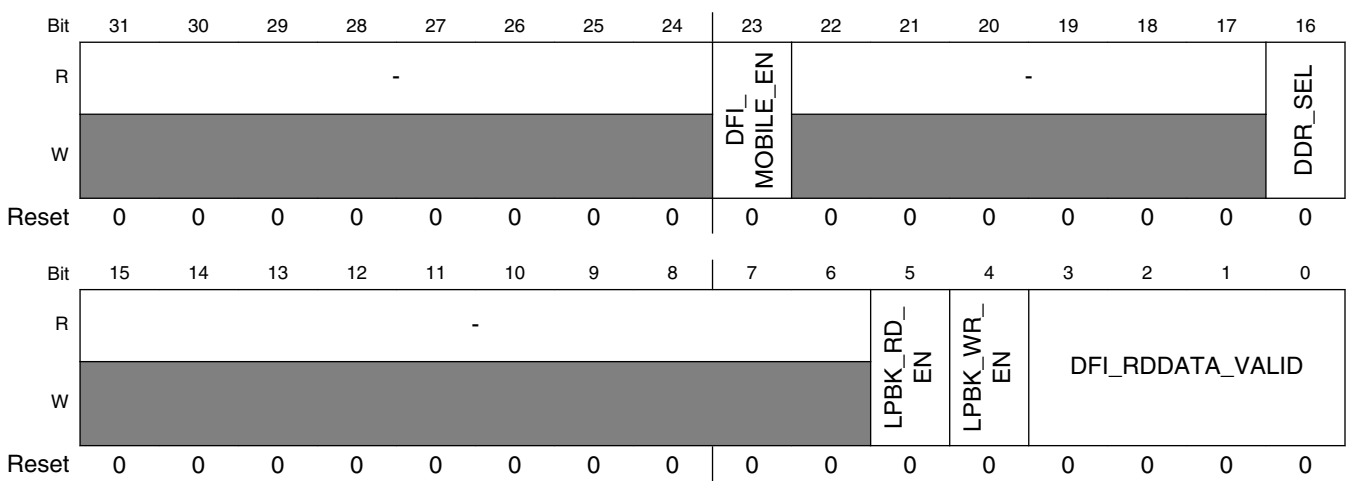


DRAM_PHY12 field descriptions

Field	Description
31–0 Reserved	This field is reserved. This parameter has no meaning for this PHY.

20.11.101 DRAM PHY Register 13 (DRAM_PHY13)

Address: DRAM_PHY13 is 1400_0000h base + 234h offset = 1400_0234h



DRAM_PHY13 field descriptions

Field	Description
31–24 -	Reserved.

Table continues on the next page...

DRAM_PHY13 field descriptions (continued)

Field	Description
23 DFI_MOBILE_EN	DFI control for Mobile ddr devices
22–17 -	Reserved.
16 DDR_SEL	Controls the PHY memory system mode. b0 = non-LPDDR2 mode b1 = LPDDR2 mode
15–6 -	Reserved.
5 LPBK_RD_EN	Enables the pad inputs specifically for external loopback. This bit is tied to the internal signal lpbk_rddata_en. b0 = Normal Operation b1 = Loopback Mode
4 LPBK_WR_EN	Enables the pad outputs specifically for external loopback. This bit is tied to the internal signal lpbk_wrdata_en. b0 = Normal Operation b1 = Loopback Mode
3–0 DFI_RDDATA_VALID	Sets the dfi_rddata_valid delay relative to dfi_rddata_en. At default, all den_phy_ctrl_reg_0_X [26:24] parameter fields have the same default values, and this field is set to den_phy_ctrl_reg_0_X [26:24] + 1. The valid ranges of this field are (phy_ctrl_reg_0_X[26:24] + 1) to MAX_VAL, using the largest value across all the slices.

20.11.102 DRAM PHY Register 14 (DRAM_PHY14)

Address: DRAM_PHY14 is 1400_0000h base + 238h offset = 1400_0238h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PHASE_DETECT_SEL			DLL_BYPASS_MODE	-				DLL_RD_DELAY_BYPASS[-7:1]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

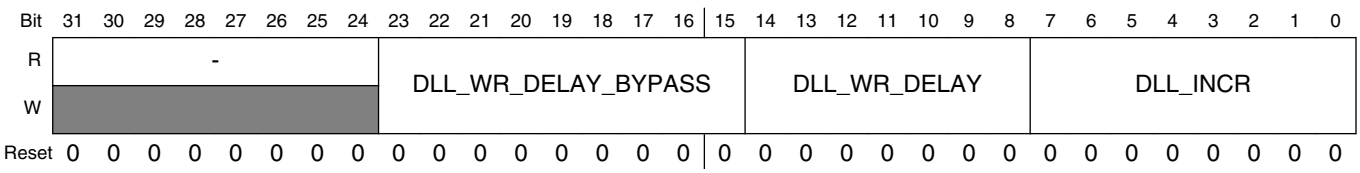
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_RD_DELAY_BYPASS[bit 0]	DLL_RD_DELAY							DLL_START_POINT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY14 field descriptions

Field	Description
31–29 PHASE_ DETECT_SEL	DLL Phase Detect Selector for the master delay line. Selects the number of delay elements to be inserted between the phase detect flip-flops. Default programming is 2 elements but if a lock condition is not detected, the user should increase the number of delay elements. b000 = 1 delay element b001 = 2 delay elements b010 = 3 delay elements b011 = 4 delay elements b100 = 5 delay elements b101 = 6 delay elements b110 = 7 delay elements b111 = 8 delay elements
28 DLL_BYPASS_ MODE	DLL bypass mode control b0 = Normal operational mode. Read DQS delay is phy_dll_ctrl_reg_0_X [14:8] and clk_wr delay is phy_dll_ctrl_reg_1_X [14:8]. b1 = Bypass Mode is on. Read DQS delay is phy_dll_ctrl_reg_0_X [23:15] and clk_wr delay is phy_dll_ctrl_reg_1_X [23:15].
27–24 -	Reserved.
23–15 DLL_RD_ DELAY_ BYPASS	Holds the read DQS delay for bypass mode. (phy_dll_ctrl_reg_0_X[28] = b1)This value directly controls the number delay elements that are active.
14–8 DLL_RD_DELAY	Holds the read DQS delay for normal mode. (phy_dll_ctrl_reg_0_X[28] = b0) Typically, this value is 1/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle.
7–0 DLL_START_ POINT	This value is loaded into the DLL at initialization and is the value at which the DLL will begin searching for a lock. This field must be set to a value greater than or equal to 4.

20.11.103 DRAM PHY Register 15 (DRAM_PHY15)

Address: DRAM_PHY15 is 1400_0000h base + 23Ch offset = 1400_023Ch



DRAM_PHY15 field descriptions

Field	Description
31–24 -	Reserved.
23–15 DLL_WR_ DELAY_ BYPASS	Holds the clk_wr delay setting when the DLL is operating in bypass mode. (phy_dll_ctrl_reg_0_X[28] = b1)
14–8 DLL_WR_ DELAY	Holds the clk_wr delay setting in normal mode. (phy_dll_ctrl_reg_0_X[28] = b0) Typically, this value is 3/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle.
7–0 DLL_INCR	DLL Increment Value. This sets the increment used by the DLL when searching for a lock. Denali recommends keeping this field small (around 0x4) to keep the steps gradual.

20.11.104 DRAM PHY Register 16 (DRAM_PHY16)

Address: DRAM_PHY16 is 1400_0000h base + 240h offset = 1400_0240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_0_B1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY16 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_0_B1	Controls the DLL parameters for data slice 1. Refer to DLL_CTRL_REG_0_B0

20.11.105 DRAM PHY Register 17 (DRAM_PHY17)

Address: DRAM_PHY17 is 1400_0000h base + 244h offset = 1400_0244h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_1_B1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY17 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_1_B1	Controls the DLL parameters for data slice 1. Refer to DLL_CTRL_REG_1_B0

20.11.106 DRAM PHY Register 18 (DRAM_PHY18)

Address: DRAM_PHY18 is 1400_0000h base + 248h offset = 1400_0248h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_0_B2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY18 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_0_B2	Controls the DLL parameters for data slice 1. Refer to DLL_CTRL_REG_0_B0

20.11.107 DRAM PHY Register 19 (DRAM_PHY19)

Address: DRAM_PHY19 is 1400_0000h base + 24Ch offset = 1400_024Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_1_B2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY19 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_1_B2	Controls the DLL parameters for data slice 1. Refer to DLL_CTRL_REG_1_B0

20.11.108 DRAM PHY Register 20 (DRAM_PHY20)

Address: DRAM_PHY20 is 1400_0000h base + 250h offset = 1400_0250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_0_B3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY20 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_0_B3	Controls the DLL parameters for data slice 1. Refer to DLL_CTRL_REG_0_B0

20.11.109 DRAM PHY Register 21 (DRAM_PHY21)

Address: DRAM_PHY21 is 1400_0000h base + 254h offset = 1400_0254h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_1_B3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY21 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_1_B3	Controls the DLL parameters for data slice 1. Refer to DLL_CTRL_REG_1_B0

20.11.110 DRAM PHY Register 22 (DRAM_PHY22)

Address: DRAM_PHY22 is 1400_0000h base + 258h offset = 1400_0258h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_0_CA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY22 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_0_CA	Controls the DLL parameters for data slice CA. Refer to DLL_CTRL_REG_0_B0

20.11.111 DRAM PHY Register 23 (DRAM_PHY23)

Address: DRAM_PHY23 is 1400_0000h base + 25Ch offset = 1400_025Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_1_CA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY23 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_1_CA	Controls the DLL parameters for data slice CA. Refer to DLL_CTRL_REG_1_B0

20.11.112 DRAM PHY Register 24 (DRAM_PHY24)

Address: DRAM_PHY24 is 1400_0000h base + 260h offset = 1400_0260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-							LPBK_ ERR_OUT	LPBK_DQ_DATA[0:8]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LPBK_DQ_DATA[7:0]								LPBK_DM_DATAT				-		LPBK_ STATUS	LPBK_ START
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY24 field descriptions

Field	Description
31–25 -	Reserved.
24 LPBK_ERR_OUT	Status signal to indicate that the logic gate had to be forced closed. b0 = Normal operation b1 = Gate close was forced
23–8 LPBK_DQ_DATA	Reports the actual data or expected data, depending on the setting of the den_phy_ctrl_reg_1_X[20] parameter bit.
7–4 LPBK_DM_DATAT	Reports the actual data mask or the expected data mask, depending on the setting of the den_phy_ctrl_reg_1_X[20] parameter bit.
3–2 -	Reserved.
1 LPBK_STATUS	Reports status of loopback errors. b0 = Last Loopback test had no errors. b1 = Last Loopback test contained data errors
0 LPBK_START	Defines the status of the loopback mode. b0 = Not in loopback mode b1 = In Loopback mode

20.11.113 DRAM PHY Register 25 (DRAM_PHY25)

Address: DRAM_PHY25 is 1400_0000h base + 264h offset = 1400_0264h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	dll_lock_val[bit 15]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

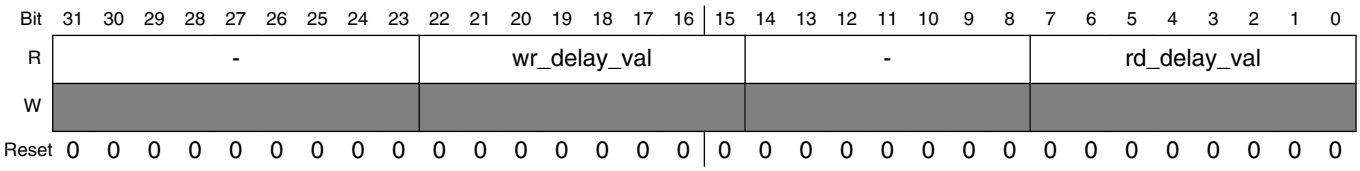
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	dll_lock_val[14:0]															dll_lock
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY25 field descriptions

Field	Description
31–1 dll_lock_val	Reports the DLL encoder value from the master DLL to the slave DLLs. The slaves use this value to set up their delays for the clk_wr and read DQS signals.
0 dll_lock	Indicates status of the DLL.

20.11.114 DRAM PHY Register 26 (DRAM_PHY26)

Address: DRAM_PHY26 is 1400_0000h base + 268h offset = 1400_0268h

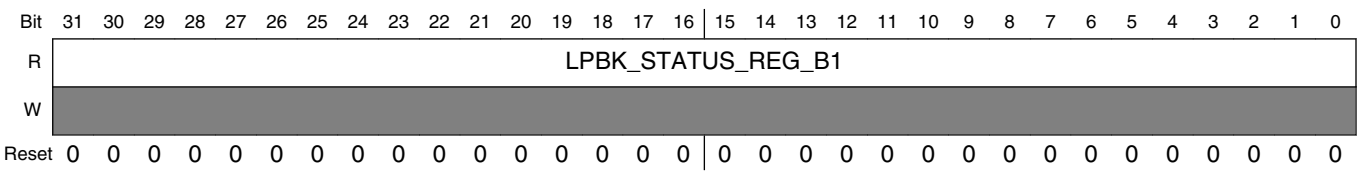


DRAM_PHY26 field descriptions

Field	Description
31–23 -	Reserved.
22–15 wr_delay_val	Holds the encoded value for the clk_wr delay line for this slice.
14–8 -	Reserved.
7–0 rd_delay_val	Holds the encoded value for the read delay line for this slice.

20.11.115 DRAM PHY Register 27 (DRAM_PHY27)

Address: DRAM_PHY27 is 1400_0000h base + 26Ch offset = 1400_026Ch

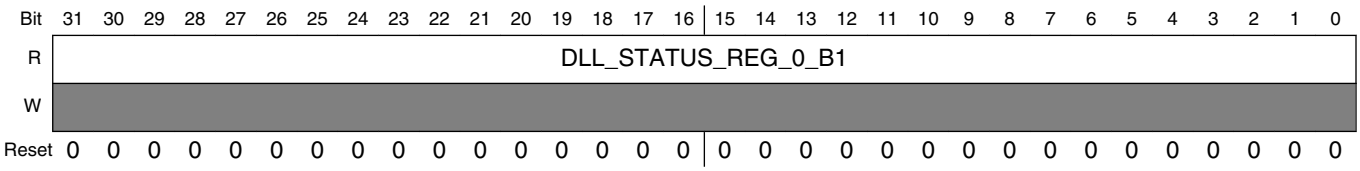


DRAM_PHY27 field descriptions

Field	Description
31–0 LPBK_STATUS_ REG_B1	Reports the Loopback status for data slice 1. Refer to LPBK_STATUS_REG_B0.

20.11.116 DRAM PHY Register 28 (DRAM_PHY28)

Address: DRAM_PHY28 is 1400_0000h base + 270h offset = 1400_0270h

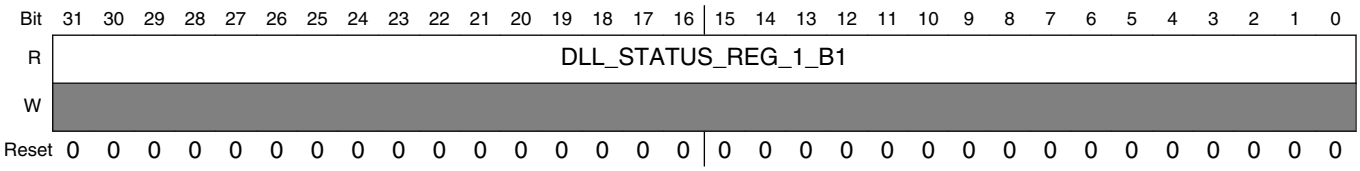


DRAM_PHY28 field descriptions

Field	Description
31–0 DLL_STATUS_ REG_0_B1	Reports the DLL status for data slice 1. Refer to DLL_STATUS_REG_0_B0.

20.11.117 DRAM PHY Register 29 (DRAM_PHY29)

Address: DRAM_PHY29 is 1400_0000h base + 274h offset = 1400_0274h

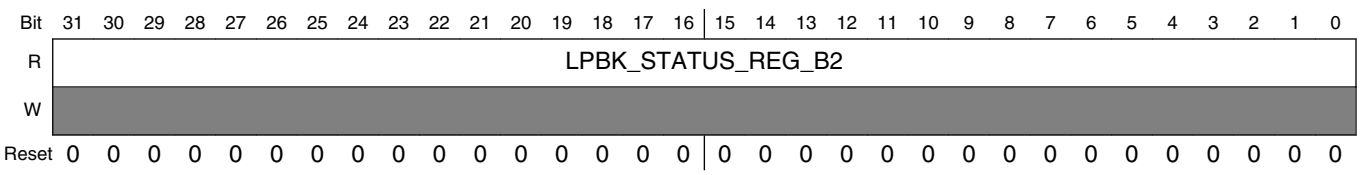


DRAM_PHY29 field descriptions

Field	Description
31–0 DLL_STATUS_ REG_1_B1	Reports the DLL status for data slice 1. Refer to DLL_STATUS_REG_1_B0.

20.11.118 DRAM PHY Register 30 (DRAM_PHY30)

Address: DRAM_PHY30 is 1400_0000h base + 278h offset = 1400_0278h

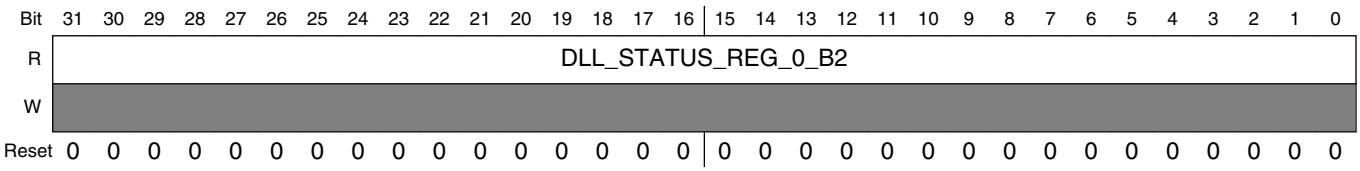


DRAM_PHY30 field descriptions

Field	Description
31–0 LPBK_STATUS_ REG_B2	Reports the Loopback status for data slice 2. Refer to LPBK_STATUS_REG_B0.

20.11.119 DRAM PHY Register 31 (DRAM_PHY31)

Address: DRAM_PHY31 is 1400_0000h base + 27Ch offset = 1400_027Ch

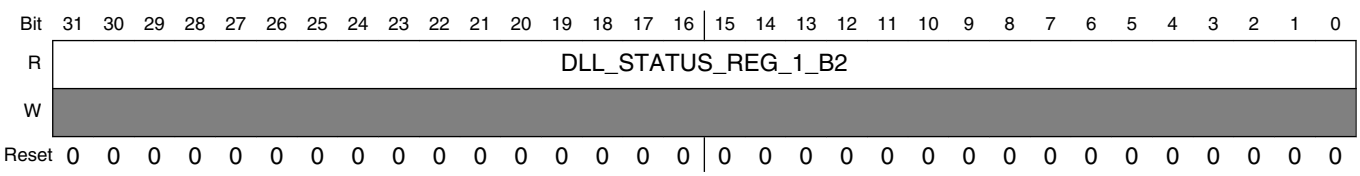


DRAM_PHY31 field descriptions

Field	Description
31–0 DLL_STATUS_ REG_0_B2	Reports the DLL status for data slice 2. Refer to DLL_STATUS_REG_0_B0.

20.11.120 DRAM PHY Register 32 (DRAM_PHY32)

Address: DRAM_PHY32 is 1400_0000h base + 280h offset = 1400_0280h



DRAM_PHY32 field descriptions

Field	Description
31–0 DLL_STATUS_ REG_1_B2	Reports the DLL status for data slice 2. Refer to DLL_STATUS_REG_1_B0.

20.11.121 DRAM PHY Register 33 (DRAM_PHY33)

Address: DRAM_PHY33 is 1400_0000h base + 284h offset = 1400_0284h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LPBK_STATUS_REG_B3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY33 field descriptions

Field	Description
31–0 LPBK_STATUS_REG_B3	Reports the Loopback status for data slice 3. Refer to LPBK_STATUS_REG_B0.

20.11.122 DRAM PHY Register 34 (DRAM_PHY34)

Address: DRAM_PHY34 is 1400_0000h base + 288h offset = 1400_0288h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_STATUS_REG_0_B3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY34 field descriptions

Field	Description
31–0 DLL_STATUS_REG_0_B3	Reports the DLL status for data slice 3. Refer to DLL_STATUS_REG_0_B0.

20.11.123 DRAM PHY Register 35 (DRAM_PHY35)

Address: DRAM_PHY35 is 1400_0000h base + 28Ch offset = 1400_028Ch

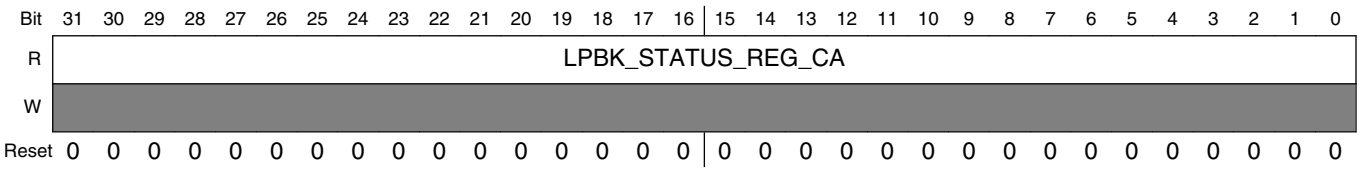
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_STATUS_REG_1_B3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DRAM_PHY35 field descriptions

Field	Description
31–0 DLL_STATUS_ REG_1_B3	Reports the DLL status for data slice 3. Refer to DLL_STATUS_REG_1_B0.

20.11.124 DRAM PHY Register 36 (DRAM_PHY36)

Address: DRAM_PHY36 is 1400_0000h base + 290h offset = 1400_0290h

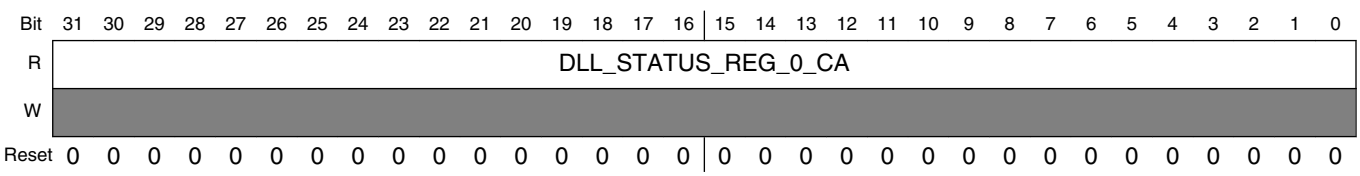


DRAM_PHY36 field descriptions

Field	Description
31–0 LPBK_STATUS_ REG_CA	Reports the Loopback status for data slice CA. Refer to LPBK_STATUS_REG_B0.

20.11.125 DRAM PHY Register 37 (DRAM_PHY37)

Address: DRAM_PHY37 is 1400_0000h base + 294h offset = 1400_0294h

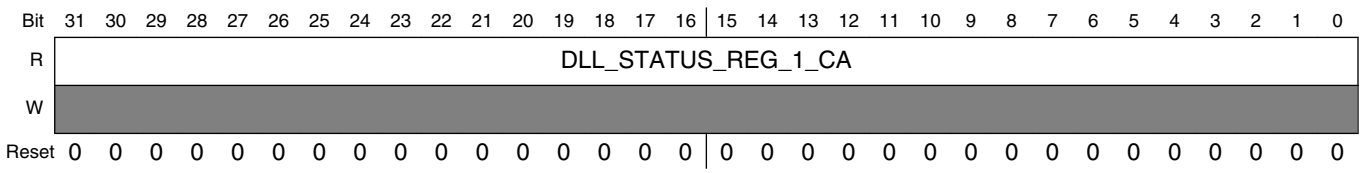


DRAM_PHY37 field descriptions

Field	Description
31–0 DLL_STATUS_ REG_0_CA	Reports the DLL status for data slice CA. Refer to DLL_STATUS_REG_0_B0.

20.11.126 DRAM PHY Register 38 (DRAM_PHY38)

Address: DRAM_PHY38 is 1400_0000h base + 298h offset = 1400_0298h



DRAM_PHY38 field descriptions

Field	Description
31–0 DLL_STATUS_ REG_1_CA	Reports the DLL status for data slice CA. Refer to DLL_STATUS_REG_1_B0.

Chapter 21
Dynamic Voltage and Frequency Scaling Core (DVFSC)

21.1 Introduction

The DVFSC allows simple dynamic voltage frequency scaling. The frequency of the core clock domain and the voltage of the core power domain can be changed on the fly while all blocks (including the ARM platform) continue their normal operation. The frequency of the core clock domain can be changed by switching temporally to an alternate PLL clock, and then get back to the updated PLL, already locked at a specific frequency, or by merely changing the post dividers division factors.

The DVFSC load tracking block allows hardware tracking on the core load and a generation of an interrupt when a frequency change is requested.

DVFSC is a sub-block of the General Power Controller (GPC). See GPC chapter for more details.

NOTE

DVFSC is a monitor that only provides an interrupt when counting exceeds the predefined value and doesn't actually send request to make a change a change of voltage and frequency. This can be done by the user in the interrupt routine or smart direct memory access (SDMA) code by using GPC or clock control module (CCM) (relocking the PLL or changing the post dividers at the CCM).

21.1.1 Overview

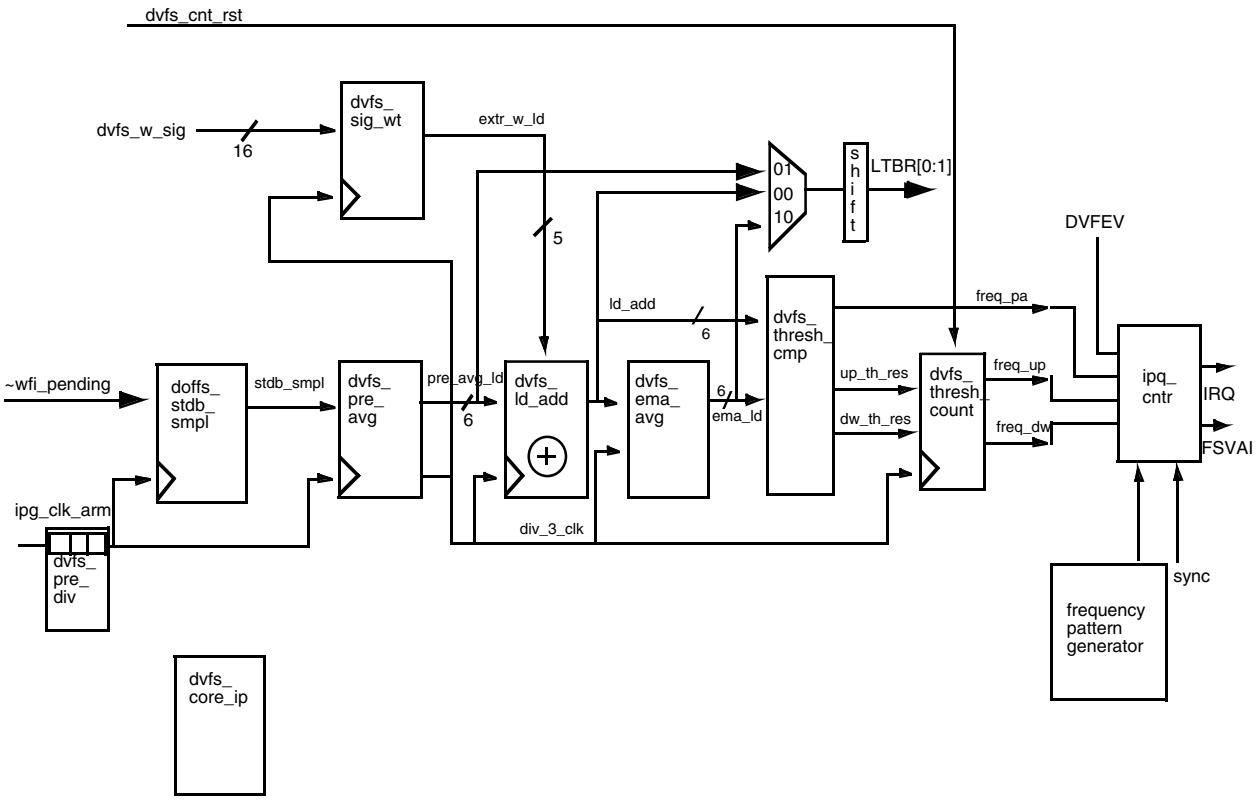


Figure 21-1. Block Diagram

21.1.2 Features

The DVFSC load tracking sub-block includes the following features:

- Configurable include/exclude of input signals:
  - ARM standby signal (idle / non-idle)
  - 16 general purpose bits
    - Configurable weight of each bit.
- Configurable generated clocks and averaging time slicing (respond time).
- Configurable panic mode respond logic (for frequency up).
- Programmable buffer for last 4,8,12, or 16 load tracking samples. Based on value in LBCF in DVFSC_CNTR. There is also a buffer full signal - LBFL.

21.2 Functional Description of DVFS Load Tracking

The DVFS load tracking sub-block includes the following features:

Configurable include/exclude of input signals:

- ARM standby signal (idle / non-idle)
- 16 general purpose load tracking signals
- Configurable weight and (level-sensitive) of GeP signals.
- Configurable generated clocks and averaging time slicing (respond time).
- Configurable panic mode respond logic (for frequency up).
- Programmable buffer for last 4,8,12, or 16 load tracking samples. Two Load Tracking Buffers are working in ping-pong mode with two buffer full signals - LBFL1, LBFL0.

21.3 Component Blocks Description

21.3.1 dvfs_stdb_smpl Block

This block samples the ccm_arm_stdb signal (ARM11 STANDBYWFI signal - idle state indicating) according to the edge of the ipg_clk_arm (ARM11 system clock) signal.

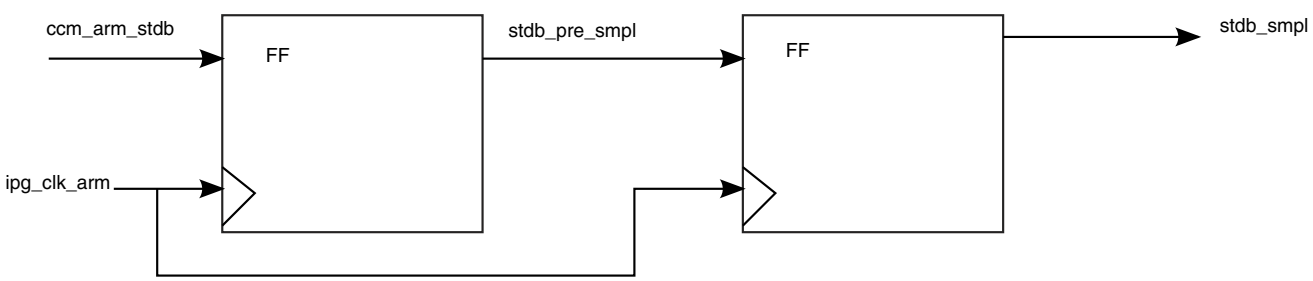


Figure 21-2. dvfs_stdb_smpl block diagram

This block synchronizes the ccm_arm_stdb signal with the ipg_clk_arm clock. The two signals enter the Flip-Flops (twice), and by doing so, are synchronized. The resulting synchronized signal is stdb_smpl.

21.3.2 dvfs_sig_wt Block

The purpose of this block is to sample the 16 GeP (general purpose) load signals, multiply each one of them by appropriate weight and sum products. The sampling is done by the slow clock div_3_clk. These signals have the only options of detection: level sampling.

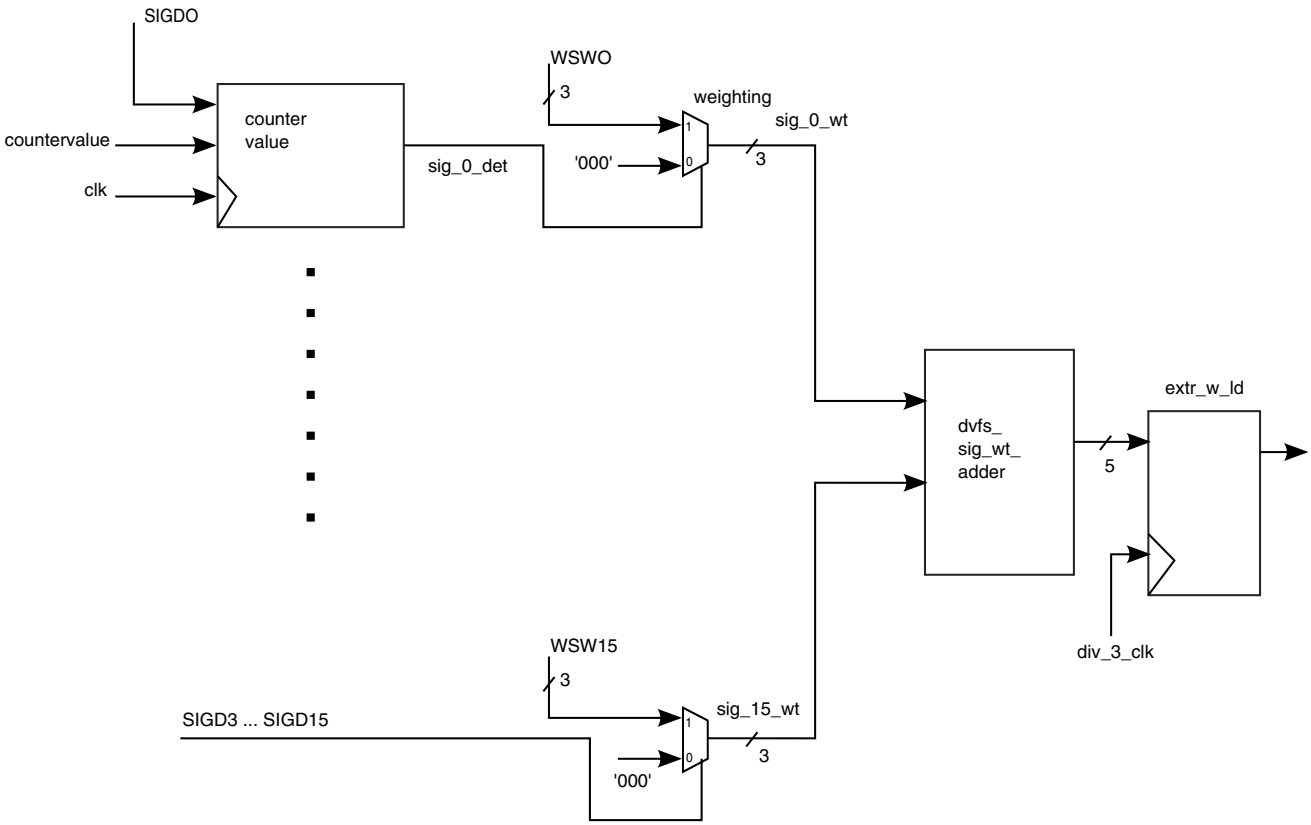


Figure 21-3. dvfs_sig_wt block diagram

There are two paths for weighted general purpose load signals to be calculated.

21.3.3 dvfs_pre_avg Block

The purpose of the dvfs_pre_avg block is to perform simple, non-overlapping averaging, reducing the sampling clock frequency and providing a level-based average index of the tracked ARM Platform load.

External signals:

- stdb_sampl - (input) sampled ARM11 standby signal
- ipg_clk_arm - (input) ARM11 system clk
- pre_avg_ld[4..0] - (output) averaged load

- div_3_clk - (output) clock, generated (reduced) from div_2_clk
- div_2_clk - (output) clock, generated (reduced) from div_1_clk
- div_1_clk - (output) clock, generated (reduced) from ipg_clk_arm

In the current implementation, the averaging is performed in three stages:

1. The first stage IDLE is sampled by divided sys_clock.
2. The second stage (counter2) performs an averaging operation with constant parameters. The counter is an nine (9) bit adder whose output is sampled by the div_2_clk clock. Five (5) highest bits of the sum_2 signal are passed to counter3 (equal to shift right operation).
3. The third (last) stage (counter3) performs an averaging operation with configurable parameters, set by the DIV3CK. The counter3 cell is a fifteen(15) bit adder with output sampled by div_3_clk. The Pre_avg_shifter_5 cell passes configurable bits from the sum_3 signal (see table1_3).

The output 5 bits avg_load signal provides a result with a tolerance of ~3%. (supported values range is 0-0.97)

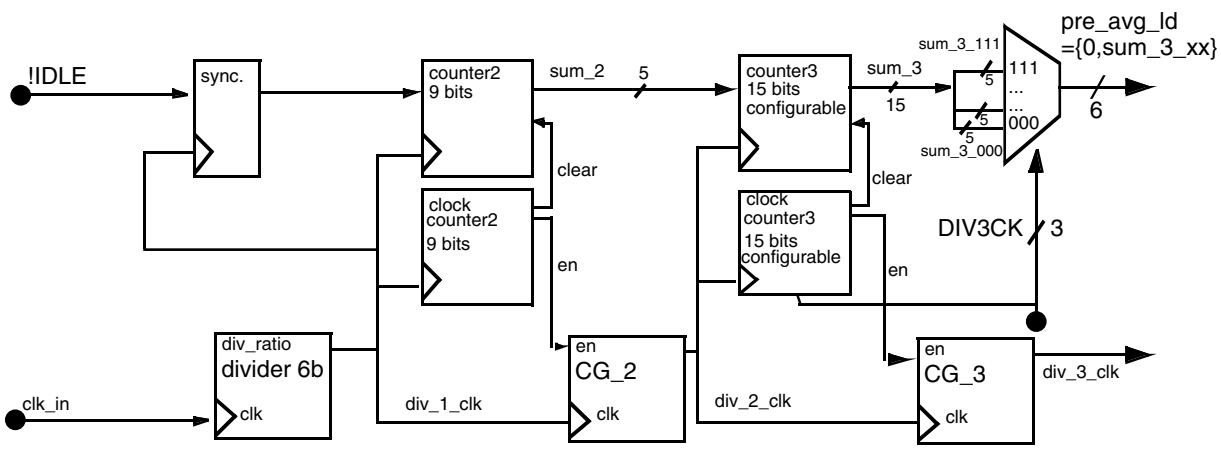


Figure 21-4. dvfs_pre_avg block diagram

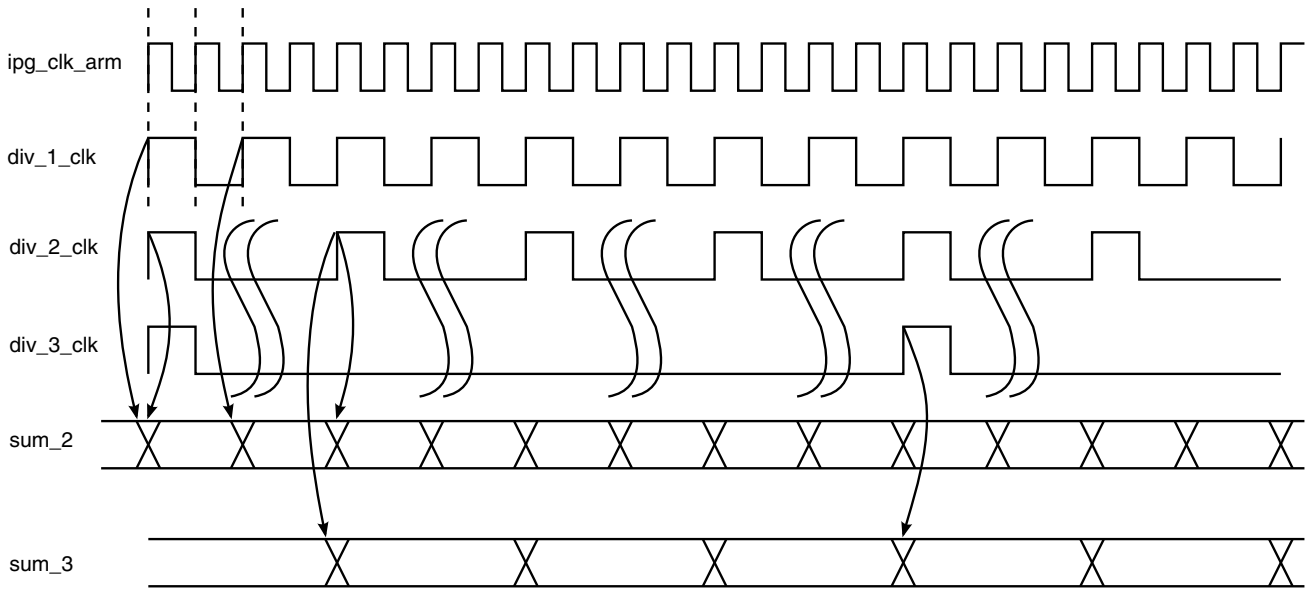


Figure 21-5. dvfs_pre_avg Clocks

The internal clocks in dvfs_avg_pre block are generated from ipg_clk_arm. The details are in the tables below:

Table 21-1. dvfs_pre_avg Signals and its Values

signal	binary max value	binary min value	decimal max value	decimal min value
sum_2	11111	0000	31	0
sum_3	111'1100'0000'0000	000'0000'0000'0000	31744	0
pre_avg_load	11111	0000	31	0

Table 21-2. dvfs_pre_avg generated clocks

clock name	generated from	ratio to source	ratio to ipg_clk_arm	max clk freq.	note
ipg_clk_arm	N/A	N/A	1	66MHz	source clock
div_1_clk	ipg_clk_arm	configurable	configurable	66MHz	configurable
div_2_clk (gated)	div_1_clk	512	configurable	128KHz	none
div_3_clk (gated)	div_2_clk	configurable	configurable	128KHz	configurable

Table 21-3. iv_3_clk Configurable Averaging

DIV3CK setting	dividing ratio	sum_3 passing bits	div_1_clk cumulative divider
00	1	4-0	1*512=512

Table continues on the next page...

Table 21-3. iv_3_clk Configurable Averaging (continued)

DIV3CK setting	dividing ratio	sum_3 passing bits	div_1_clk cumulative divider
001	4	6-2	4*512=2048
010	16	8-4	16*512=8192
011	64	10-6	64*512=32768
100	256	12-8	256*512=131072
101	1024	16-10	1024*512=524288

21.3.4 dvfs_ld_add Block

The dvfs_ld_add block sums the ARM Platform load, tracked by idle/non-idle signal and the load, detected from the additional load signals, weighted by signal_weighting block. The adder should perform the following operation:

extr_w_ld[4..0] + {0,pre_avg_ld[4..0]}

The input signals of five (5) bits produce output signal of six (6) bits, providing 3% resolution in the range of 0-1.97 of load indication. (1 is equal to 100% load tracking with no additional signals include). The output ld_add[5..0] is sampled by div_3_clk signal.

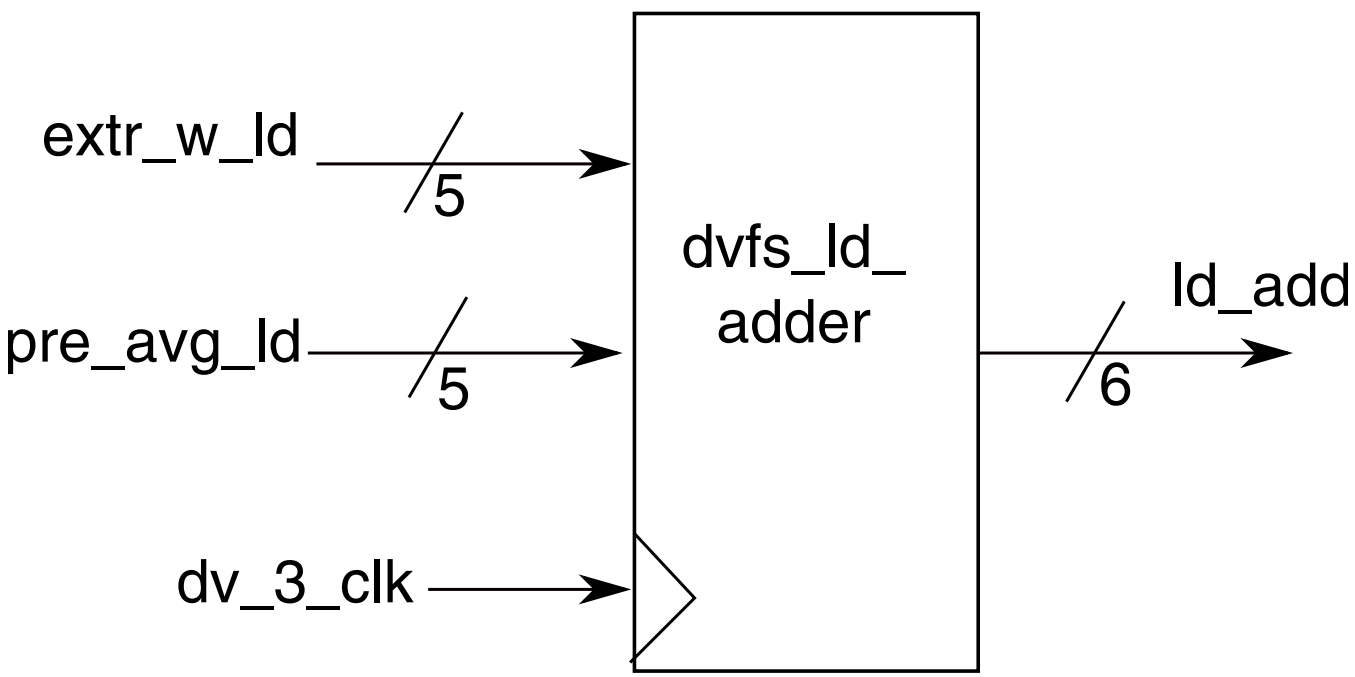


Figure 21-6. dvfs_ld_add block diagram

21.3.5 dvfs_ema_avg Block

The purpose of dvfs_ema_avg (EMA - Exponential Moving Average) block is to calculate an exponential moving average of the tracked ARM Platform load. The parameters of EMA are defined by EMAC field of the DVFSC_EMAC register. These parameters set how many samples of simple average will be taken into account.

The EMA formula is: EMA(i) = a*X(i) + (1-a)*EMA(i-1)

where:

a = 2 / (N+1);

N is the number of samples taken into account.

X(i) = current input sample;

EMA(i-1) = previous value of EMA.

By setting the value of 'a', the behavior of EMA is defined.

In the following table, the parameter 'a' is listed relative to the amount of the X(i) samples taken into account (('N')) in the equation above. (The resolution of the digital multiplier is limited, hence the lowest values of the 'a' can be linked to a range of included samples in EMA instead of single number.)

Also included in the following table is the amount of cycles in which the lower frequency will be masked from the moment the DVFS is enabled (and not between frequency switches), so that enough information about the history can be gained before the frequency is lowered.

EMA settings

samples included in EMA calculation (N)	Number of cycles while lower frequency interrupt is masked	'a' value (decimal)	'a' value, adjusted by binary resolution	'a' value (binary)								
----------		----------	----------	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 8
1	6	1.000	1.000	1	0	0	0	0	0	0	0	0
2	11	0.667	0.668	0	1	0	1	0	1	0	1	1
3	11	0.500	0.500	0	1	0	0	0	0	0	0	0
4	22	0.400	0.398	0	0	1	1	0	0	1	1	0
5	22	0.333	0.332	0	0	1	0	1	0	1	0	1
6	22	0.286	0.285	0	0	1	0	0	1	0	0	1
7	22	0.250	0.250	0	0	1	0	0	0	0	0	0

Table continues on the next page...

samples included in EMA calculation (N)	Number of cycles while lower frequency interrupt is masked	'a' value (decimal)	'a' value, adjusted by binary resolution	'a' value (binary)								
8	43	0.222	0.223	0	0	0	1	1	1	0	0	1
9	43	0.200	0.199	0	0	0	1	1	0	0	1	1
10	43	0.182	0.180	0	0	0	1	0	1	1	1	0
11	43	0.167	0.168	0	0	0	1	0	1	0	1	1
12	43	0.154	0.152	0	0	0	1	0	0	1	1	1
13	43	0.143	0.141	0	0	0	1	0	0	1	0	1
14	43	0.133	0.133	0	0	0	1	0	0	0	1	0
15	43	0.125	0.125	0	0	0	1	0	0	0	0	0
16	86	0.118	0.117	0	0	0	0	1	1	1	1	0
17	86	0.111	0.109	0	0	0	0	1	1	1	0	0
18	86	0.105	0.105	0	0	0	0	1	1	0	1	1
19	86	0.100	0.102	0	0	0	0	1	1	0	1	0
20	86	0.095	0.094	0	0	0	0	1	1	0	0	0
21	86	0.091	0.090	0	0	0	0	1	0	1	1	1
22	86	0.087	0.086	0	0	0	0	1	0	1	1	0
23	86	0.083	0.082	0	0	0	0	1	0	1	0	1
25	86	0.077	0.078	0	0	0	0	1	0	1	0	0
26	86	0.074	0.074	0	0	0	0	1	0	0	1	1
28	86	0.069	0.070	0	0	0	0	1	0	0	1	0
30	86	0.065	0.066	0	0	0	0	1	0	0	0	1
31	86	0.063	0.063	0	0	0	0	1	0	0	0	0
33	173	0.059	0.059	0	0	0	0	0	1	1	1	1
35	173	0.056	0.055	0	0	0	0	0	1	1	1	0
38	173	0.051	0.051	0	0	0	0	0	1	1	0	1
42	173	0.047	0.047	0	0	0	0	0	1	1	0	0
45	173	0.043	0.043	0	0	0	0	0	1	0	1	1
50	173	0.039	0.039	0	0	0	0	0	1	0	1	0
56	173	0.035	0.035	0	0	0	0	0	1	0	0	1
~64	173	0.031	0.031	0	0	0	0	0	1	0	0	0
~74	256	0.027	0.027	0	0	0	0	0	0	1	1	1
~86	256	0.023	0.023	0	0	0	0	0	0	1	1	0
~100	256	0.020	0.020	0	0	0	0	0	0	1	0	1
~128	256	0.016	0.016	0	0	0	0	0	0	1	0	0

Table continues on the next page...

samples included in EMA calculation (N)	Number of cycles while lower frequency interrupt is masked	'a' value (decimal)	'a' value, adjusted by binary resolution	'a' value (binary)								
~170	512	0.012	0.012	0	0	0	0	0	0	0	1	1
~260	512	0.008	0.008	0	0	0	0	0	0	0	1	0
~500	512	0.004	0.004	0	0	0	0	0	0	0	0	1
N/A		0.000	0.000	0	0	0	0	0	0	0	0	0

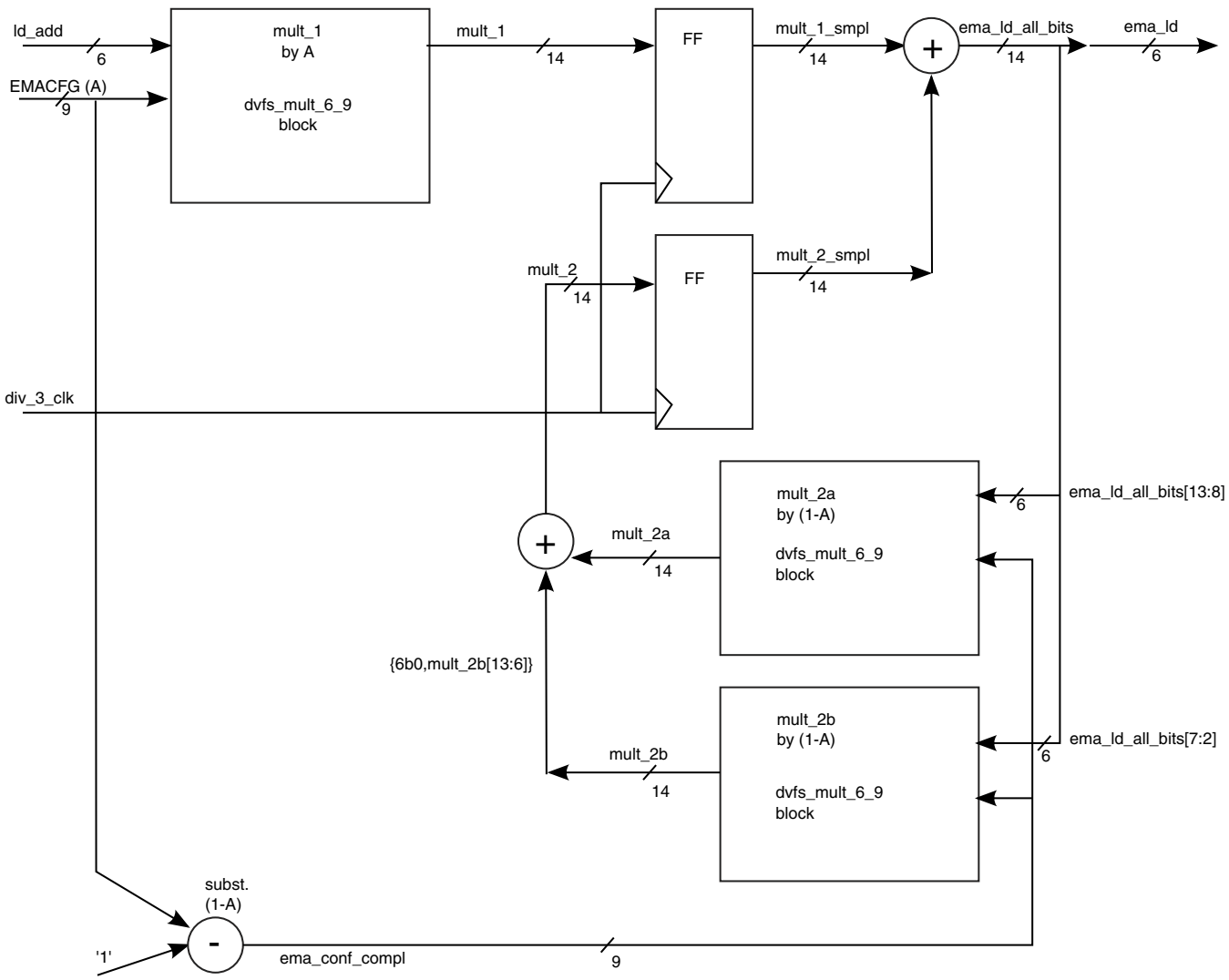


Figure 21-7. dvfs_ema_avg block diagram

The EMA block inputs are as follows:

- ld_add[5..0] - load level
- EMACFG[8..0] - 'a' parameter of EMA algorithm

- div_2_clk - fast clock don't see this in the diagram
- div_3_clk - slow clock

The EMA block outputs the following:

- ema_ld[5..0] - result of EMA algorithm (by div_3_clk)

The mult_a block provides multiplying between comp_load (6 bits) and EMACFG (9 bits). For the multiply operation, a faster clock (for internal sum) is required- this is provided by div_2_clk. Div_2_clk is faster than the div_3_clk by a factor of at least 16, which is enough for 6*9 or 9*6 operations. Only the highest 6 bits are taken from the result of mult_a.

The mult_b block operates in a similar manner.

The adder block's output is sampled with div_3_clk clock.

21.3.6 dvfs_thres_cmp Block

The dvfs_thres_cmp block compares the ARM Platform load value to programmable threshold levels. The comparators as shown in the drawing below are used:

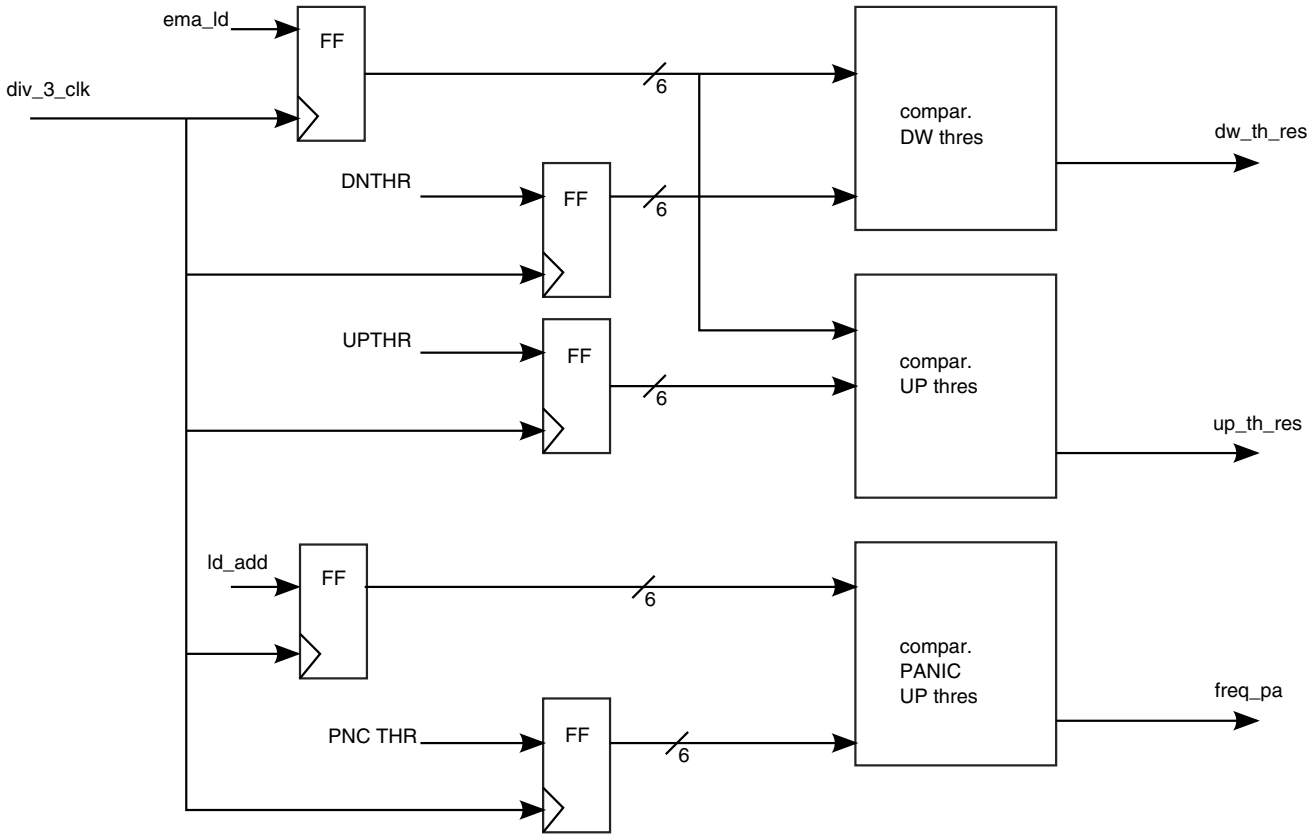


Figure 21-8. dvfs_tresh_cmp Diagram

The block is composed of three (3) comparators:

1. compar_dw_thres comparator, which compares between ema_ld signal (output of EMA block) and DNTHR signal (taken from config register, bits DNTHR).
2. compar_up_thres comparator, which compares between ema_ld signal (output of EMA block) and UPTHR signal (taken from config register, bits UPTHR).
3. compar_panic_up_thres comparator, which compares between ld_add signal (output of load_adder block) and PNCTHR signal (taken from config register, bits PNCTHR).

NOTE

The current implementation of this block enables step-by-step down frequency change. For more advanced option, the number of the DW threshold comparators should be increased (up to three for four-level DVFS).

21.3.7 dvfs_thresh_count Block

The purpose of the dvfs_thres_count block is to count consecutive threshold overcomes of dw_th_res and up_th_res (outputs of threshold_comp block). If any of the counters (see Figure 21-9 below) receives a null (zero) input synchronous with the clk3 signal, the counter is reset.

If the counter reaches a user defined value, its output is set to an active level. These counters are reset each time frequency scaling occurs or if the threshold overcomes are consecutive.

This block's output signals are saved in configuration register 0, bits [3,2].

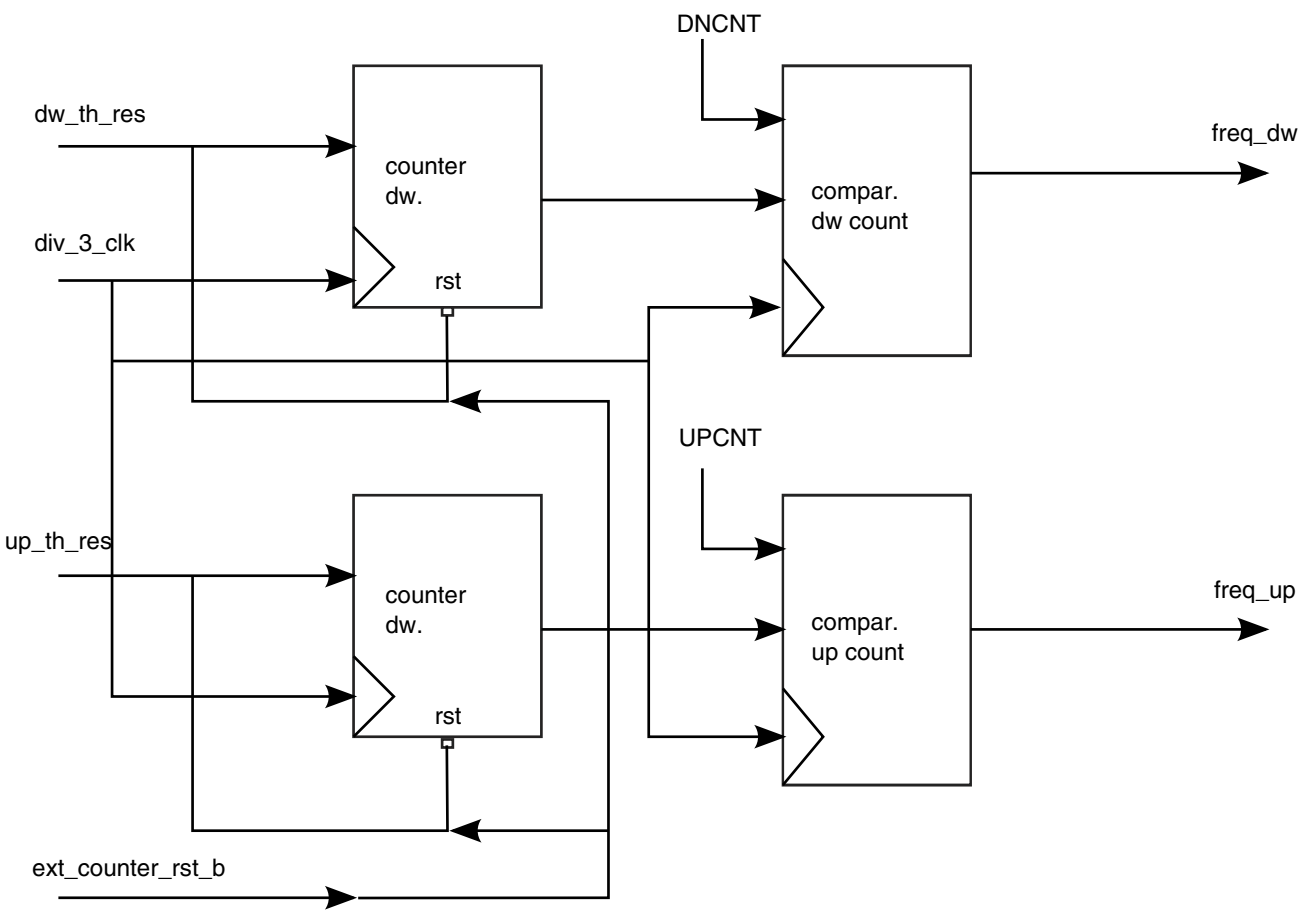


Figure 21-9. Thresh_counters block diagram

21.3.8 Load Tracking Buffer Register

The purpose of the load tracking buffer register is to save last 16 samples of the tracked load (before EMA operation). Hence, the 4 bits of ld_add signal (depending on the LTBRSH) are saved continuously, overwriting each time the latest sample. Each save is carried upon detecting an edge of the div_3_clk signal.

21.3.9 Frequency Pattern Generator

Frequency pattern generator is able to manage the frequency update requests periodically, additionally to main frequency update requests (if bit FPUE is set).

The periodic requests are created following the DVFSPT0, DVFSPT1, DVFSPT2 and DVFSPT3 register values, as described in Figure 21-10.

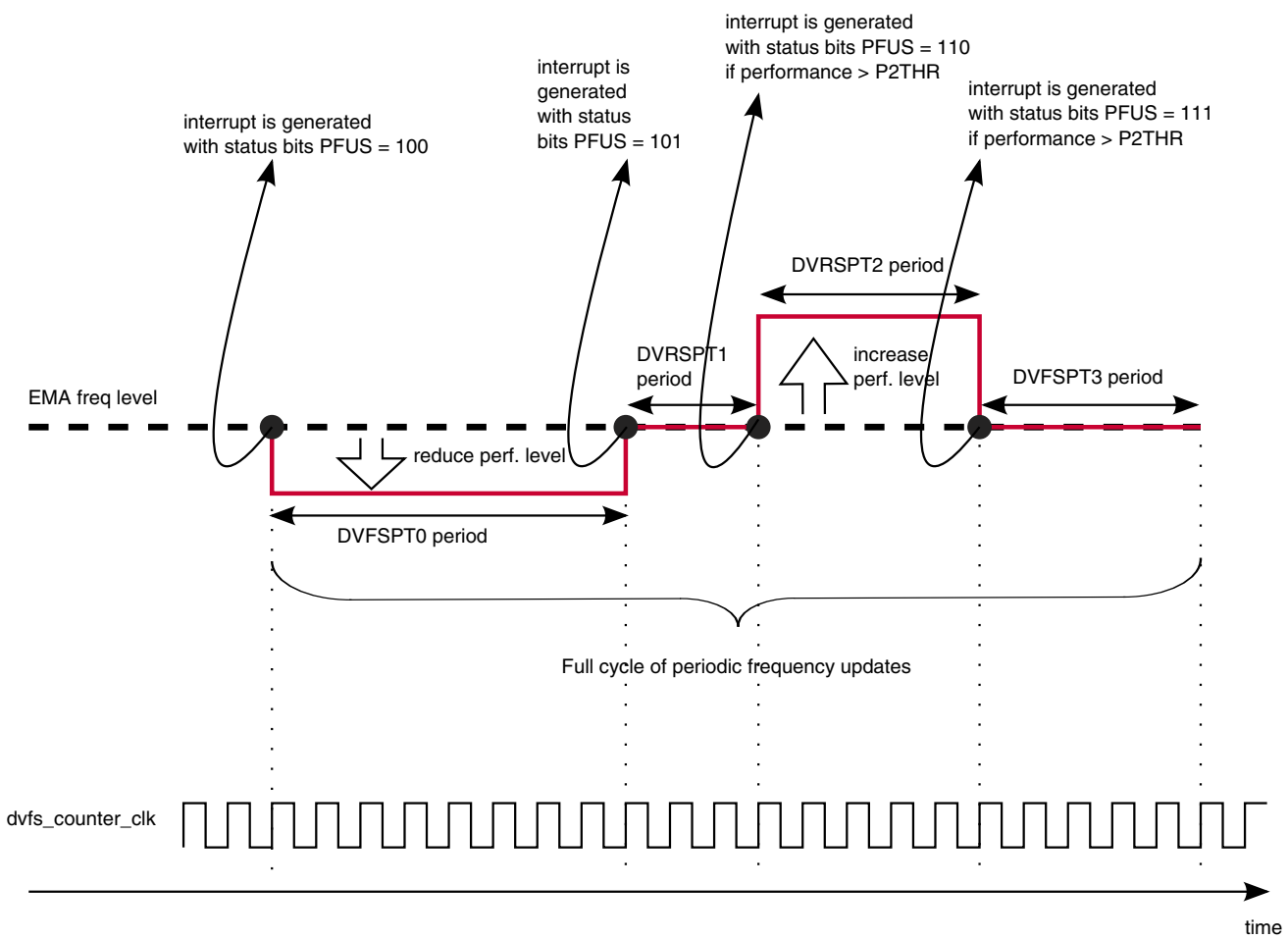


Figure 21-10. DVFS periodic frequency update requests

In case that one of the DVSPT0-DVFSPT3 period is set to "0", such frequency update will be skipped.

The periodic frequency update status is reflected in PFUS bits (reduce/increase performance request is an example - the actual steps taken upon period expiration are defined by s/w routine).

dvfs_counter_clk is ckih divided 64: 26MHz/64=0.40625MHz. The DVFSPT0, DVFSPT1, DVFSPT2 and DVFSPT3 counter are selected for 17 bits each to provide delay of 2^{18}-1=262143 counts, that are equal to 645ms. On the other hand, clk cycle of 0.40625MHz is ~2.46us, that is fast enough to provide high resolution for frequency management for tasks.

The DVFSPT2 and DVFSPT3 begin indication (interrupt) are conditional - only if the current performance is greater than P2THR bits at DVFSPT2 start, the interrupts will be created. Otherwise, the pattern delay will be counted, but without interrupt generation.

21.4 DVFS Output Event/interrupt Configuration

Event/Interrupt will be always high as long as LBFL is'1' and was not cleared by SW. Unless DVFEV (always event) is asserted. Then the event/interrupted will be toggled up and down every toggle of div_3_clk.

21.4.1 Interrupts

DVFS generates an interrupt that indicates that frequency and voltage update is needed. The user has to read the FSVAIM bits in order to know which change needs to be done.

21.5 Initialization Information

The user has to configure threshold values for load tracking and for counters and then enable the DVFS. When an interrupt is received, the user will change the frequency and the voltage in the interrupt handler.

NOTE

DVFS is a monitor that only provides an interrupt when counting exceeds the predefined value and doesn't actually send request to make a change a change of voltage and frequency. This can be done by the user in the interrupt routine or SDMA code by using GPC or CCM (re locking the PLL or changing the post dividers at the CCM).

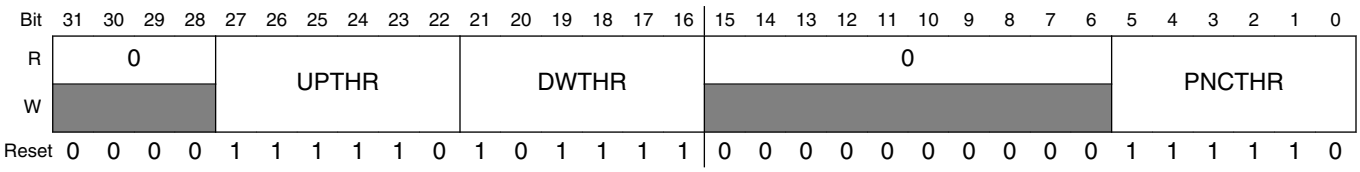
21.6 Programmable Registers

DVFSC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FD_C17F	DVFSC Thresholds (DVFSC_THRS)	32	R/W	0FAF_003Eh	21.6.1/ 1039
53FD_C183	DVFSC Counters thresholds (DVFSC_COUN)	32	R/W	0007_0020h	21.6.2/ 1039
53FD_C187	DVFSC general purpose bits weight (DVFSC_SIG1)	32	R/W	0000_0000h	21.6.3/ 1040
53FD_C18B	DVFSC general purpose bits weight (DVFSC_SIG0)	32	R/W	0000_0000h	21.6.4/ 1041
53FD_C18F	DVFSC general purpose bit 0 weight counter (DVFSC_GPC0)	32	R/W	0000_0000h	21.6.5/ 1042
53FD_C193	DVFSC general purpose bit 1 weight counter (DVFSC_GPC1)	32	R/W	0000_0000h	21.6.6/ 1043
53FD_C197	DVFSC general purpose bits enables (DVFSC_GPBT)	32	R/W	0000_0000h	21.6.7/ 1044
53FD_C19B	DVFSC EMAC settings (DVFSC_EMAC)	32	R/W	0000_0008h	21.6.8/ 1045
53FD_C19F	DVFSC Control (DVFSC_CNTR)	32	R/W	A900_0808h	21.6.9/ 1045
53FD_C1A3	DVFSC Load Tracking Register 0, portion 0 (DVFSC_LTR0_0)	32	R	0000_0000h	21.6.10/ 1048
53FD_C1A7	DVFSC Load Tracking Register 0, portion 1 (DVFSC_LTR0_1)	32	R	0000_0000h	21.6.11/ 1049
53FD_C1AB	DVFSC Load Tracking Register 1, portion 0 (DVFSC_LTR1_0)	32	R	0000_0000h	21.6.12/ 1050
53FD_C1AF	DVFS Load Tracking Register 3, portion 1 (DVFSC_LTR1_1)	32	R	0000_0000h	21.6.13/ 1051
53FD_C1B3	DVFSC pattern 0 length (DVFSC_PT0)	32	R/W	0000_0010h	21.6.14/ 1051
53FD_C1B7	DVFSC pattern 1 length (DVFSC_PT1)	32	R	0000_0010h	21.6.15/ 1052
53FD_C1BB	DVFSC pattern 2 length (DVFSC_PT2)	32	R/W	0000_0010h	21.6.16/ 1053
53FD_C1BF	DVFSC pattern 3 length (DVFSC_PT3)	32	R/W	0000_0010h	21.6.17/ 1053

21.6.1 DVFSC Thresholds (DVFSC_THRS)

Address: DVFSC_THRS is 53FD_C17Fh base + 0h offset = 53FD_C17Fh

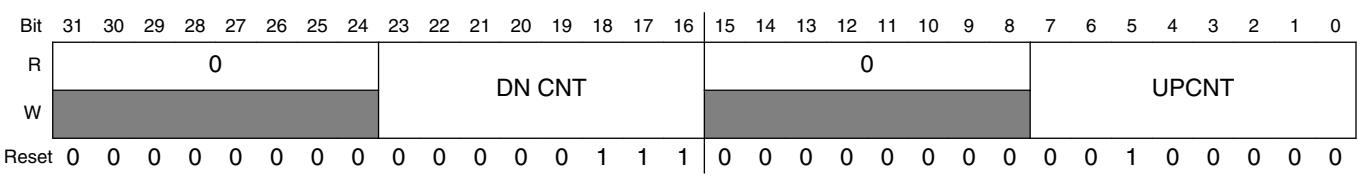


DVFSC_THRS field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value zero. Reserved
27–22 UPTHR	Upper threshold for load tracking
21–16 DWTHR	Down threshold for load tracking
15–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5–0 PNCTHR	Panic threshold for load tracking

21.6.2 DVFSC Counters thresholds (DVFSC_COUN)

Address: DVFSC_COUN is 53FD_C17Fh base + 4h offset = 53FD_C183h



DVFSC_COUN field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value zero. Reserved
23–16 DN CNT	Down counter threshold value
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–0 UPCNT	UP counter threshold value

21.6.3 DVFSC general purpose bits weight (DVFSC_SIG1)

Address: DVFSC_SIG1 is 53FD_C17Fh base + 8h offset = 53FD_C187h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WSW15			WSW14			WSW13		WSW12				WSW11			WSW10 [-13:2]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSW10[1:0]		WSW9			WSW8			WSW7			WSW6			-	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_SIG1 field descriptions

Field	Description
31–29 WSW15	General purpose load tracking signal weight dvfs_w_sig[15]
28–26 WSW14	General purpose load tracking signal weight dvfs_w_sig[14]
25–23 WSW13	General purpose load tracking signal weight dvfs_w_sig[13]
22–20 WSW12	General purpose load tracking signal weight dvfs_w_sig[12]
19–17 WSW11	General purpose load tracking signal weight dvfs_w_sig[11]
16–14 WSW10	General purpose load tracking signal weight dvfs_w_sig[10]
13–11 WSW9	General purpose load tracking signal weight dvfs_w_sig[9]
10–8 WSW8	General purpose load tracking signal weight dvfs_w_sig[8]
7–5 WSW7	General purpose load tracking signal weight dvfs_w_sig[7]
4–2 WSW6	General purpose load tracking signal weight dvfs_w_sig[6]
1–0 -	Reserved

21.6.4 DVFSC general purpose bits weight (DVFSC_SIG0)

Address: DVFSC_SIG0 is 53FD_C17Fh base + Ch offset = 53FD_C18Bh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSW5			WSW4			WSW3			WSW2			-				WSW1										WSW0						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

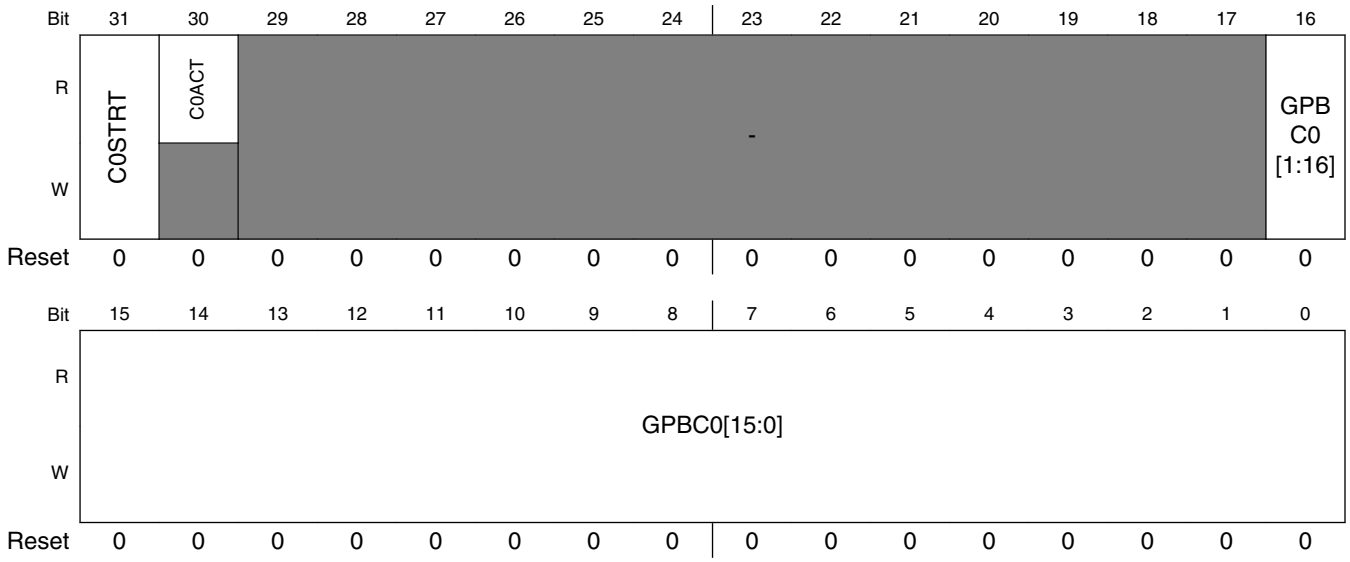
DVFSC_SIG0 field descriptions

Field	Description
31–29 WSW5	General purpose load tracking signal weight dvfs_w_sig[5]
28–26 WSW4	General purpose load tracking signal weight dvfs_w_sig[4]
25–23 WSW3	General purpose load tracking signal weight dvfs_w_sig[3]
22–20 WSW2	General purpose load tracking signal weight dvfs_w_sig[2]
19–12 -	Reserved
11–6 WSW1	General purpose load tracking signal weight dvfs_w_sig[1]. This value is relevant during GPC1 counting period or when GPB1 is set.
5–0 WSW0	General purpose load tracking signal weight dvfs_w_sig[0]. This value is relevant during GPC0 counting period or when GPB0 is set.

21.6.5 DVFSC general purpose bit 0 weight counter (DVFSC_GPC0)

DVFS general purpose bits weight counter.

Address: DVFSC_GPC0 is 53FD_C17Fh base + 10h offset = 53FD_C18Fh



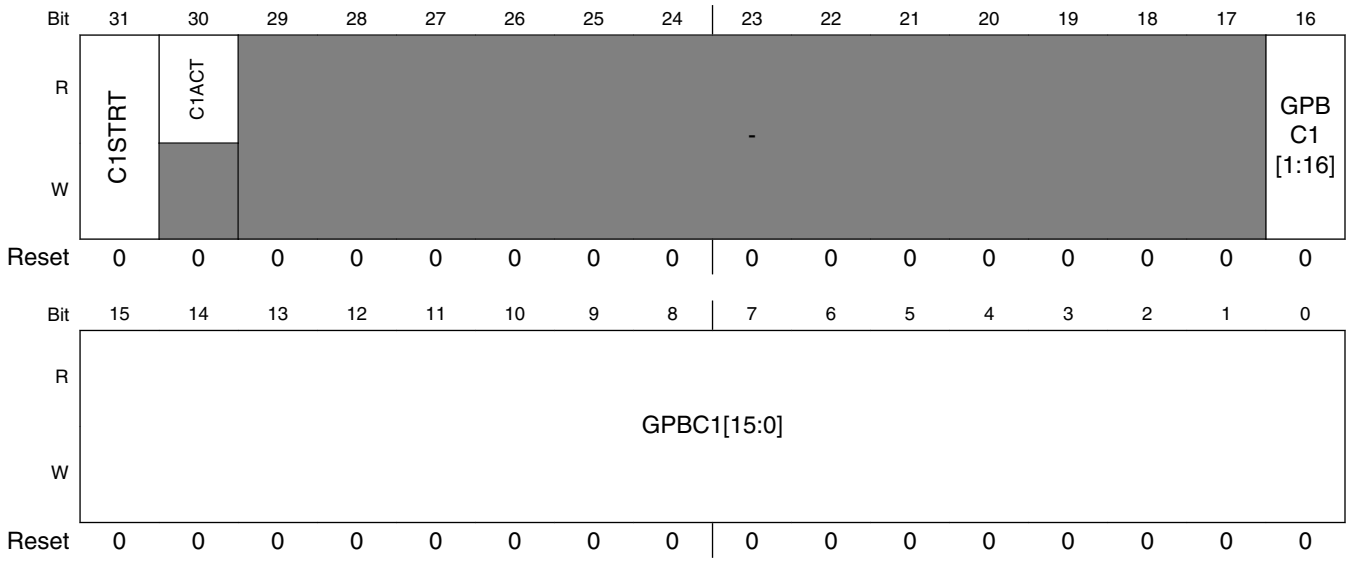
DVFSC_GPC0 field descriptions

Field	Description
31 C0STRT	C0STRT-Counter 0 start Setting of this bit will initialize down counting of the GPC0 value. Bit is self-cleared next cycle after setting. Any setting of this bit will re-start GPC0 counter to the GPC0 value. GPB0 bit disables (overrides) GPC0 counter-WSW0 weight is applicable continuously
30 C0ACT	C0ACT-Counter 0 active indicator 1 General Purpose bit0 counter didn't reach value of "0" - the WSW0 is provided to DVFS calculation 0 General Purpose bit0 counter reached value of "0" - the instead of WSW0, "0" (zero) is provided to DVFS calculation
29–17 -	Reserved
16–0 GPBC0	GPBC0 - General Purpose bits Counter 0 During period of this counter the GeP bit 0 will be set and WSW0 will be added to the calculations.

21.6.6 DVFSC general purpose bit 1 weight counter (DVFSC_GPC1)

DVFS general purpose bits weight counter1.

Address: DVFSC_GPC1 is 53FD_C17Fh base + 14h offset = 53FD_C193h



DVFSC_GPC1 field descriptions

Field	Description
31 C1STRT	C1STRT - Counter 1start Setting of this bit will initialize down counting of the GPC1 value. Bit is self-cleared next cycle after setting. Any setting of this bit will re-start GPC1 counter to the GPC1 value. GPB1 bit disables (overrides) GPC1 counter - WSW1 weight is applicable continuously
30 C1ACT	C1ACT - Counter 1 active indicator 1 General Purpose bit1 counter didn't reach value of "0" - the WSW1 is provided to DVFS calculation 0 General Purpose bit1 counter reached value of "0" - the instead of WSW1, "0" (zero) is provided to DVFS calculation
29–17 -	Reserved
16–0 GPBC1	GPBC1 - General Purpose bits Counter 1 During period of this counter the GeP bit 1 will be set and WSW1 will be added to the calculations.

21.6.7 DVFSC general purpose bits enables (DVFSC_GPBT)

Address: DVFSC_GPBT is 53FD_C17Fh base + 18h offset = 53FD_C197h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	reserved																GPB15	GPB14	GPB13	GPB12	GPB11	GPB10	GPB9	GPB8	GPB7	GPB6	GPB5	GPB4	GPB3	GPB2	GPB1	GPB0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_GPBT field descriptions

Field	Description
31–16 Reserved	This field is reserved. N/A
15 GPB15	General purpose bit 15. Its weight is set by WSW15 value.
14 GPB14	General purpose bit 14. Its weight is set by WSW14 value.
13 GPB13	General purpose bit 13. Its weight is set by WSW13 value.
12 GPB12	General purpose bit 12. Its weight is set by WSW12 value.
11 GPB11	General purpose bit 11. Its weight is set by WSW11 value.
10 GPB10	General purpose bit 10. Its weight is set by WSW10 value.
9 GPB9	General purpose bit 9. Its weight is set by WSW9 value.
8 GPB8	General purpose bit 8. Its weight is set by WSW8 value.
7 GPB7	General purpose bit 7. Its weight is set by WSW7 value.
6 GPB6	General purpose bit 6. Its weight is set by WSW6 value.
5 GPB5	General purpose bit 5. Its weight is set by WSW5 value.
4 GPB4	General purpose bit 4. Its weight is set by WSW4 value.
3 GPB3	General purpose bit 3. Its weight is set by WSW3 value.
2 GPB2	General purpose bit 2. Its weight is set by WSW2 value.
1 GPB1	General purpose bit 1. Its weight is set by WSW1 value. IF set (1), the GPBC1 operation is disregarded, WSW1 value is applied continuously.

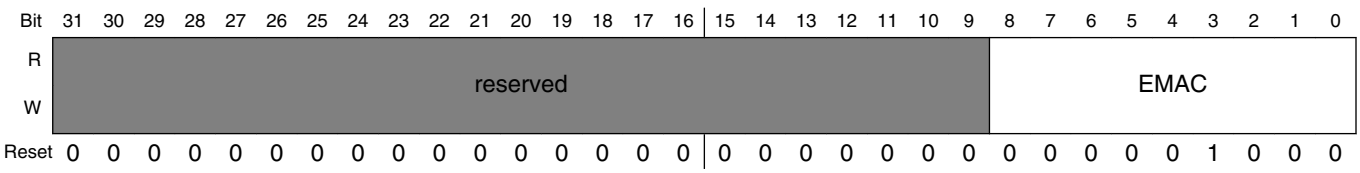
Table continues on the next page...

DVFSC_GPBT field descriptions (continued)

Field	Description
0 GPB0	General purpose bit 0. Its weight is set by WSW0 value. IF set (1), the GPBC0 operation is disregarded, WSW0 value is applied continuously.

21.6.8 DVFSC EMAC settings (DVFSC_EMAC)

Address: DVFSC_EMAC is 53FD_C17Fh base + 1Ch offset = 53FD_C19Bh



DVFSC_EMAC field descriptions

Field	Description
31–9 Reserved	This field is reserved. Reserved
8–0 EMAC	EMAC - EMA control value

21.6.9 DVFSC Control (DVFSC_CNTR)

Table 21-14. DIV3CK division

DIV3CK setting	dividing ratio	sum_3 passing bits	div_1_clk cumulative divider
00	1	4-0	1*512=512
001	4	6-2	4*512=2048
010	16	8-4	16*512=8192
011	64	10-6	64*512=32768
100	256	12-8	256*512=131072
101	1024	16-10	1024*512=524288

Table 21-15. Preliminary Divider definition

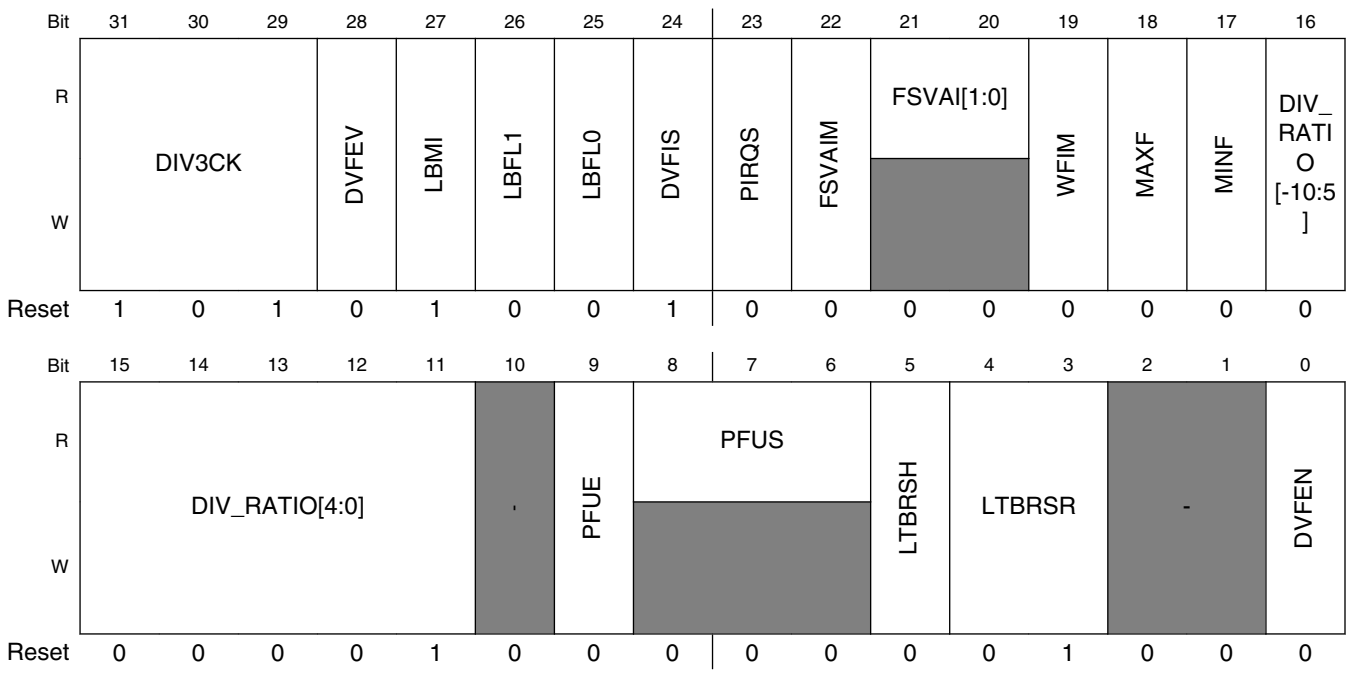
DIV_RATIO value	ARM clk division ratio
000000	1
000001	2

Table continues on the next page...

Table 21-15. Preliminary Divider definition (continued)

DIV_RATIO value	ARM clk division ratio
000010	3
...	...

Address: DVFSC_CNTR is 53FD_C17Fh base + 20h offset = 53FD_C19Fh



DVFSC_CNTR field descriptions

Field	Description
31–29 DIV3CK	DIV3CK - div_3_clk division ratio inside the DVFSC. According to the DVFSC Control (DVFSC_CNTR)
28 DVFEV	Always give a DVFS event. 0 Do not give an event always. 1 Always give event.
27 LBMI	Load buffer full mask interrupt. This bit masks the generation of this interrupt. Load buffer full interrupt is masked (LBFL0 and LBFL1 bits still will be updated, but interrupt won't be generated) 0 Load buffer full interrupt is enabled.
26 LBFL1	Load buffer 1 - full status bit. This bit indicates that log buffer registers are full. The bit is set to 1 automatically. An interrupt will be generated if LBMI bit is set to '0' Load buffer0 is full.Write '1' to clear. (write '0' leaves bit unchanged)

Table continues on the next page...

DVFSC_CNTR field descriptions (continued)

Field	Description
	0 Load buffer0 is not full.
25 LBFL0	Load buffer 0 - full status bit. This bit indicates that log buffer registers are full. The bit is set to 1 automatically. An interrupt will be generated if LBMI bit is set to "0" Load buffer1 is full. Write '1' to clear. (write '0' leaves bit unchanged) 0 Load buffer1 is not full.
24 DVFIS	DVFS Interrupt select. These bits define destination of DVFS interrupts. ARM platform interrupt will be generated for DVFS events. 0 SDMA interrupt will be generated for DVFS events.
23 PIRQS	PIRQS - Pattern IRQ Source * write '1' to clear. Writing '1' will clear interrupt if interrupt was from pattern 1 DVFS IRQ source was from pattern 0 DVFS IRQ source was not from pattern
22 FSVAIM	DVFS Frequency adjustment interrupt mask. This bit masks the DVFS frequency adjustment interrupt. FSVAI status bits will be still asserted in relevant cases. interrupt is masked. 0 Interrupt is enabled.
21–20 FSVAI[1:0]	FSVAI DVFS Frequency adjustment interrupt. These status bits indicate that the system frequency should be changed. 00 no interrupt 01 frequency should be increased. Low priority interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MAXF = 1 (highest frequency). 10 frequency should be decreased. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MINF= 1 (lowest frequency). 11 frequency should be increased immediately. High priority interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if MAXF = 1 (highest frequency).
19 WFIM	DVFS Wait for Interrupt mask bit 0 Wait for interrupt not masked 1 Wait for interrupt masked.
18 MAXF	Maximum frequency reached. Interrupt will not be created in maximum frequency reached and frequency increase required. 1 max frequency reached 0 max frequency not reached
17 MINF	Minimum frequency reached. Interrupt will not be created in minimum frequency reached and frequency decrease required. 1 min frequency reached 0 min frequency not reached

Table continues on the next page...

DVFSC_CNTR field descriptions (continued)

Field	Description
16–11 DIV_RATIO	DIV_RATIO - Divider value. Divider divides the input ARM clock, following the table DVFSC Control (DVFSC_CNTR)
10 -	Reserved
9 PFUE	PFUE - Period Frequency Update Enable 1 enabled 0 disabled
8–6 PFUS	PFUS - Periodic Frequency Update Status 000 no update 100 DVFSPT0 period, previous finished (can be performance level decrease) 101 DVFSPT1 period, previous finished (can be EMA-detected performance level) 110 DVFSPT2 period, previous finished (can be performance level increase) 111 DVFSPT3 period, previous finished (can be EMA-detected performance level)
5 LTBRSH	LTBRSH - Load Tracking Buffer Register Shift: 0 values of [5:2] of the selected input are saving in Load Tracking Buffer 1 values of [4:1] of the selected input are saving in Load Tracking Buffer
4–3 LTBRSR	LTBRSR - Load Tracking Buffer Register Source: 00 pre_ld_add 01 ld_add 10 after ema 11 reserved
2–1 -	Reserved
0 DVFEN	DVFEN DVFS enable. This bit enables the DVFSC. **NOTE:** Between disable and enable there has to be at least 3 cycles of div_3_clk. 1 DVFSC enabled. 0 DVFSC disabled.

21.6.10 DVFSC Load Tracking Register 0, portion 0 (DVFSC_LTR0_0)

Address: DVFSC_LTR0_0 is 53FD_C17Fh base + 24h offset = 53FD_C1A3h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS0_7				LTS0_6				LTS0_5				LTS0_4				LTS0_3				LTS0_2				LTS0_1				LTS0_0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_LTR0_0 field descriptions

Field	Description
31–28 LTS0_7	Load Tracking Sample 7
27–24 LTS0_6	Load Tracking Sample 6
23–20 LTS0_5	Load Tracking Sample 5
19–16 LTS0_4	Load Tracking Sample 4
15–12 LTS0_3	Load Tracking Sample 3
11–8 LTS0_2	Load Tracking Sample 2
7–4 LTS0_1	Load Tracking Sample 1
3–0 LTS0_0	Load Tracking Sample 0

21.6.11 DVFSC Load Tracking Register 0, portion 1 (DVFSC_LTR0_1)

Address: DVFSC_LTR0_1 is 53FD_C17Fh base + 28h offset = 53FD_C1A7h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS0_15				LTS0_14				LTS0_13				LTS0_12				LTS0_11				LTS0_10				LTS0_9				LTS0_8			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_LTR0_1 field descriptions

Field	Description
31–28 LTS0_15	Load Tracking Sample 15
27–24 LTS0_14	Load Tracking Sample 14
23–20 LTS0_13	Load Tracking Sample 13
19–16 LTS0_12	Load Tracking Sample 12
15–12 LTS0_11	Load Tracking Sample 11
11–8 LTS0_10	Load Tracking Sample 10

Table continues on the next page...

DVFSC_LTR0_1 field descriptions (continued)

Field	Description
7–4 LTS0_9	Load Tracking Sample 9
3–0 LTS0_8	Load Tracking Sample 8

21.6.12 DVFSC Load Tracking Register 1, portion 0 (DVFSC_LTR1_0)

Address: DVFSC_LTR1_0 is 53FD_C17Fh base + 2Ch offset = 53FD_C1ABh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS1_7				LTS1_6				LTS1_5				LTS1_4				LTS1_3				LTS1_2				LTS1_1				LTS1_0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DVFSC_LTR1_0 field descriptions

Field	Description
31–28 LTS1_7	Load Tracking Sample 7
27–24 LTS1_6	Load Tracking Sample 6
23–20 LTS1_5	Load Tracking Sample 5
19–16 LTS1_4	Load Tracking Sample 4
15–12 LTS1_3	Load Tracking Sample 3
11–8 LTS1_2	Load Tracking Sample 2
7–4 LTS1_1	Load Tracking Sample 1
3–0 LTS1_0	Load Tracking Sample 0

21.6.13 DVFS Load Tracking Register 3, portion 1 (DVFSC_LTR1_1)

Address: DVFSC_LTR1_1 is 53FD_C17Fh base + 30h offset = 53FD_C1AFh

Bit	31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
R	LTS1_15	LTS1_14	LTS1_13	LTS1_12	LTS1_11	LTS1_10	LTS1_9	LTS1_8
W								
Reset	0000	0000	0000	0000	0000	0000	0000	0000

DVFSC_LTR1_1 field descriptions

Field	Description
31–28 LTS1_15	Load Tracking Sample 15
27–24 LTS1_14	Load Tracking Sample 14
23–20 LTS1_13	Load Tracking Sample 13
19–16 LTS1_12	Load Tracking Sample 12
15–12 LTS1_11	Load Tracking Sample 11
11–8 LTS1_10	Load Tracking Sample 10
7–4 LTS1_9	Load Tracking Sample 9
3–0 LTS1_8	Load Tracking Sample 8

21.6.14 DVFSC pattern 0 length (DVFSC_PT0)

Address: DVFSC_PT0 is 53FD_C17Fh base + 34h offset = 53FD_C1B3h

Bit	31–24	23–18	17	16
R	-		PT0A	FPTN0 [1:16]
W				
Reset	00000000	000000	0	0

Bit	15–8	7–0
R	FPTN0[15:0]	
W		
Reset	00000000	00010000

DVFSC_PT0 field descriptions

Field	Description
31–18 -	Reserved
17 PT0A	PT0A - Pattern 0 currently active (read-only) 1 active 0 non-active
16–0 FPTN0	FPTN0 - Frequency pattern 0 counter During period of this counter the frequency will be reduced from the EMA-detected level.

21.6.15 DVFSC pattern 1 length (DVFSC_PT1)

Address: DVFSC_PT1 is 53FD_C17Fh base + 38h offset = 53FD_C1B7h

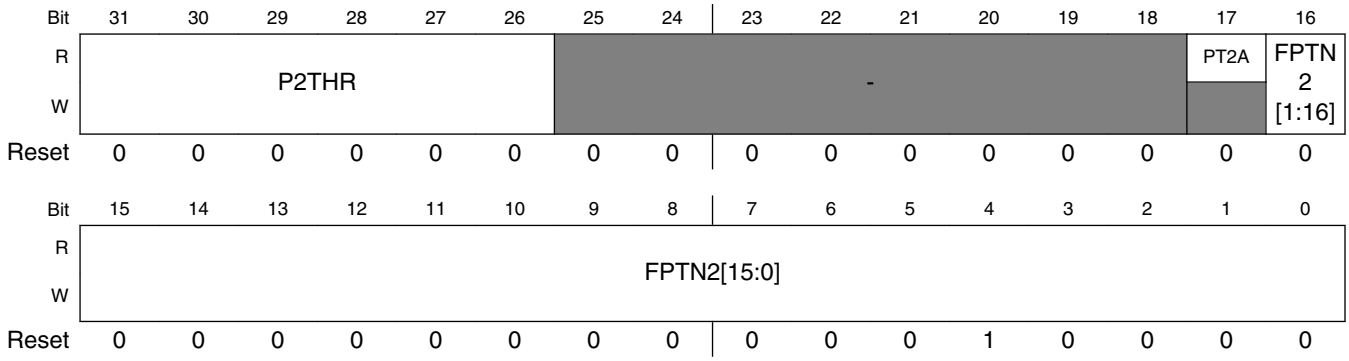
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-														PT1A		FPTN1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

DVFSC_PT1 field descriptions

Field	Description
31–18 -	Reserved
17 PT1A	PT1A - Pattern 1 currently active (read-only) 1 active 0 non-active
16–0 FPTN1	FPTN1 - Frequency pattern 1 counter During period of this counter the frequency will be set to the EMA-detected level.

21.6.16 DVFSC pattern 2 length (DVFSC_PT2)

Address: DVFSC_PT2 is 53FD_C17Fh base + 3Ch offset = 53FD_C1BBh

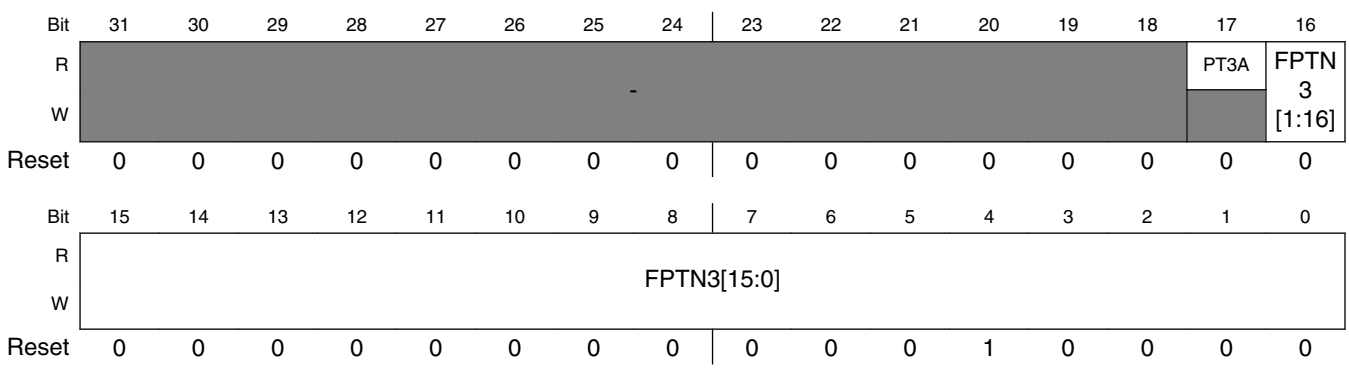


DVFSC_PT2 field descriptions

Field	Description
31–26 P2THR	P2THR - Pattern 2 Threshold Threshold of current DVFS load (after EMA), for generating interrupts with PFUS indicators 110, 111. If the current performance is greater than the P2THR value, the interrupts will be generated. Otherwise, pattern delay will be counted, but without interrupt generation.
25–18 -	Reserved
17 PT2A	PT2A - Pattern 2 currently active (read-only) 1 active 0 non-active
16–0 FPTN2	FPTN2 - Frequency pattern 2 counter During period of this counter the frequency will be increased to higher, than detected by the EMA-detected level.

21.6.17 DVFSC pattern 3 length (DVFSC_PT3)

Address: DVFSC_PT3 is 53FD_C17Fh base + 40h offset = 53FD_C1BFh



DVFSC_PT3 field descriptions

Field	Description
32–18 -	Reserved
17 PT3A	PT3A - Pattern 3 currently active (read-only) 1 active 0 non-active
16–0 FPTN3	FPTN3 - Frequency pattern 3 counter During period of this counter the frequency will be set to the EMA-detected level.

Chapter 22
Enhanced Configurable SPI (ECSPI)

22.1 Overview

This chapter describes a block integrated into an SoC. The chapter is intended for a block driver software developer. It describes block-level operation and programming. To understand how the block is integrated at the SoC level, a system software developer should see discussions of the block in the appropriate SoC-level chapter(s).

The Enhanced Configurable Serial Peripheral Interface (ECSPI) is a full-duplex, synchronous, four-wire serial communication block. The ECSPI contains a 64 x 32 receive buffer (RXFIFO) and a 64 x 32 transmit buffer (TXFIFO). With data FIFOs, the ECSPI allows rapid data communication with fewer software interrupts. The following figure shows a block diagram of the ECSPI.

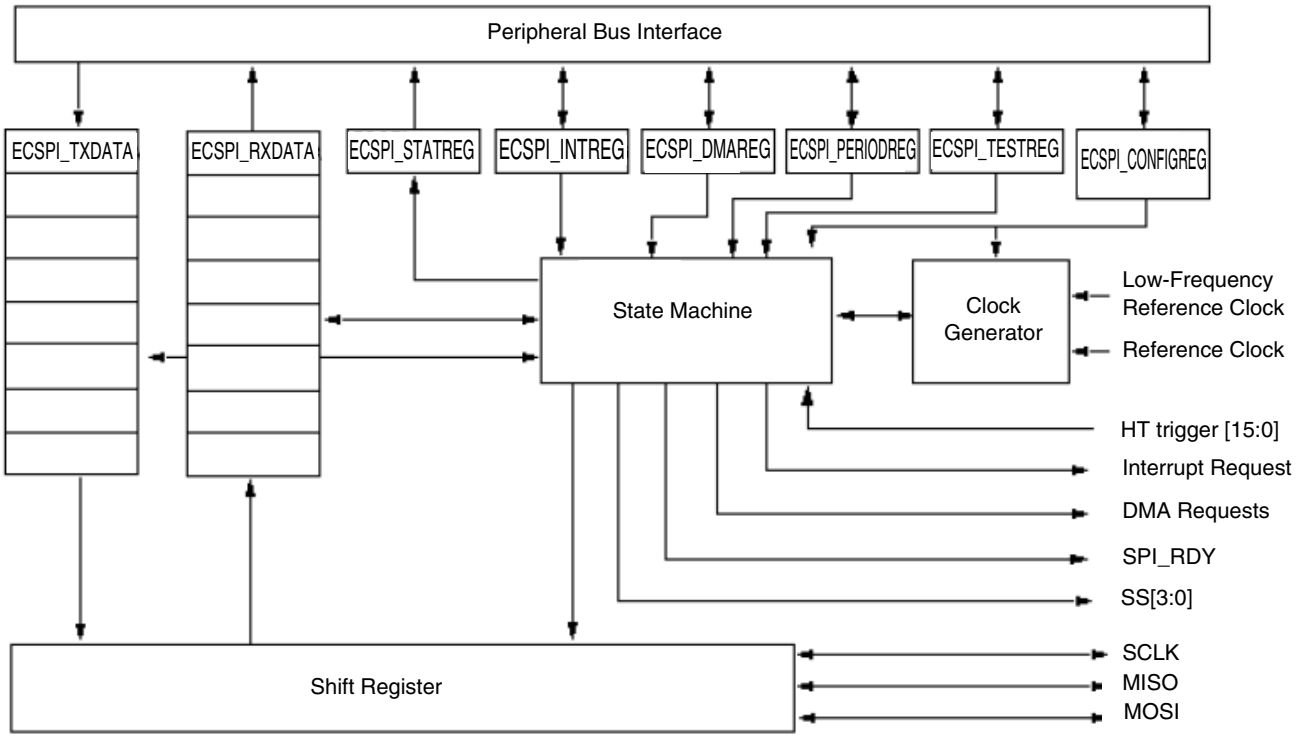


Figure 22-1. ECSPI Block Diagram

22.1.1 Features

Key features of the ECSPI include:

- Full-duplex synchronous serial interface
- Master/Slave configurable
- Four Chip Select (SS) signals to support multiple peripherals
- Transfer continuation function allows unlimited length data transfers
- 32-bit wide by 64-entry FIFO for both transmit and receive data
- 32-bit wide by 16-entry FIFO for HT message data
- Polarity and phase of the Chip Select (SS) and SPI Clock (SCLK) are configurable
- Direct Memory Access (DMA) support
- Max operation frequency up to the reference clock frequency.

22.1.2 Modes and Operations

The ECSPI supports the modes described in the indicated sections:

- Master Mode

- Slave Mode
- Low Power Modes

As described in Operations, the ECSPI supports the operations described in the indicated sections:

- Typical Master Mode
  - Master Mode with SPI_RDY
  - Master Mode with Wait States
  - Master Mode with SS_CTL[3:0] Control
  - Master Mode with Phase Control
- Typical Slave Mode

22.2 External Signals

The table below describes all ECSPI Block I/Os.

Table 22-1. Block I/O Signals

Signal	I/O	Description	Reset State[1]	Pull-Up/ Down[1]
$\overline{SS}$[3:0]	I/O	Chip selects	1	-
SCLK	I/O	SPI clock	0	Active
MISO	I/O	Master data in; slave data out	0	Passive
MOSI	I/O	Master data out; slave data in	0	-
SPI_RDY	I	Master data out; slave data in	0	Active

1. The reset state values and pull-up/down requirements provided in this table are from the block-level perspective. To understand how the block is integrated at the SoC level, the system software developer must see discussions of the block in the appropriate SoC-level chapter(s). For example, a block signal that requires a pull-up could be integrated with a pull-up option built into the SoC. In this case, the system software developer must ensure the proper programming at the SoC level.

The figure below shows the ECSPI in master mode connected to four external devices in a one-way communication link.

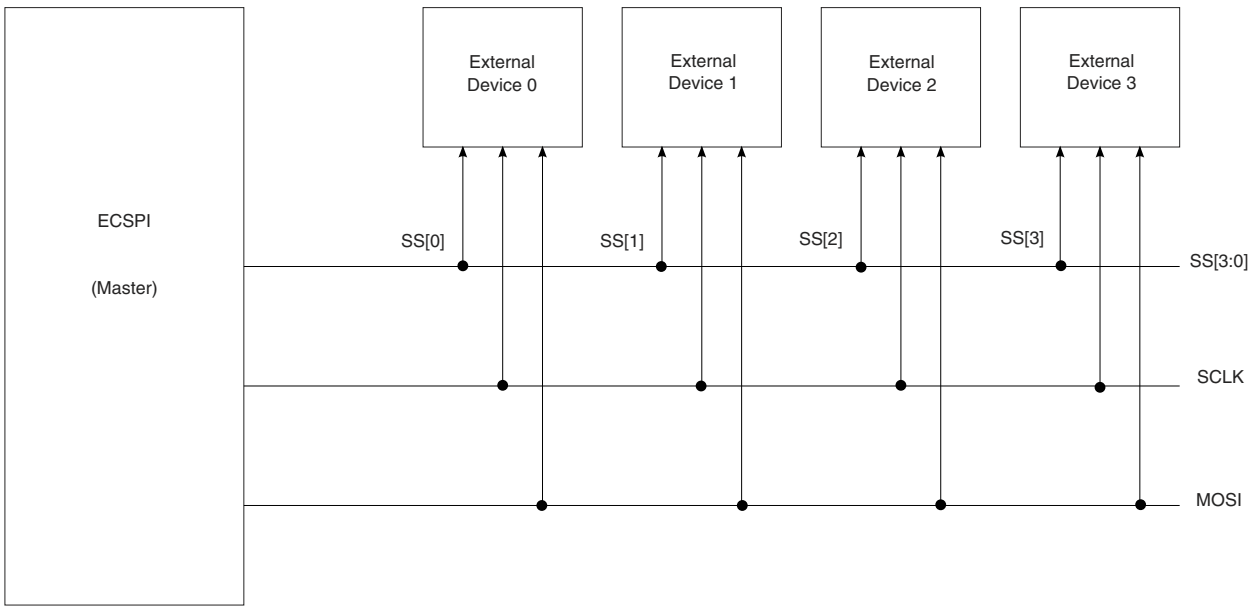


Figure 22-2. Example Connection Diagram

22.3 Functional Description

This section provides a complete functional description of the ECSPI. The figure below shows the relationship of SCLK and data lines while ECSPI has been configured with different POL and PHA settings.

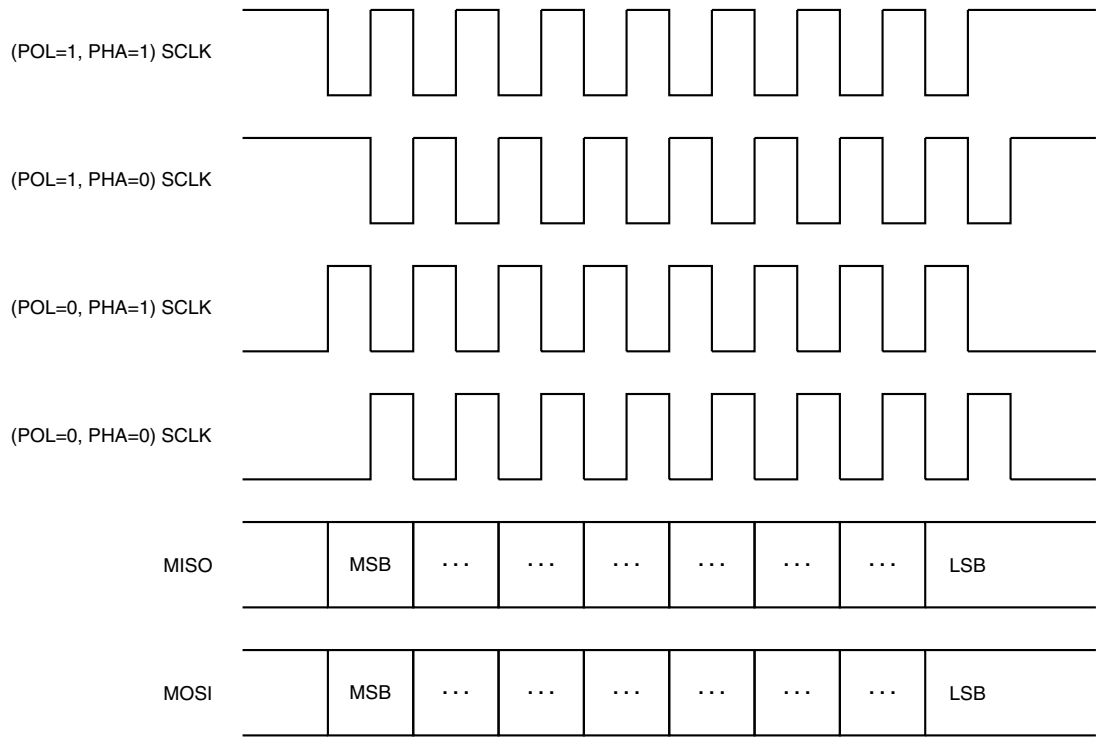


Figure 22-3. ECSPI SCLK, MISO, and MOSI Relationship

22.3.1 Master Mode

When the ECSPI is configured as a master, it uses a serial link to transfer data between the ECSPI and an external slave device. One of the Chip Select (SS) signals and the clock signal (SCLK) are used to transfer data between two devices.

If the external device is a transmit-only device, the ECSPI master's output port can be ignored and used for other purposes. In order to use the internal TXFIFO and RXFIFO, two auxiliary output signals, Chip Select (SS) and SPI_RDY, are used for data transfer rate control. Software can also configure the sample period control register to a fixed data transfer rate.

22.3.2 Slave Mode

When the ECSPI is configured as a slave, software can configure the ECSPI Control register to match the external SPI master's timing. In this configuration, Chip Select ($\overline{SS}$) becomes an input signal, and is used to control data transfers through the Shift register, as well as to load/store the data FIFO.

22.3.3 Hardware Trigger (HT) Mode

In hardware trigger (HT) mode, the behavior of the ECSPI is similar to master mode. The differences are in the trigger mechanism and the burst length.

When the block is in HT mode, the SPI burst is no longer triggered by software, setting the XCH bit or writing to the TXFIFO. Instead, there are 16 input signals HT trigger[15:0] associated with each slot of the 16-entry ECSPI_MSGDATA FIFO, and a positive pulse will trigger the ECSPI to transmit a SPI burst with data in ECSPI_MSGDATA FIFO. And in this case, the burst length is limited in 32-bit, but also configurable. Only SPI channel 0 can be used in HT mode.

22.3.4 Low Power Modes

The ECSPI does not operate under low power mode. It holds its operation when its clock is gated off in master mode. In slave mode, the ECSPI does not respond when its clock is gated off.

22.3.5 Operations

This section describes the ECSPI's operations.

22.3.5.1 Typical Master Mode

The ECSPI master uses the Chip Select (SS) signal to enable an external SPI device, and uses the SCLK signal to transfer data in and out of the Shift register. The SPI_RDY enables fast data communication with fewer software interrupts. By programming the ECSPI_PERIODREG register accordingly, the ECSPI can be used for a fixed data transfer rate.

When the ECSPI is in Master mode the SS, SCLK, and MOSI are output signals, and the MISO signal is an input.

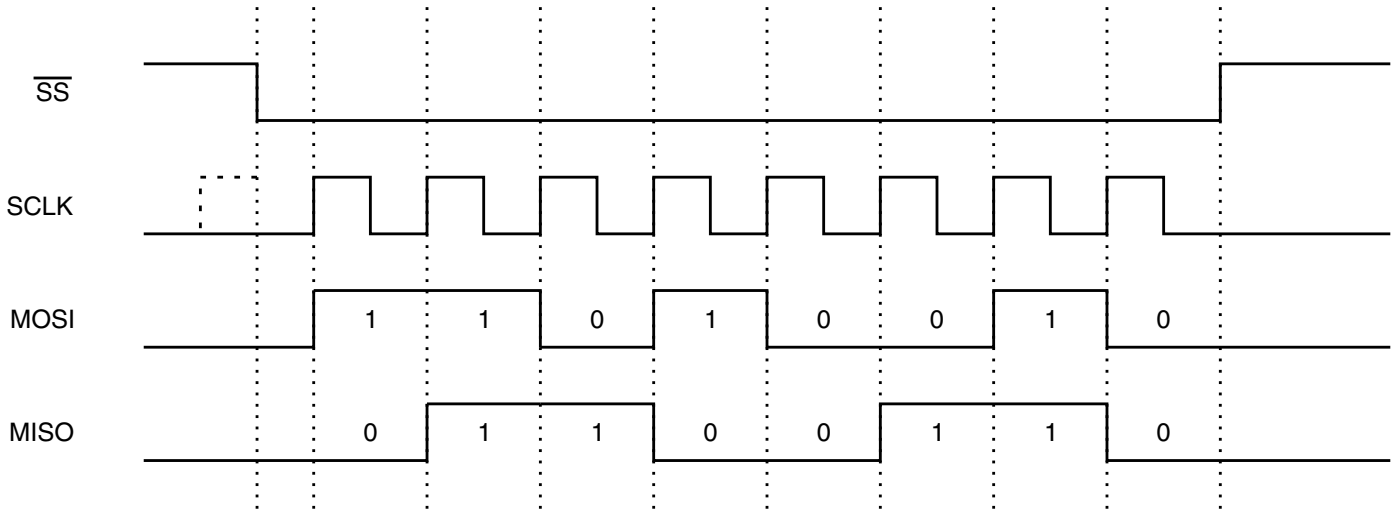


Figure 22-4. Typical SPI Burst (8-bit Transfer)

In the above figure, the Chip Select (SS) signal enables the selected external SPI device, and the SCLK synchronizes the data transfer. The MOSI and MISO signals change on rising edge of SCLK and the MISO signal is latched on the falling edge of the SCLK. The figure above shows a data of 0xD2 is shifted out, and a data of 0x66 is shifted in.

22.3.5.1.1 Master Mode with SPI_RDY

By default, the ECSPI does not use the SPI_RDY signal in master mode (MODE =1).

A SPI burst begins when the following events happen:

- The ECSPI is enabled, TXFIFO has data in it, and ECSPI_CONREG[XCH] bit or the ECSPI_CONREG[SMC] bit is set.
- When the SPI Data Ready Control (ECSPI_CONREG[DRCTL]) bits contains either 01 or 10, the SPI_RDY signal controls when a SPI burst starts.

A SPI burst is defined as a bus transaction that starts when the slave select is asserted and ends when the slave select is negated. The Chip Select (SS) signal will remain asserted until all the bits in a SPI burst are shifted out.

If ECSPI_CONREG[DRCTL] is set to 01, the SPI burst can be triggered only if a falling edge of the SPI_RDY signal has been detected.

The following figure shows the relationship between a SPI burst and the falling edge of SPI_RDY signal.

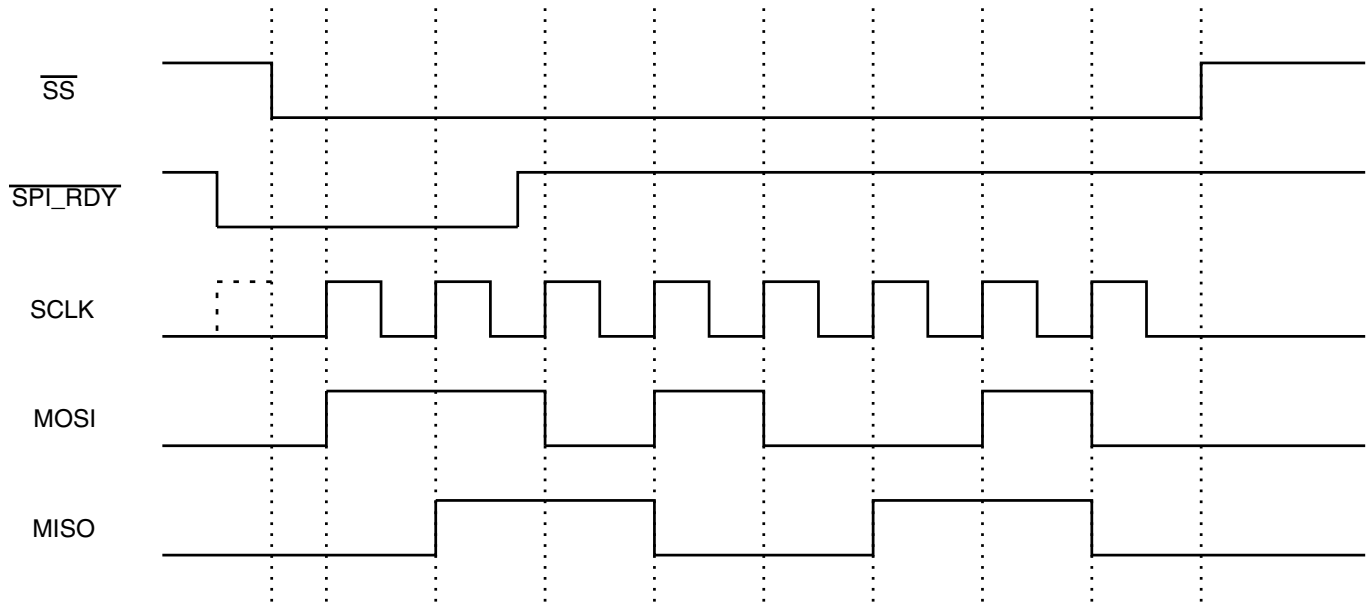


Figure 22-5. Relationship Between a SPI Burst and SPI_RDY: Falling-Edge Triggered

A SPI burst does not start until the falling edge of the SPI_RDY signal is detected. The next SPI burst starts when the next SPI_RDY falling edge is detected, after the last burst has finished.

If SPI Data Ready Control (ECSPI_CONREG[DRCTL]) is set to 10, the SPI burst can be triggered only if the SPI_RDY signal is low.

The following figure shows the relationship between a SPI burst and the SPI_RDY signal. The SPI burst does not begin until the SPI_RDY signal goes low. The ECSPI will keep transmitting SPI burst if the SPI_RDY signal remains low.

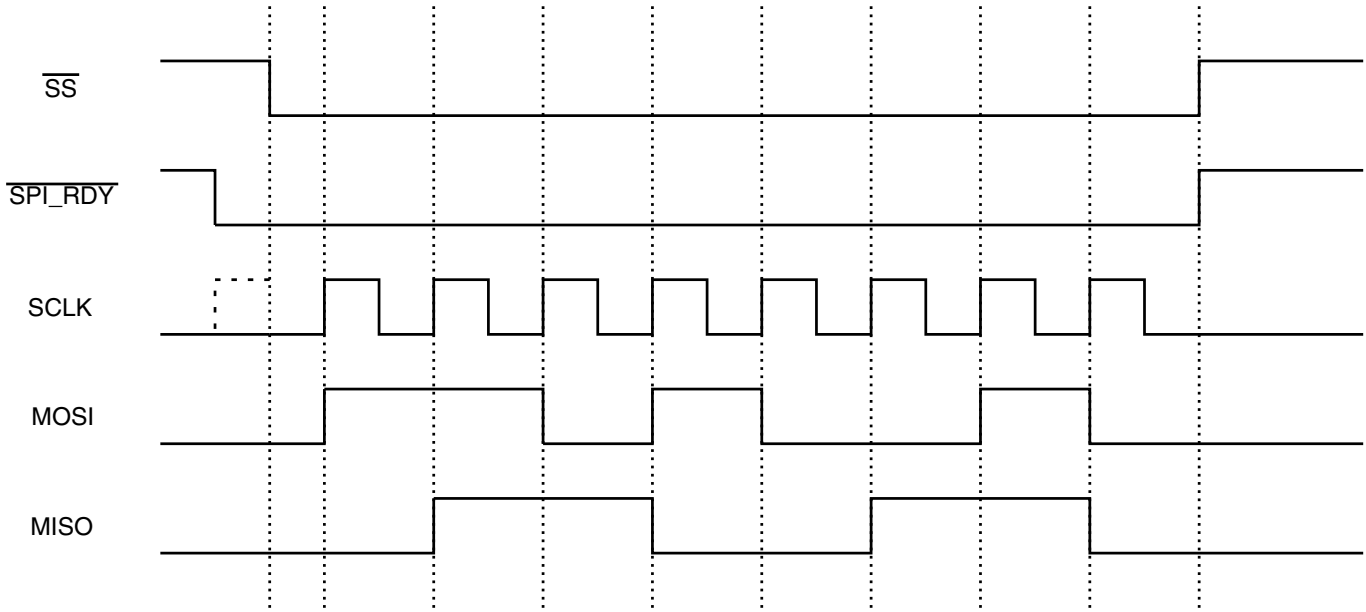


Figure 22-6. Relationship Between a SPI Burst and SPI_RDY: Low-Level Triggered

22.3.5.1.2 Master Mode with Wait States

Wait states can be inserted between SPI bursts. This provides a way for software to slow down the SPI burst to meet the timing requirements of a slower SPI device.

The following figure shows wait states inserted between SPI bursts.

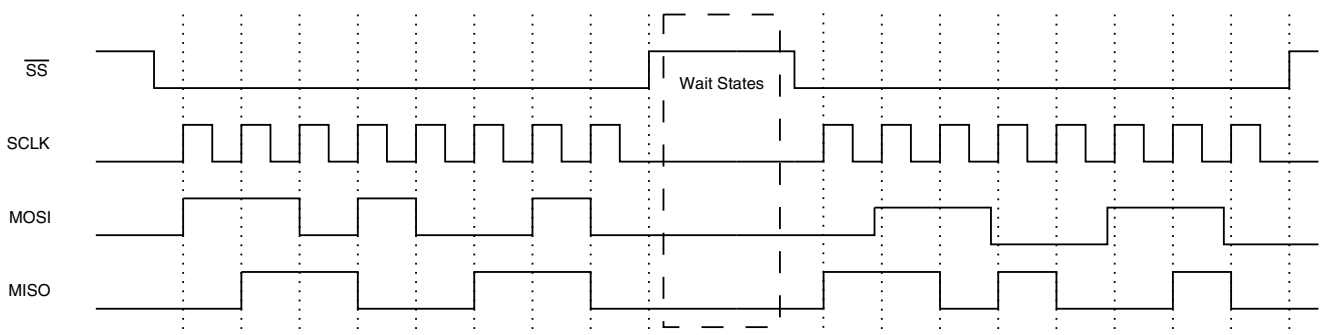


Figure 22-7. SPI Bursts with Wait States

In this case, the number of wait states is controlled by ECSPI_PERIODREG[SAMPLE PERIOD] and the wait states' clock source is selected by ECSPI_PERIODREG[CSRC].

22.3.5.1.3 Master Mode with SS_CTL[3:0] Control

The SPI SS Control (SS_CTL[3:0]) controls whether the current operation is single burst or multiple bursts. When the SPI SS Wave Form Select (SS_CTL[3:0]) is set, the current operation is multiple bursts transfer. When the SPI SS Wave Form Select (SS_CTL[3:0]) bit is cleared, the current operation is single burst transfer. A SPI burst can contains multiple words as defined in the BURST LENGTH field of the ECSPI_CONREG register.

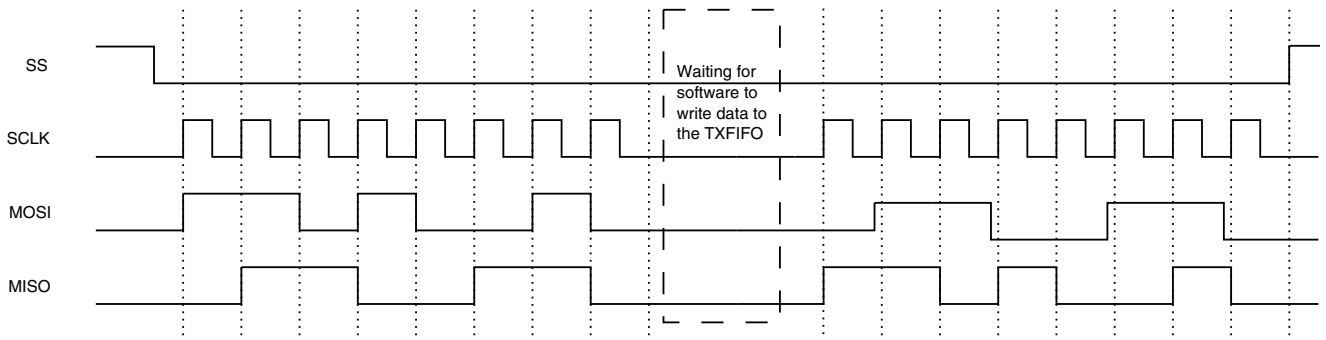


Figure 22-8. SPI Burst While SS_CTL[3:0] is Clear

In Figure 22-8, two 8-bit bursts in the TXFIFO have been combined and transmitted in one SPI burst. The maximum length of a single SPI burst is defined in the BURST LENGTH field of the ECSPI_CONREG control register. (Figure 22-8 corresponds to a BURST LENGTH of 8.) This provides a way for transferring a longer SPI burst by writing data into TXFIFO while the ECSPI is transmitting.

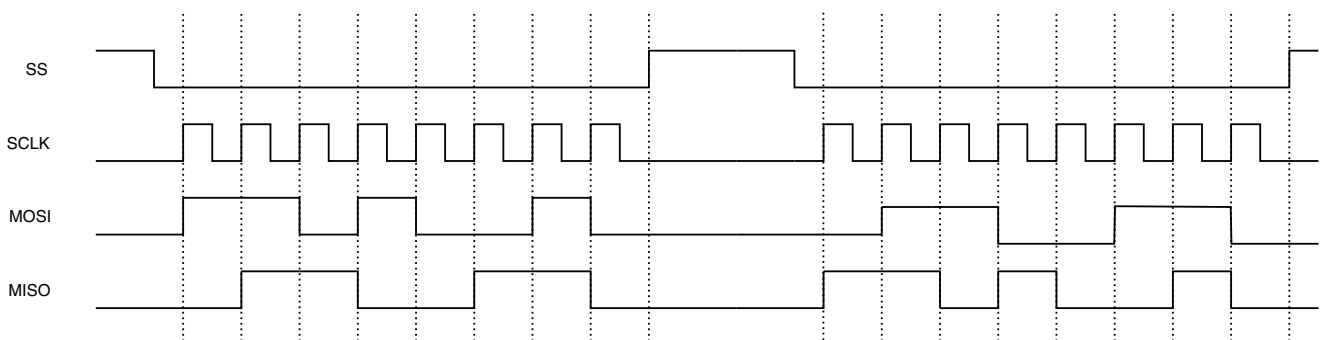


Figure 22-9. SPI Bursts While SS_CTL[3:0] is Set

In Figure 22-9, two FIFO entries are transmitted, one entry with each SPI burst. The ECSPI will continue to transmit SPI bursts until the TXFIFO is empty. When wait states can be inserted between SPI bursts, the SS will negate between SPI bursts until the wait states finish.

22.3.5.1.4 Master Mode with Phase Control

The Phase Control (ECSPI_CONREG[PHA]) bit controls how the transmit data shifts out and the receive data shifts in.

When the Phase control (ECSPI_CONREG[PHA]) bit is set, the transmit data will shift out on the rising edge of SCLK, and the receive data is latched on the falling edge of SCLK. The most-significant bit is output on the first rising SCLK edge.

When ECSPI_CONREG[PHA] is cleared, the transmit data is shifted out on the falling edge of SCLK and the receive data is latched on the rising edge of SCLK. The MSB is output when the host processor loads the transmitted data.

Inverting the SCLK polarity does not impact the edge-triggered operations because they are internal to the serial peripheral interface master.

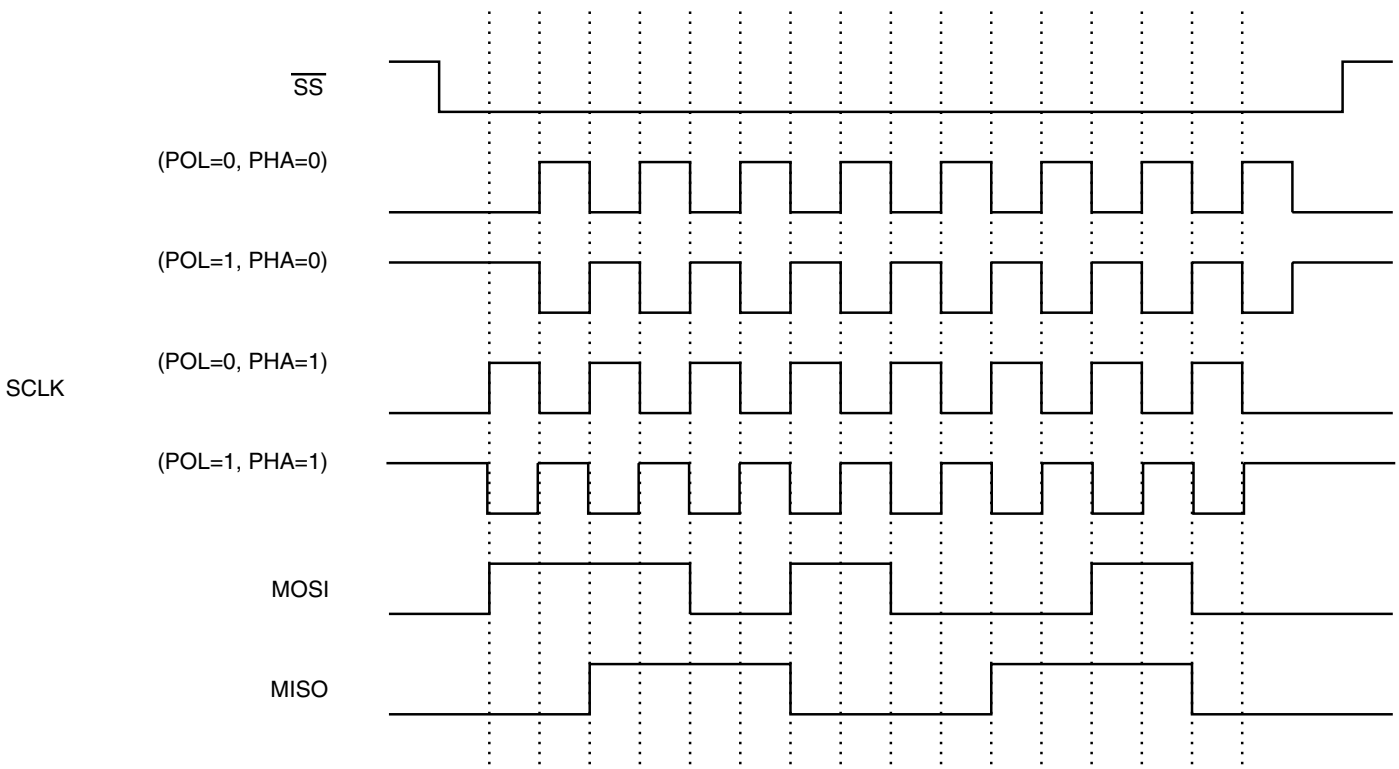


Figure 22-10. SPI Burst with Different POL and PHA Configurations

22.3.5.2 Typical Slave Mode

When the ECSPI is configured as a slave (Mode = 0), software can configure the ECSPI Control register to match the external SPI master's timing. In this configuration, SS becomes an input signal, and is used to latch data in and out of the internal data Shift registers, as well as to advance the data FIFO.

The SS, SCLK, and MOSI are inputs and MISO is output. Most of the timing diagrams are similar to the diagrams shown previously for the SPI in Master mode (Mode = 1), because the inputs come from a SPI master device.

However, the timing is different when SS is used to advance the data FIFO. When the SS_CTL[3:0] is set while the ECSPI is configured in Slave mode, the data FIFO will advance on the rising edge of the SS signal. When the polarity is reversed (SSPOL = 1), the data FIFO will advance on the falling edge of the SS signal.

The figure below shows a SPI burst in which the data FIFO is advanced by the rising edge of the SS signal.

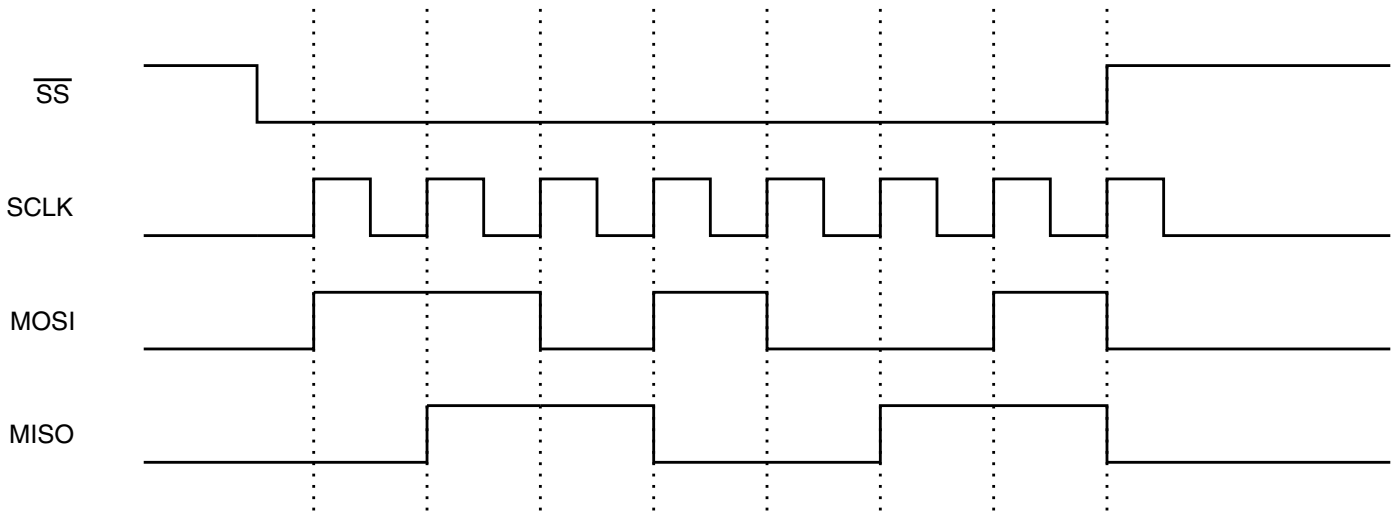


Figure 22-11. Advancing the Data FIFO on the Rising Edge of $\overline{SS}$

In the above case, only the most significant 7 bits are loaded to the RXFIFO.

22.3.6 Clocks

This section describes clocks and special clocking requirements of the block.

ECSPI has the following clock inputs:

- Reference Clock is the reference clock.
- Low-Frequency Reference Clock is a 32KHz input clock optionally used for counting wait states.

22.3.7 Reset

Whenever a device reset occurs, a reset is performed on the ECSPI, resetting all registers to their default values.

Software can reset the block using the CONREG[EN] bit; see Control Register (ECSPI_CONREG).

22.3.8 Interrupts

Interrupt control provides a way to manage the ECSPI FIFOs:

- For transmitting data, software can enable the TXFIFO empty, TXFIFO data request, and TXFIFO full interrupts to maintain the TXFIFO using an interrupt service routine.
- For receiving data, software can enable the RXFIFO ready, RXFIFO data request, and RXFIFO full interrupts to retrieve data from the RXFIFO using an interrupt service routine.

Other interrupt sources can be used to control or debug the SPI bursts:

- The transfer-completed interrupt means that there is no data left in the TXFIFO and that the data in the Shift register has been shifted out.
- The RXFIFO overflow interrupt means that the RXFIFO received more than 64 words and will not accept any other words.

.

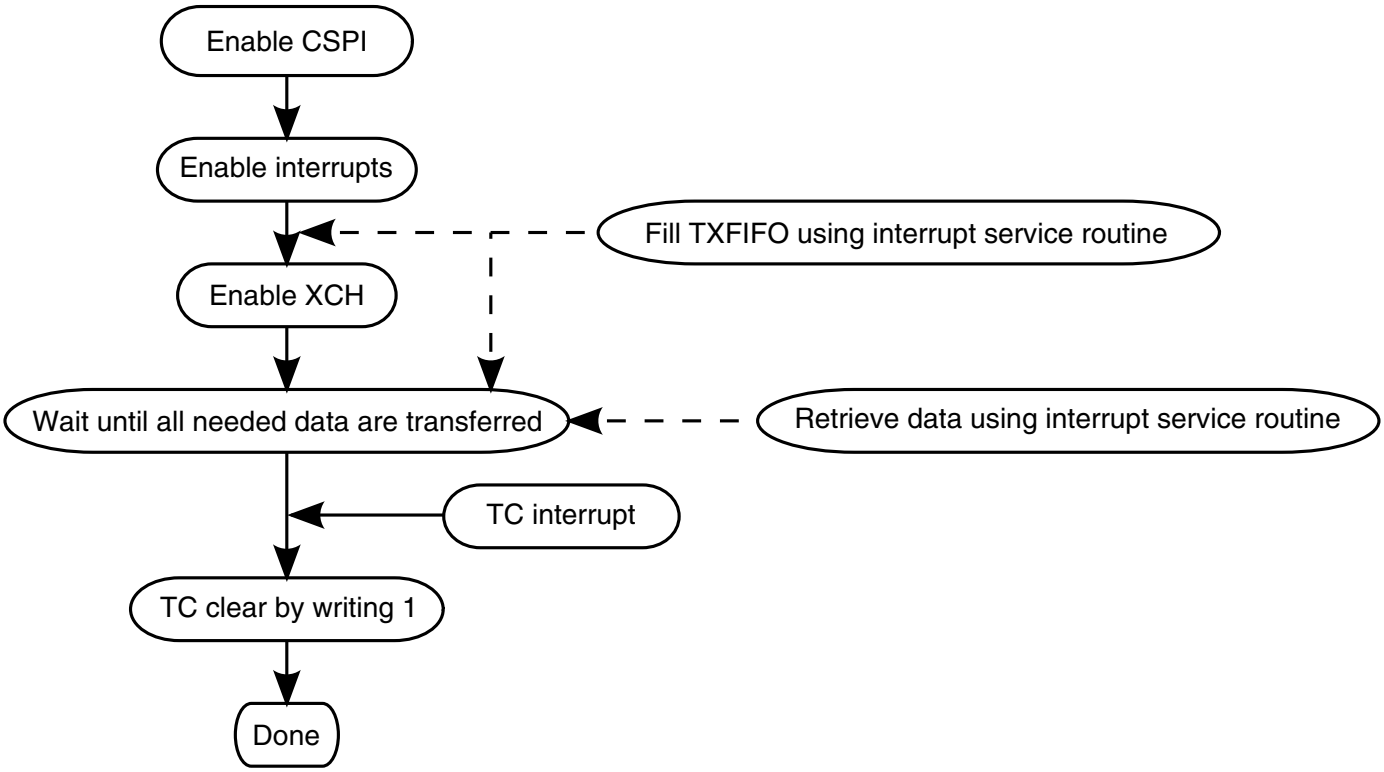


Figure 22-12. Program Sequence of SPI Burst Using Interrupt Control

22.3.9 DMA

DMA control provides another method to utilize the FIFOs in the ECSPI. By using DMA request and acknowledge signals, larger amounts of data can be transferred, and will reduce interrupts and host processor loading. When the appropriate conditions are matched, the block will send out a DMA request.

The DMA can deal with the following conditions:

- TXFIFO empty
- TXFIFO data request
- RXFIFO data request
- RXFIFO full

The figure below shows a program sequence of SPI bursts using DMA control.

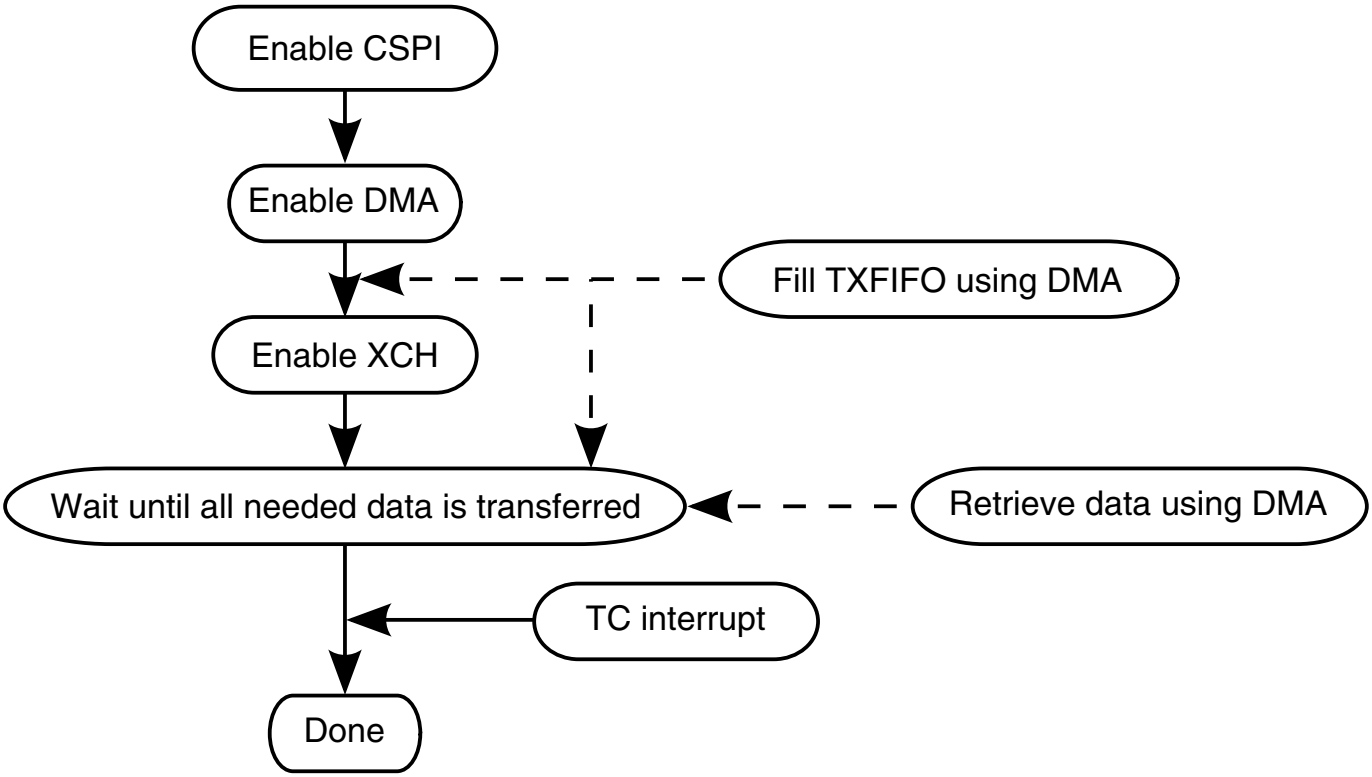


Figure 22-13. Program Sequence of SPI Burst Using DMA

22.3.10 Byte Order

The ECSPI does not support byte re-ordering in hardware.

22.4 Initialization

This section provides initialization information for ECSPI.

To initialize the block:

1. Clear the EN bit in ECSPI_CONREG to reset the block.
2. Enable the clocks for ECSPI.
3. Set the EN bit in ECSPI_CONREG to put ECSPI out of reset.
4. Configure corresponding IOMUX for ECSPI external signals.
5. Configure registers of ECSPI properly according to the specifications of the external SPI device.

22.5 Applications

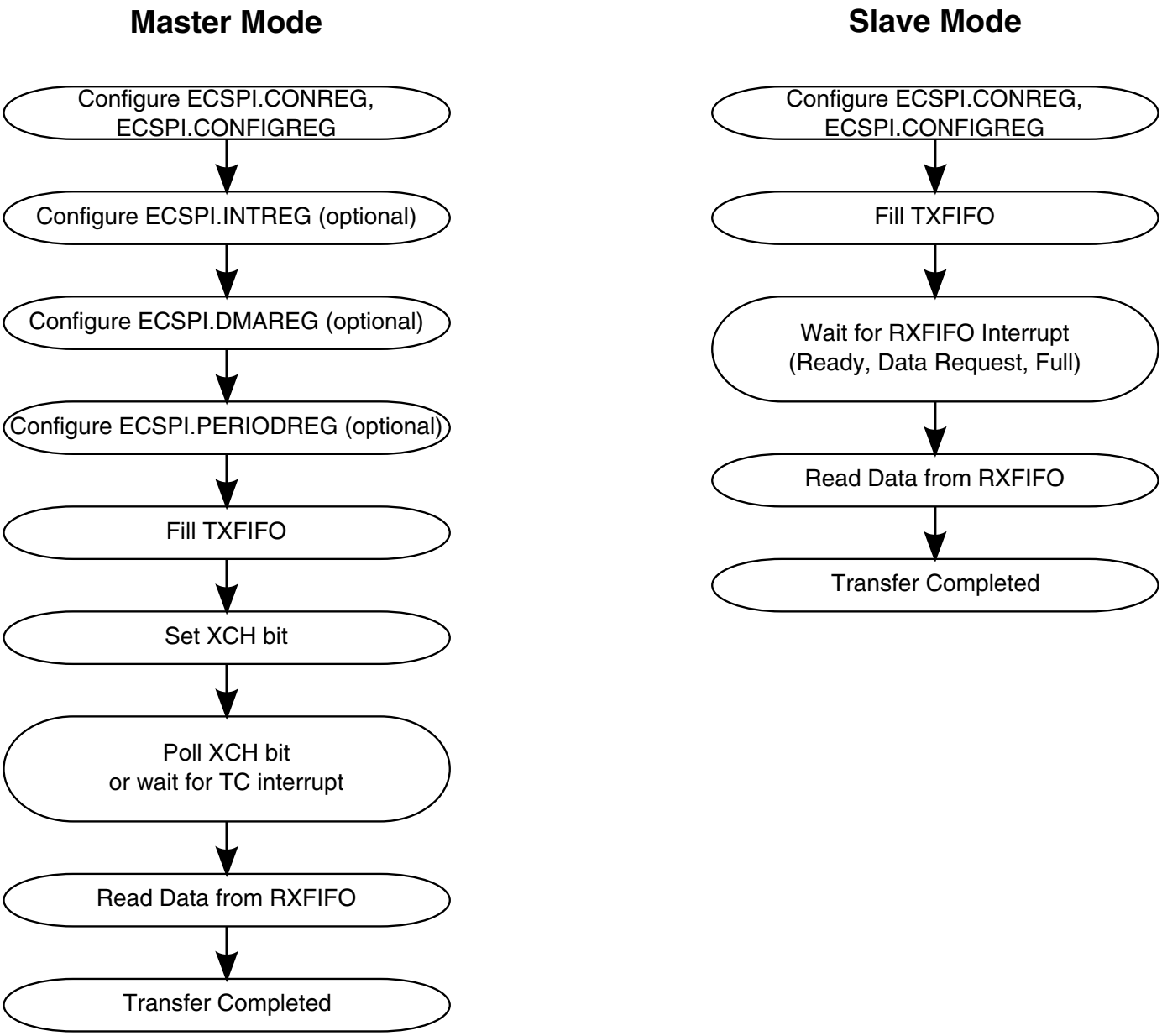


Figure 22-14. Flowchart of the ECSPI Operation

22.6 Programmable Registers

This section includes the block memory map and detailed descriptions of all registers. For the base address of a particular block instantiation, see the system memory map.

ECSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5001_0000	Receive Data Register (ECSPI-1_RXDATA)	32	R	0000_0000h	22.6.1/ 1072
5001_0004	Transmit Data Register (ECSPI-1_TXDATA)	32	W	0000_0000h	22.6.2/ 1073
5001_0008	Control Register (ECSPI-1_CONREG)	32	R/W	0000_0000h	22.6.3/ 1074
5001_000C	Config Register (ECSPI-1_CONFIGREG)	32	R/W	0000_0000h	22.6.4/ 1076
5001_0010	Interrupt Control Register (ECSPI-1_INTREG)	32	R/W	0000_0000h	22.6.5/ 1078
5001_0014	DMA Control Register (ECSPI-1_DMAREG)	32	R/W	0000_0000h	22.6.6/ 1079
5001_0018	Status Register (ECSPI-1_STATREG)	32	R/W	0000_0003h	22.6.7/ 1081
5001_001C	Sample Period Control Register (ECSPI-1_PERIODREG)	32	R/W	0000_0000h	22.6.8/ 1082
5001_0020	Test Control Register (ECSPI-1_TESTREG)	32	R/W	0000_0000h	22.6.9/ 1083
5001_0040	Message Data Register (ECSPI-1_MSGDATA)	32	W	0000_0000h	22.6.10/ 1084
63FA_C000	Receive Data Register (ECSPI-2_RXDATA)	32	R	0000_0000h	22.6.1/ 1072
63FA_C004	Transmit Data Register (ECSPI-2_TXDATA)	32	W	0000_0000h	22.6.2/ 1073
63FA_C008	Control Register (ECSPI-2_CONREG)	32	R/W	0000_0000h	22.6.3/ 1074
63FA_C00C	Config Register (ECSPI-2_CONFIGREG)	32	R/W	0000_0000h	22.6.4/ 1076
63FA_C010	Interrupt Control Register (ECSPI-2_INTREG)	32	R/W	0000_0000h	22.6.5/ 1078
63FA_C014	DMA Control Register (ECSPI-2_DMAREG)	32	R/W	0000_0000h	22.6.6/ 1079

Table continues on the next page...

ECSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FA_C018	Status Register (ECSPI-2_STATREG)	32	R/W	0000_0003h	22.6.7/ 1081
63FA_C01C	Sample Period Control Register (ECSPI-2_PERIODREG)	32	R/W	0000_0000h	22.6.8/ 1082
63FA_C020	Test Control Register (ECSPI-2_TESTREG)	32	R/W	0000_0000h	22.6.9/ 1083
63FA_C040	Message Data Register (ECSPI-2_MSGDATA)	32	W	0000_0000h	22.6.10/ 1084

22.6.1 Receive Data Register (ECSPIx_RXDATA)

The Receive Data register (ECSPI_RXDATA) is a read-only register that forms the top word of the 64 x 32 receive FIFO. This register holds the data received from an external SPI device during a data transaction. Only word-sized read operations are allowed.

Addresses: ECSPI-1_RXDATA is 5001_0000h base + 0h offset = 5001_0000h

ECSPI-2_RXDATA is 63FA_C000h base + 0h offset = 63FA_C000h

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

ECSPIx_RXDATA field descriptions

Field	Description
31–0 ECSPI_RXDATA	Receive Data. This register holds the top word of the receive data FIFO. The FIFO is advanced for each read of this register. The data read is undefined when the Receive Data Ready (RR) bit in the Interrupt Control/Status register is cleared. Zeros are read when ECSPI is disabled.

22.6.2 Transmit Data Register (ECSPIx_TXDATA)

The Transmit Data (ECSPI_TXDATA) register is a write-only data register that forms the bottom word of the 64 x 32 TXFIFO. The TXFIFO can be written to as long as it is not full, even when the SPI Exchange bit (XCH) in ECSPI_CONREG is set. This allows software to write to the TXFIFO during a SPI data exchange process. Writes to this register are ignored when the ECSPI is disabled (ECSPI_CONREG[EN] bit is cleared).

Addresses: ECSPI-1_TXDATA is 5001_0000h base + 4h offset = 5001_0004h

ECSPI-2_TXDATA is 63FA_C000h base + 4h offset = 63FA_C004h

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

ECSPIx_TXDATA field descriptions

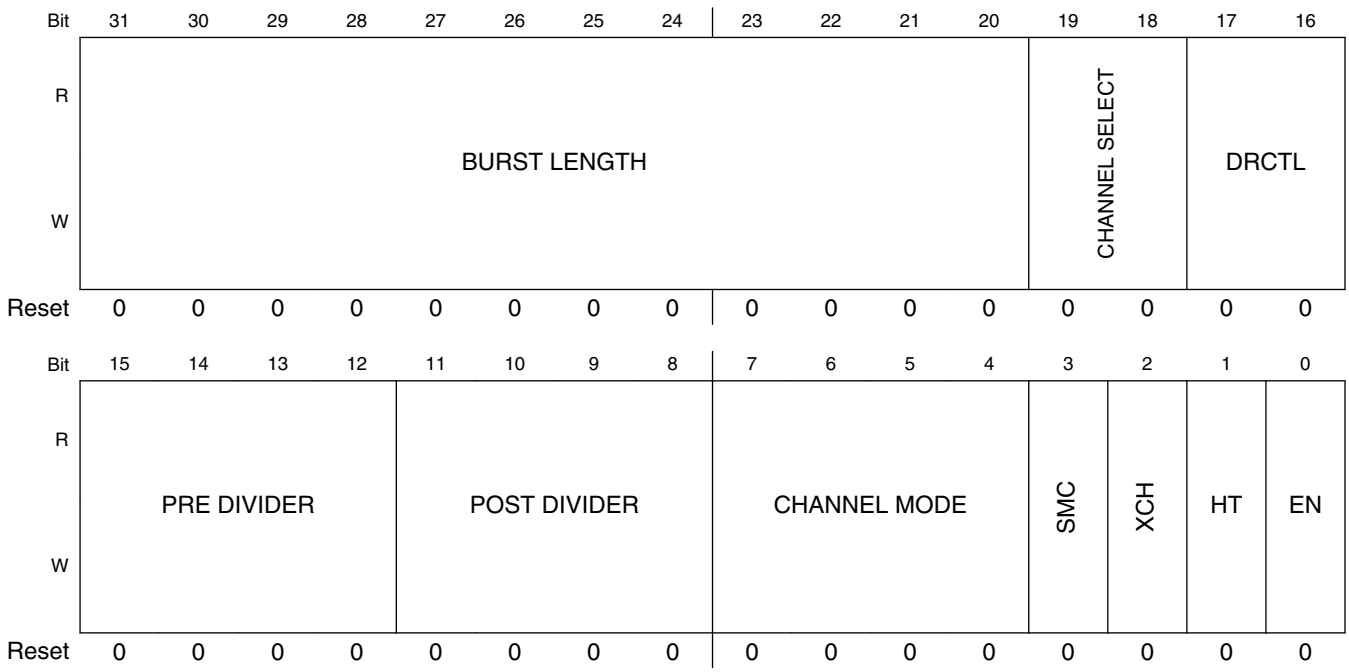
Field	Description
31–0 ECSPI_TXDATA	Transmit Data. This register holds the top word of data loaded into the FIFO. Data written to this register must be a word operation. The number of bits actually transmitted is determined by the BIT_COUNT field of the corresponding SPI Control register. If this field contains more bits than the number specified by BIT_COUNT, the extra bits are ignored. For example, to transfer 10 bits of data, a 32-bit word must be written to this register. Bits 9-0 are shifted out and bits 31-10 are ignored. When the ECSPI is operating in Slave mode, zeros are shifted out when the FIFO is empty. Zeros are read when ECSPI is disabled.

22.6.3 Control Register (ECSPIx_CONREG)

The Control Register (ECSPI_CONREG) allows software to enable the ECSPI , configure its operating modes, specify the divider value, and SPI_RDY control signal, and define the transfer length.

Addresses: ECSPI-1_CONREG is 5001_0000h base + 8h offset = 5001_0008h

ECSPI-2_CONREG is 63FA_C000h base + 8h offset = 63FA_C008h



ECSPIx_CONREG field descriptions

Field	Description
31–20 BURST LENGTH	Burst Length. This field defines the length of a SPI burst to be transferred. The Chip Select (SS) will remain asserted until all bits in a SPI burst are shifted out. A maximum of 2^12 bits can be transferred in a single SPI burst. In master mode, it controls the number of bits per SPI burst. Since the shift register always loads 32-bit data from transmit FIFO, only the n least-significant (n = BURST LENGTH + 1) will be shifted out. The remaining bits will be ignored. In slave mode, only when SS_CTL is cleared, this field will take effect in the transfer. Number of Valid Bits in a SPI burst. 0x000 A SPI burst contains the 1 LSB in a word. 0x001 A SPI burst contains the 2 LSB in a word. 0x002 A SPI burst contains the 3 LSB in a word. 0x01F A SPI burst contains all 32 bits in a word. 0x020 A SPI burst contains the 1 LSB in first word and all 32 bits in second word. 0x021 A SPI burst contains the 2 LSB in first word and all 32 bits in second word.

Table continues on the next page...

ECSPIx_CONREG field descriptions (continued)

Field	Description
	0xFFE A SPI burst contains the 31 LSB in first word and 2^7 -1 words. 0xFFF A SPI burst contains 2^7 words.
19–18 CHANNEL SELECT	SPI CHANNEL SELECT bits. Select one of four external SPI Master/Slave Devices. In master mode, these two bits select the external slave devices by asserting the Chip Select (SSn) outputs. Only the selected Chip Select (SSn) signal can be active at a given time; the remaining three signals will be negated. 00 Channel 0 is selected. Chip Select 0 (SS0) will be asserted. 01 Channel 1 is selected. Chip Select 1 (SS1) will be asserted. 10 Channel 2 is selected. Chip Select 2 (SS2) will be asserted. 11 Channel 3 is selected. Chip Select 3 (SS3) will be asserted.
17–16 DRCTL	SPI Data Ready Control. This field selects the utilization of the $\overline{\text{SPI\_RDY}}$ signal in master mode. ECSPI checks this field before it starts an SPI burst. 00 The $\overline{\text{SPI\_RDY}}$ signal is a don't care. 01 Burst will be triggered by the falling edge of the SPI_RDY signal (edge-triggered). 10 Burst will be triggered by a low level of the SPI_RDY signal (level-triggered). 11 Reserved.
15–12 PRE DIVIDER	SPI Pre Divider. ECSPI uses a two-stage divider to generate the SPI clock. This field defines the pre-divider of the reference clock. 0000 Divide by 1. 0001 Divide by 2. 0010 Divide by 3. 1101 Divide by 14. 1110 Divide by 15. 1111 Divide by 16.
11–8 POST DIVIDER	SPI Post Divider. ECSPI uses a two-stage divider to generate the SPI clock. This field defines the post-divider of the reference clock using the equation: 2^{n} . 0000 Divide by 1. 0001 Divide by 2. 0010 Divide by 4. 1110 Divide by 2^{14} . 1111 Divide by 2^{15} .
7–4 CHANNEL MODE	SPI CHANNEL MODE selects the mode for each SPI channel. CHANNEL MODE[3] is for SPI channel 3. CHANNEL MODE[2] is for SPI channel 2. CHANNEL MODE[1] is for SPI channel 1. CHANNEL MODE[0] is for SPI channel 0. 0 Slave mode. 1 Master mode.
3 SMC	Start Mode Control. This bit applies only to channels configured in Master mode (CHANNEL MODE = 1).

Table continues on the next page...

ECSPIx_CONREG field descriptions (continued)

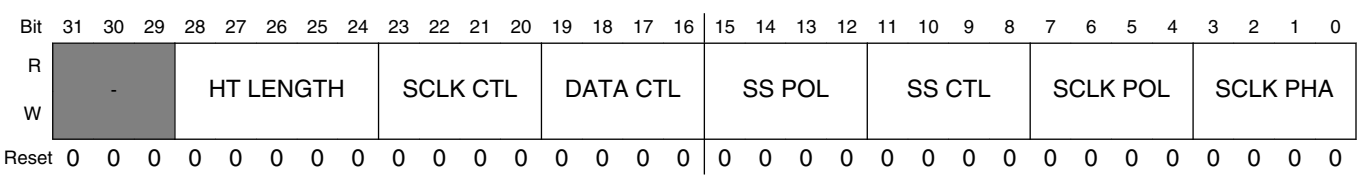
Field	Description
	It controls how the ECSPI starts a SPI burst, either through the SPI exchange bit, or immediately when the TXFIFO is written to. 0 SPI Exchange Bit (XCH) controls when a SPI burst can start. Setting the XCH bit will start a SPI burst or multiple bursts. This is controlled by the SPI SS Wave Form Select (SS_CTL). Refer to XCH and SS_CTL descriptions. 1 Immediately starts a SPI burst when data is written in TXFIFO.
2 XCH	SPI Exchange Bit. This bit applies only to channels configured in Master mode (CHANNEL MODE = 1). If the Start Mode Control (SMC) bit is cleared, writing a 1 to this bit starts one SPI burst or multiple SPI bursts according to the SPI SS Wave Form Select (SS_CTL). The XCH bit remains set while either the data exchange is in progress, or when the ECSPI is waiting for an active input if SPIRDY is enabled through DRCTL. This bit is cleared automatically when all data in the TXFIFO and the shift register has been shifted out. 0 Idle. 1 Initiates exchange (write) or busy (read).
1 HT	Hardware Trigger Enable. This bit is used in master mode only. It enables hardware trigger (HT) mode. 0 Disable HT mode. 1 Enable HT mode.
0 EN	SPI Block Enable Control. This bit enables the ECSPI. This bit must be set before writing to other registers or initiating an exchange. Writing zero to this bit disables the block and resets the internal logic with the exception of the ECSPI_CONREG. The block's internal clocks are gated off whenever the block is disabled. 0 Disable the block. 1 Enable the block.

22.6.4 Config Register (ECSPIx_CONFIGREG)

The Config Register (ECSPI_CONFIGREG) allows software to configure each SPI channel, configure its operating modes, specify the phase and polarity of the clock, configure the Chip Select (SS), and define the HT transfer length.

Addresses: ECSPI-1_CONFIGREG is 5001_0000h base + Ch offset = 5001_000Ch

ECSPI-2_CONFIGREG is 63FA_C000h base + Ch offset = 63FA_C00Ch



ECSPIx_CONFIGREG field descriptions

Field	Description
31–29 -	Reserved
28–24 HT LENGTH	HT LENGTH. This field defines the message length in HT Mode. The length in bits of one message is (HT LENGTH + 1).
23–20 SCLK CTL	SCLK CTL. This field controls the inactive state of SCLK for each SPI channel. SCLK CTL[3] is for SPI channel 3. SCLK CTL[2] is for SPI channel 2. SCLK CTL[1] is for SPI channel 1. SCLK CTL[0] is for SPI channel 0. 0 Stay high. 1 Stay low.
19–16 DATA CTL	DATA CTL. This field controls inactive state of the data line for each SPI channel. DATA CTL[3] is for SPI channel 3. DATA CTL[2] is for SPI channel 2. DATA CTL[1] is for SPI channel 1. DATA CTL[0] is for SPI channel 0. 0 Stay high. 1 Stay low.
15–12 SS POL	SPI SS Polarity Select. In both Master and Slave modes, this field selects the polarity of the Chip Select (SS) signal. SS POL[3] is for SPI channel 3. SS POL[2] is for SPI channel 2. SS POL[1] is for SPI channel 1. SS POL[0] is for SPI channel 0. 0 Active low. 1 Active high.
11–8 SS CTL	SPI SS Wave Form Select. In master mode, this field controls the output wave form of the Chip Select (SS) signal when the SMC (Start Mode Control) bit is cleared. The SS_CTL are ignored if the SMC bit is set. SS CTL[3] is for SPI channel 3. SS CTL[2] is for SPI channel 2. SS CTL[1] is for SPI channel 1. SS CTL[0] is for SPI channel 0. In slave mode, this bit controls when the SPI burst is completed. 1 An SPI burst is completed by the Chip Select (SS) signal edges. (SSPOL = 0: rising edge; SSPOL = 1: falling edge) The RXFIFO is advanced whenever a Chip Select (SS) signal edge is detected or the shift register contains 32-bits of valid data. 0 In master mode - only one SPI burst will be transmitted.

Table continues on the next page...

ECSPIx_CONFIGREG field descriptions (continued)

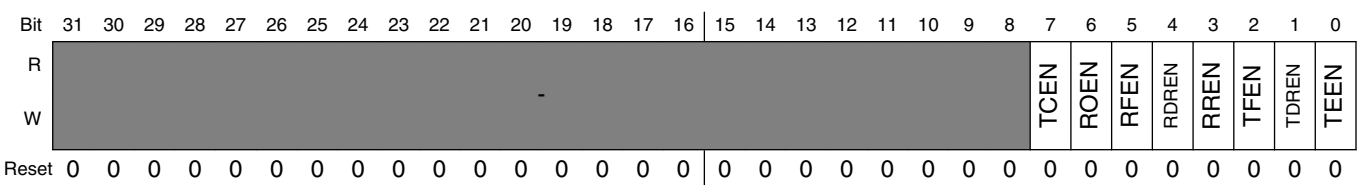
Field	Description
	1 In master mode - Negate Chip Select (SS) signal between SPI bursts. Multiple SPI bursts will be transmitted. The SPI transfer will automatically stop when the TXFIFO is empty. 0 In slave mode - an SPI burst is completed when the number of bits received in the shift register is equal to (BURST LENGTH + 1). Only the n least-significant bits (n = BURST LENGTH[4:0] + 1) of the first received word are valid. All bits subsequent to the first received word in RXFIFO are valid. 1 In slave mode - an SPI burst is completed by the Chip Select (SS) signal edges. (SSPOL = 0: rising edge; SSPOL = 1: falling edge) The RXFIFO is advanced whenever a Chip Select (SS) signal edge is detected or the shift register contains 32-bits of valid data.
7–4 SCLK POL	SPI Clock Polarity Control. This field controls the polarity of the SCLK signal. See Figure 22-10 for more information. SCLK POL[3] is for SPI channel 3. SCLK POL[2] is for SPI channel 2. SCLK POL[1] is for SPI channel 1. SCLK POL[0] is for SPI channel 0. 0 Active high polarity (0 = Idle). 1 Active low polarity (1 = Idle).
3–0 SCLK PHA	SPI Clock/Data Phase Control. This field controls the clock/data phase relationship. See Figure 22-10 for more information. SCLK PHA[3] is for SPI channel 3. SCLK PHA[2] is for SPI channel 2. SCLK PHA[1] is for SPI channel 1. SCLK PHA[0] is for SPI channel 0. 0 Phase 0 operation. 1 Phase 1 operation.

22.6.5 Interrupt Control Register (ECSPIx_INTREG)

The Interrupt Control Register (ECSPI_INTREG) enables the generation of interrupts to the host processor. If the ECSPI is disabled, this register reads zero.

Addresses: ECSPI-1_INTREG is 5001_0000h base + 10h offset = 5001_0010h

ECSPI-2_INTREG is 63FA_C000h base + 10h offset = 63FA_C010h



ECSPIx_INTREG field descriptions

Field	Description
31–8 -	Reserved
7 TCEN	Transfer Completed Interrupt enable. This bit enables the Transfer Completed Interrupt. 0 Disable 1 Enable
6 ROEN	RXFIFO Overflow Interrupt enable. This bit enables the RXFIFO Overflow Interrupt. 0 Disable 1 Enable
5 RFEN	RXFIFO Full Interrupt enable. This bit enables the RXFIFO Full Interrupt. 0 Disable 1 Enable
4 RDREN	RXFIFO Data Request Interrupt enable. This bit enables the RXFIFO Data Request Interrupt when the number of data entries in the RXFIFO is greater than RX THRESHOLD. 0 Disable 1 Enable
3 RREN	RXFIFO Ready Interrupt enable. This bit enables the RXFIFO Ready Interrupt. 0 Disable 1 Enable
2 TFEN	TXFIFO Full Interrupt enable. This bit enables the TXFIFO Full Interrupt. 0 Disable 1 Enable
1 TDREN	TXFIFO Data Request Interrupt enable. This bit enables the TXFIFO Data Request Interrupt when the number of data entries in the TXFIFO is less than or equal to TX THRESHOLD. 0 Disable 1 Enable
0 TEEN	TXFIFO Empty Interrupt enable. This bit enables the TXFIFO Empty Interrupt. 0 Disable 1 Enable

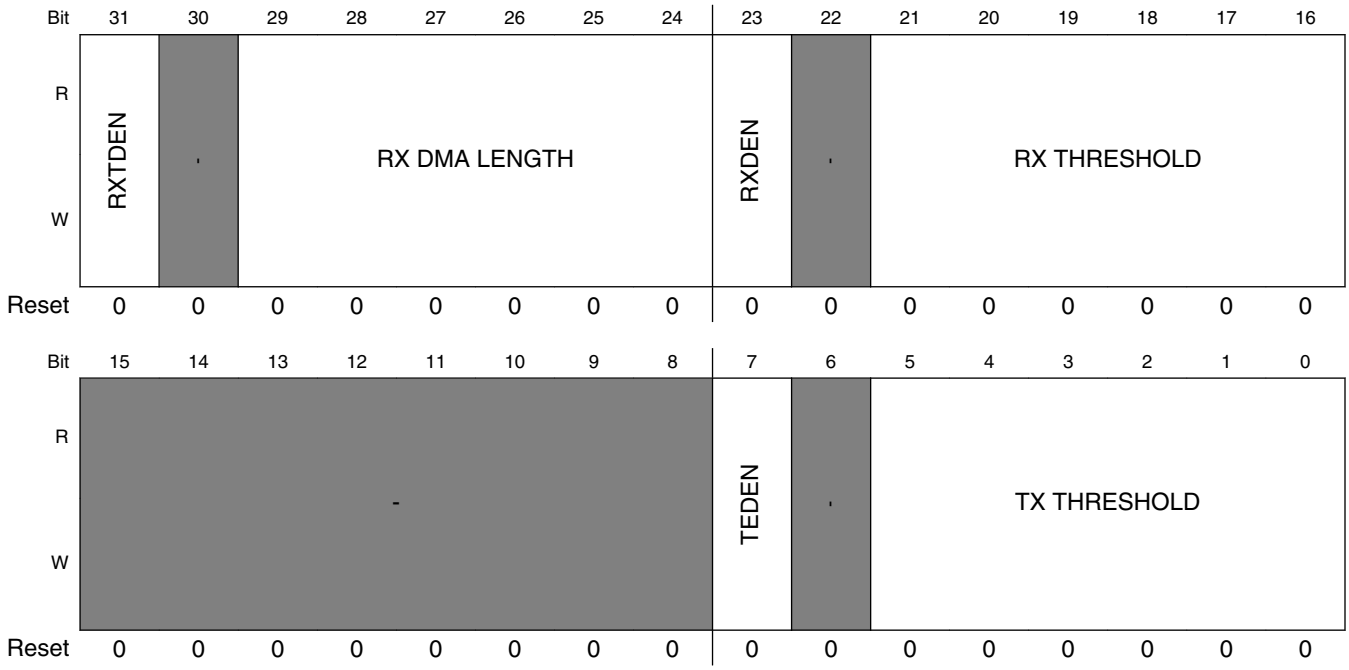
22.6.6 DMA Control Register (ECSPIx_DMAREG)

The Direct Memory Access Control Register (ECSPI_DMAREG) provides software a way to use an on-chip DMA controller for ECSPI data. Internal DMA request signals enable direct data transfers between the ECSPI FIFOs and system memory. The ECSPI sends out DMA requests when the appropriate FIFO conditions are matched.

If the ECSPI is disabled, this register is read as 0.

Addresses: ECSPI-1_DMAREG is 5001_0000h base + 14h offset = 5001_0014h

ECSPI-2_DMAREG is 63FA_C000h base + 14h offset = 63FA_C014h



ECSPIx_DMAREG field descriptions

Field	Description
31 RXTDEN	RXFIFO TAIL DMA Request Enable. This bit enables an internal counter that is increased at each read of the RXFIFO. This counter is cleared automatically when it reaches RX DMA LENGTH. If the number of words remaining in the RXFIFO is greater than or equal to RX DMA LENGTH, a DMA request is generated even if it is less than or equal to RX THRESHOLD. 0 Disable 1 Enable
30 -	Reserved
29–24 RX DMA LENGTH	RX DMA LENGTH. This field defines the burst length of a DMA operation. Applies only when RXTDEN is set.
23 RXDEN	RXFIFO DMA Request Enable. This bit enables/disables the RXFIFO DMA Request. 0 Disable 1 Enable
22 -	Reserved
21–16 RX THRESHOLD	RX THRESHOLD. This field defines the FIFO threshold that triggers a RX DMA/INT request. A RX DMA/INT request is issued when the number of data entries in the RXFIFO is greater than RX THRESHOLD.
15–8 -	Reserved
7 TEDEN	TXFIFO Empty DMA Request Enable. This bit enables/disables the TXFIFO Empty DMA Request.

Table continues on the next page...

ECSPIx_DMAREG field descriptions (continued)

Field	Description
	0 Disable 1 Enable
6 -	Reserved
5–0 TX THRESHOLD	TX THRESHOLD. This field defines the FIFO threshold that triggers a TX DMA/INT request. A TX DMA/INT request is issued when the number of data entries in the TXFIFO is greater than TX THRESHOLD.

22.6.7 Status Register (ECSPIx_STATREG)

The ECSPI Status Register (ECSPI_STATREG) reflects the status of the ECSPI's operating condition. If the ECSPI is disabled, this register reads 0x0000_0003.

Addresses: ECSPI-1_STATREG is 5001_0000h base + 18h offset = 5001_0018h

ECSPI-2_STATREG is 63FA_C000h base + 18h offset = 63FA_C018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								TC	RO	RF	RDR	RR	TF	TDR	TE
W									w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

ECSPIx_STATREG field descriptions

Field	Description
31–8 -	Reserved
7 TC	Transfer Completed Status bit. Writing 1 to this bit clears it. 0 Transfer in progress. 1 Transfer completed.
6 RO	RXFIFO Overflow. When set, this bit indicates that RXFIFO has overflowed. Writing 1 to this bit clears it. 0 RXFIFO has no overflow. 1 RXFIFO has overflowed.
5 RF	RXFIFO Full. This bit is set when the RXFIFO is full. 0 Not Full. 1 Full.

Table continues on the next page...

ECSPIx_STATREG field descriptions (continued)

Field	Description
4 RDR	RXFIFO Data Request. 0 When RXTDE is set - Number of data entries in the RXFIFO is not greater than RX THRESHOLD. 1 When RXTDE is set - Number of data entries in the RXFIFO is greater than RX THRESHOLD or a DMA TAIL DMA condition exists. 0 When RXTDE is clear - Number of data entries in the RXFIFO is not greater than RX THRESHOLD. 1 When RXTDE is clear - Number of data entries in the RXFIFO is greater than RX THRESHOLD.
3 RR	RXFIFO Ready. This bit is set when one or more words are stored in the RXFIFO. 0 No valid data in RXFIFO. 1 More than 1 word in RXFIFO.
2 TF	TXFIFO Full. This bit is set when if the TXFIFO is full. 0 TXFIFO is not Full. 1 TXFIFO is Full.
1 TDR	TXFIFO Data Request. 0 Number of empty slots in TXFIFO is greater than TX THRESHOLD. 1 Number of empty slots in TXFIFO is not greater than TX THRESHOLD.
0 TE	TXFIFO Empty. This bit is set if the TXFIFO is empty. 0 TXFIFO contains one or more words. 1 TXFIFO is empty.

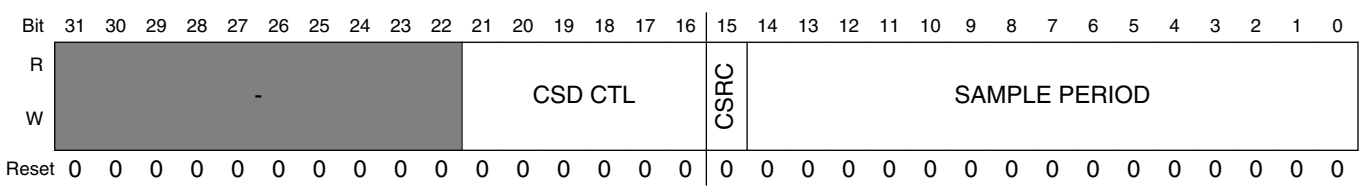
22.6.8 Sample Period Control Register (ECSPIx_PERIODREG)

The Sample Period Control Register (ECSPI_PERIODREG) provides software a way to insert delays (wait states) between consecutive SPI transfers. Control bits in this register select the clock source for the sample period counter and the delay count indicating the number of wait states to be inserted between data transfers.

The delay counts apply only when the current channel is operating in Master mode (ECSPI_CONREG[CHANNEL MODE] = 1).ECSPI_PERIODREG also contains the CSD CTRL field used to insert a delay between the Chip Select's active edge and the first SPI Clock edge.

Addresses: ECSPI-1_PERIODREG is 5001_0000h base + 1Ch offset = 5001_001Ch

ECSPI-2_PERIODREG is 63FA_C000h base + 1Ch offset = 63FA_C01Ch



ECSPIx_PERIODREG field descriptions

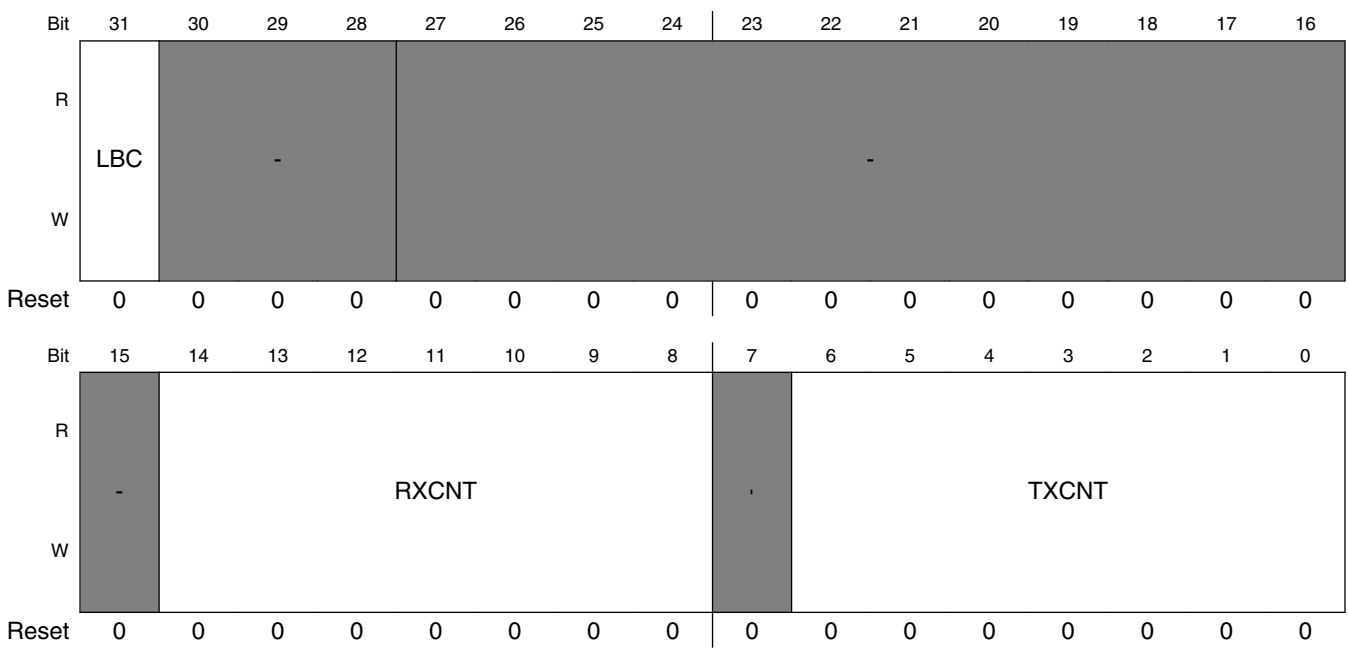
Field	Description
31–22 -	Reserved
21–16 CSD CTL	Chip Select Delay Control bits. This field defines how many SPI clocks will be inserted between the chip select's active edge and the first SPI clock edge. The range is from 0 to 63.
15 CSRC	Clock Source Control. This bit selects the clock source for the sample period counter. 0 SPI Clock (SCLK) 1 Low-Frequency Reference Clock (32.768 KHz)
14–0 SAMPLE PERIOD	Sample Period Control. These bits control the number of wait states to be inserted in data transfers. During the idle clocks, the state of the SS output will operate according to the SS_CTL control field in the ECSPI_CONREG register. 0x0000 0 wait states inserted 0x0001 1 wait state inserted ... ... 0x7FFE 32766 wait states inserted 0x7FFF 32767 wait states inserted

22.6.9 Test Control Register (ECSPIx_TESTREG)

The Test Control Register (ECSPI_TESTREG) provides software a mechanism to internally connect the receive and transmit devices of the ECSPI, and monitor the contents of the receive and transmit FIFOs.

Addresses: ECSPI-1_TESTREG is 5001_0000h base + 20h offset = 5001_0020h

ECSPI-2_TESTREG is 63FA_C000h base + 20h offset = 63FA_C020h



ECSPIx_TESTREG field descriptions

Field	Description
31 LBC	Loop Back Control. This bit is used in Master mode only. When this bit is set, the ECSPI connects the transmitter and receiver sections internally, and the data shifted out from the most-significant bit of the shift register is looped back into the least-significant bit of the Shift register. In this way, a self-test of the complete transmit/receive path can be made. The output pins are not affected, and the input pins are ignored. 0 Not connected. 1 Transmitter and receiver sections internally connected for Loopback.
30–28 -	Reserved, all bits should be ignored.
27–15 -	Reserved
14–8 RXCNT	RXFIFO Counter. This field indicates the number of words in the RXFIFO.
7 -	Reserved
6–0 TXCNT	TXFIFO Counter. This field indicates the number of words in the TXFIFO.

22.6.10 Message Data Register (ECSPIx_MSGDATA)

The Message Data Register (ECSPI_MSGDATA) forms the top word of the 16 x 32 MSG Data FIFO. Only word-size accesses are allowed for this register. Reads to this register return zero, and writes to this register store data in the MSG Data FIFO. See Hardware Trigger (HT) ModeIn hardware trigger (HT) mode, the behavior of the ECSPI is similar to master mode. The differences are in the trigger mechanism and the burst length. .

Addresses: ECSPI-1_MSGDATA is 5001_0000h base + 40h offset = 5001_0040h

ECSPI-2_MSGDATA is 63FA_C000h base + 40h offset = 63FA_C040h

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

ECSPIx_MSGDATA field descriptions

Field	Description
31–0 ECSPI_ MSGDATA	ECSPI_MSGDATA holds the top word of MSG Data FIFO. The MSG Data FIFO is advanced for each write of this register. The data read is zero. The data written to this register is stored in the MSG Data FIFO.

Chapter 23
External Interface Module (EIM)

23.1 Overview

The EIM handles the interface to devices external to the chip, including generation of chip selects, clock and control for external peripherals and memory. It provides asynchronous access to devices with SRAM-like interface and synchronous access to devices with NOR-Flash-like or PSRAM-like interface.

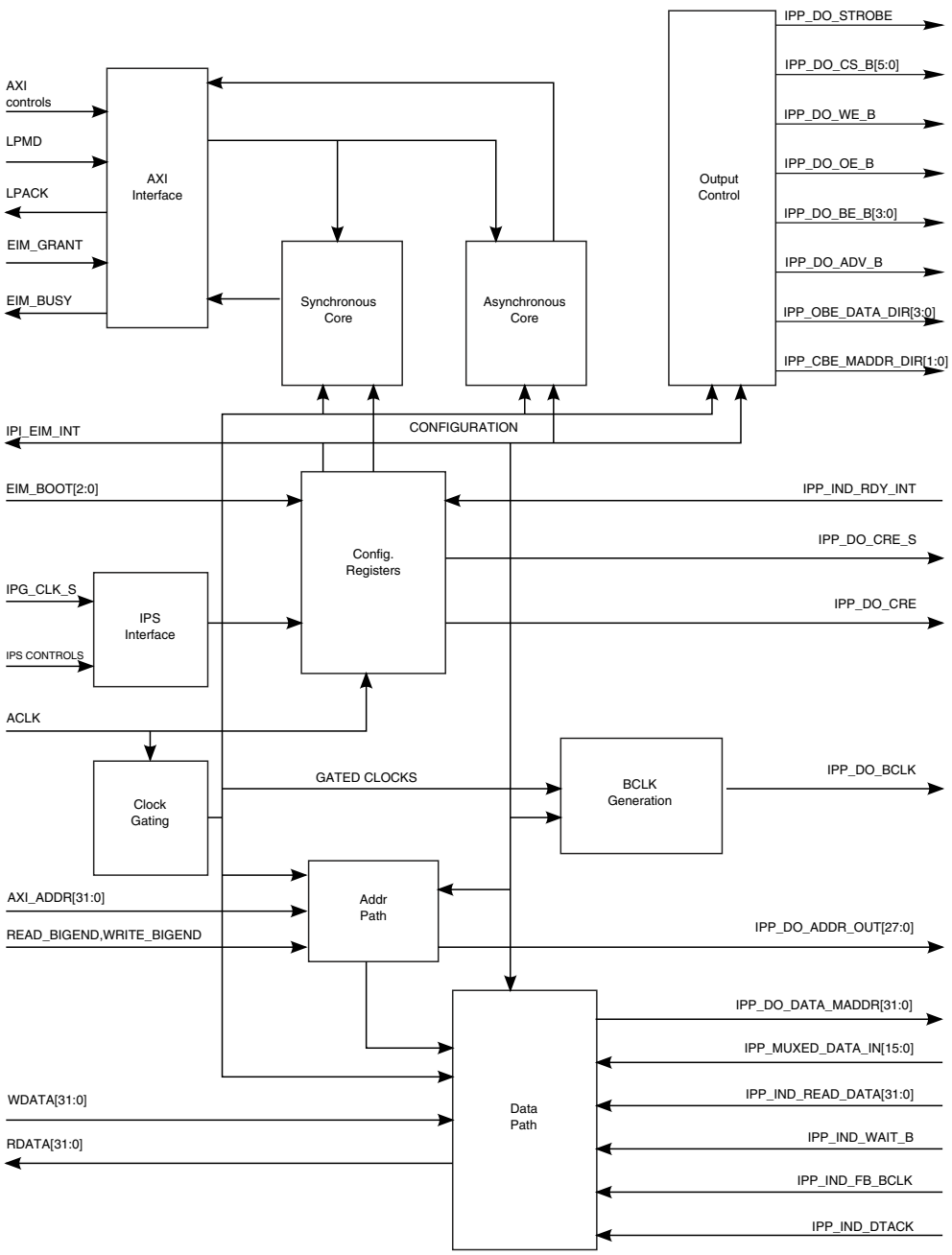


Figure 23-1. EIM Diagram

23.2 Features

- Up to four chip selects for external devices
  - Flexible address decoding. Each chip select memory space determined separately, according to VIA port configuration (see Chip Select Memory Map). Configurable Chip Select 0 base address (by VIA)
  - Individual select signal for each one of the memory space defined. Up to 6 memory spaces may be defined and programmed individually.
  - 28-bit external address bus, max memory size can be 256MByte (2 Gigabit).
- Selectable Write Protection for each Chip Select
- Support for multiplexed address / data bus operation x16 and x32 port size
- Programmable Data Port Size for each Chip Select (x8, x16 and x32)
- Programmable Wait-State generator for each Chip Select, for write and read accesses separately
- Asynchronous accesses with programmable setup and hold times for control signals
- Support for Asynchronous page mode accesses (x16 and x32 port size)
- Independent synchronous Memory Burst Read Mode support for NOR-Flash and PSRAM memories (x16 and x32 port size)
- Independent synchronous Memory Burst Write Mode support for PSRAM and NOR-Flash like memories (CellularRAM™ from Micron, Infineon, and Cypress, OneNAND™ and utRAM™ from Samsung, and COSMORAM™ from Toshiba)
- Support of NAND-Flash devices with NOR-Flash like interface - MDOC™ (M-Systems), OneNAND™ (Samsung)
- Independent programable variable/fix Latency support for read and write synchronous (burst) mode
- Support for Big Endian and Little Endian operation modes per access
- ARM AXI slave interface. One ID at a time support.
- External Interrupt support, RDY_INT signal function as external interrupt
- Boot from external device support according to boot signals, using RDY_INT signal
  - RDY signal support assertion after reset for MDOC™ (M-Systems) device
  - INT signal support assertion after reset for OneNAND™ (Samsung) device

23.3 Modes of Operation

The EIM has the following modes of operation:

- Asynchronous Mode
- Asynchronous Page Mode
- Multiplexed Address/Data mode

- Burst Clock Mode
- Low Power Modes
- Boot Mode

See details in the EIM Operational Modes.

23.3.1 Asynchronous Mode

This is a non-burst mode that is used for SRAM access. In this mode, a single data is read/written with each access (asserted address).

All controls' timings are controlled by preset values in Chip Select Configuration Registers.

23.3.2 Asynchronous Page Read Mode

Setting the APR bit causes the EIM to perform memory burst accesses by emulating page mode operation.

The external address asserts for each piece of data. The initial access timing is according to RWSC field, and the next address assertions timing is according to PAT field. When APR bit is set, RCSN OEN, RADVN and RBEN fields are ignored for burst access to the external device.

The page size can be set via the BL field to 2, 4, 8, 16, or 32 words (the word size is determined by the DSZ field).

23.3.3 Multiplexed Address/Data Mode

In this mode, multiplexing addresses and data bits on the same pins is supported for synchronous/asynchronous accesses to x8/x16/ x32 data width memory devices.

For more information about the pins that drive data/address in 8/16/32 non-muxed mode and 16/32 muxed mode, refer to the EIM Internal Module Multiplexing table in the EIM Internal Pads Allocation chapter of the datasheet.

Table 23-1. EIM multiplexing

Setup	Non Multiplexed Address/Data Mode						Multiplexed Address/ Data mode	
	8 Bit			16 Bit		32 Bit	16 Bit	32 Bit
	MUM = 0, DSZ = 100	MUM = 0, DSZ = 101	MUM = 0, DSZ = 111	MUM = 0, DSZ = 001	MUM = 0, DSZ = 010	MUM = 0, DSZ = 011	MUM = 1, DSZ = 001	MUM = 1, DSZ = 011
A[15:0]	EIM_DA[15:0]	EIM_DA[15:0]	EIM_DA[15:0]	EIM_DA[15:0]	EIM_DA[15:0]	EIM_DA[15:0]	EIM_DA[15:0]	EIM_DA[15:0]
A[27:16]	Pin multiplexed over DISP pins.	Pin multiplexed over DISP pins.	Pin multiplexed over DISP pins.	Pin multiplexed over DISP pins.	Pin multiplexed over DISP pins.	Pin multiplexed over DISP pins.	Pin multiplexed over DISP pins.	Pin multiplexed over DISP pins.
D[7:0], EIM_EB0	Pin multiplexed over EPDC pins.	-	-	Pin multiplexed over EPDC pins.	-	Pin multiplexed over EPDC pins.	EIM_DA[7:0]	EIM_DA[7:0]
D[15:8], EIM_EB1	-	Pin multiplexed over EPDC pins.	-	Pin multiplexed over EPDC pins.	-	Pin multiplexed over EPDC pins.	EIM_DA[15:8]	EIM_DA[15:8]
D[23:16], EIM_EB2	-	-	-	-	Pin Multiplexed over EPDC pins	Pin Multiplexed over EPDC pins	-	Pin multiplexed over EPDC pins.
D[31:24], EIM_EB3	-	-	Pin Multiplexed over EPDC pins	-	Pin Multiplexed over EPDC pins	Pin Multiplexed over EPDC pins	-	Pin multiplexed over EPDC pins.

23.3.4 Burst Clock Mode

The controller has the ability to support a burst synchronous operations in a various frequencies, depending on the frequency of the input clock supplied by the system (EIM clock).

The EIM clock can be divided by one, two, three or four, and its frequency can be changed according to the requirements. Variable and fix latency are supported for this mode, according to the external device requirements.

- Synchronous read mode. This is a burst mode, which is used for reading from Flash/ PSRAM memory devices. In this mode, after address assertion a burst of sequential data can be read. Data exchange is carried out according to BCLK being generated by EIM. An access is delayed according to external WAIT_B signal assertion (signal from the memory device).
- Synchronous write mode. A burst mode used for accessing external devices, which support synchronous write type of access (PSRAM protocol). In this mode, after

address assertion a burst of sequential data can be written to the external device. Access may be delayed according to WAIT_B signal assertion (signal from the memory device) before first piece of data arrived to the external device.

NOTE

Maximum frequency of the EIM main clock is 133Mhz. It may be reduced by the system for special cases of external devices, which demand a different frequency then integer division of the 133MHz clock.

23.3.5 Low Power Modes

The Input Clock is gated by ACT_CS bits. When all the ACT_CS are negated (all CS disable) the internal clock is turned off; awready/wready & arready signal are de-asserted and the master can't access the EIM.

23.3.6 Boot Mode

It is possible to perform a boot operation from external device located on CS0. The configuration of the relevant bits are done with boot Mode signals according to the external device parameters (for example, port size and protocol assertion).

See System Boot for more details.

Table 23-2. EIM Boot configuration

EIM_BOOT bits	EIM affected bits	EIM Register	Boot Value
2	MUM OEA	EIM_CS0GCR1 EIM_CS0RCR1	Configurable by fuses; If SRC.SBMR[BOOT_CFG2[7:6]] = 0b00, value = 0x100 If SRC.SBMR[BOOT_CFG2[7:6]] = 0b00, value = 0x010
[1:0]	DSZ[1:0]	EIM_CS0GCR1	

23.4 External Signal Description

23.4.1 Signals Overview

Table 23-3. Block I/Os

Name	Function	Input / Output	Notes
WEIM_A[27:16]	Address Bus MSB	Output	
WEIM_DA[15:0]	Address/Data Bus LSB	I/O	NFC data lines can be multiplexed on these lines as well
WEIM_BCLK	Burst Clock	Output	
WEIM_CRE	Memory Register Set	Output	
WEIM_CS[5:0]	Chip Selects	Output	
WEIM_DTACK_B	Data Acknowledge	Input	
WEIM_D[31:16]	Data MSB	I/O	
WEIM_EB[3:0]	Enable Byte Signals	Output	
WEIM_LBA	Load Burst Address	Output	
WEIM_OE	Output Enable	Output	
WEIM_RW	Read/Write	Output	
WEIM_WAIT	Memory Ready/Busy/Wait	Input	

23.4.2 Detailed Signal Descriptions

The following is a detailed description of the EIM signals mentioned in Table 23-3.

Table 23-4. Block I/O Descriptions

Name	I/O	Description
WEIM_A[27:16]	O	MSB Address Bus
WEIM_DA[15:0]	I/O	LSB multiplexed Address/Data Bus
WEIM_BCLK	O	Burst Clock (BCLK). This active-high output signal is used to clock external burst-capable devices to synchronize the loading and incrementing of addresses and delivery of burst read and write data to/from the EIM. Its behavior is affected by the BCM field in the EIM_WCR and the SWR, SRD, BCD, and BCS fields of the EIM_CSxGCR1.
WEIM_CRE	O	Used as CRE/PS for CellularRam memory. It is used for the Mode Register Set command. This signal can be configured as active low or active high. See CRE and CREP field descriptions of the EIM_CS*x*GCR1 registers.
WEIM_CS[5:0]	O	Chip Selects. These signals are active-low. Behavior is affected by the RCSA and RCSN fields of the EIM_CSxRCR1 registers and the WCSA and WCSN fields of the EIM_CSxWCR1registers.
WEIM_DTACK_B	I	Data Acknowledge, asynchronous access. This input is used as a data acknowledge signal for single asynchronous accesses.
WEIM_D[31:16]	I/O	MSB Data Bus

Table continues on the next page...

Table 23-4. Block I/O Descriptions (continued)

Name	I/O	Description
WEIM_EB[3:0]	O	Byte Enable. These active-low output signals indicate valid data bytes for the current access. They may be configured to assert for write cycles only. EIM_EB[0] corresponds to DATA_OUT[7:0] EIM_EB[1] corresponds to DATA_OUT[15:8] EIM_EB[2] corresponds to DATA_OUT[23:16] EIM_EB[3] corresponds to DATA_OUT[31:24]. For asynchronous write accesses, behavior is affected by the WBEA and WBEN fields of the EIM_CS1WCR1-EIM_CS5WCR1 Registers. On synchronous or asynchronous read accesses, these signals are always asserted at the start of the access and negated at end of the access.
WEIM_LBA	O	Address Valid. This active-low output signal is asserted during burst mode accesses to cause the external burst capable device to load a new starting burst address. Assertion of LBA indicates that a valid address is present on the address bus. Its behavior is affected by the SWR, SRD, BCD, and BCS fields of the EIM_CSxGCR1 registers, the RADVA and RADVN fields of the EIM_CSxRCR1 registers, and the WADVA and WADVN fields of the EIM_CSxWCR1 registers. In asynchronous mode, LBA length is affected by the RADVA, WADVA, RADVN, and WADVN fields. Minimum length of LBA signal in all modes is one EIM clock cycle.
WEIM_OE	O	Output Enable. This active-low output signal indicates the bus access is a read and enables external devices to drive the data bus with read data. Its behavior is affected by the OEA and OEN bit fields in the Chip Select Configuration Registers.
WEIM_RW	O	Memory Write Enable. This active-low output signal indicates the bus access is a write and enables external devices to sample the data bus. Its behavior is affected by the WEA and WEN bit fields in the Chip Select Configuration Registers.
WEIM_WAIT	I	Ready/Busy/Wait signal. This active-low input signal is asserted by external burst capable devices which support fixed or variable latency of data. It is serviced in synchronous mode only (EIM_CSxGCR1[SWR, SRD] =1). WAIT will have a pull up resistor in I/O. The signal indicates whether the External device is ready for data transaction or not. Busy cycles (or wait cycles) of the external device can occur at the start of a Burst access or at page boundary crossover. **NOTE:** For burst devices, WAIT output should be configured to change one cycle before data is ready (before delay). **NOTE:** Some External devices may not use this input signal for ready state indication (fix latency without WAIT signal monitoring). For these devices EIM should be configured accordingly (see RFL, WFL, and PSZ field descriptions). **NOTE:** This is same as what is shown in IP_IND_WAIT_B

23.4.3 Other Important Block I/O Signals Internal to the SoC

The following table provides a description of other signals which are internal to the i.MX50 that are important to understand the function of EIM.

Name	I/O	Description
WEIM_FB_BCLK	Input	Burst Clock Feedback. This block input is used to sample read data during high transfer speeds. The signal provides feedback from the I/O pad of the BCLK output pin and tends to align more closely with data from the external memory device.
EIM_BOOT[2:0]	Input	EIM Boot Configuration. These block inputs determine the reset state of DSZ[1:0] and MUM. See Table 5-4 for detailed description.
ACLK	Input	AXI clock, maximum frequency 133 Mhz
IPG_CLK_S	Input	EIM module IPG clock
RST_B	Input	Active low HW reset
EIM_WARM_RESET	Input	Warm Reset. If this signal is asserted the rst_b will reset only the internal FF and state machine while S/W registers will keep their current state.This signal is active high signal.

23.5 Chip Select Memory Map

Table 23-6. EIM Chip Select Memory Map

Address	Space Size	Use	Access
EIM_NFC_BASE/ACT_CS/ADDRS0 inputs	0MB-256MB	CS0 memory region	R/W
ACT_CS/ADDRS1 inputs	0MB-256MB	CS1 memory region	R/W
ACT_CS/ADDRS2 inputs	0MB-256MB	CS2 memory region	R/W
ACT_CS/ADDRS3 inputs	0MB-256MB	CS3 memory region	R/W

23.6 Functional Description

This section provides the functional description for the EIM.

23.6.1 Clocks

The EIM has four input clocks.

- ACLK: EIM clock (main clock, AXI clock) with a Max frequency of 133Mhz. Can be gated externally when there is no active AXI access.
- ACLK_SLOW: EIM all time running ACLK. Used for FF that must be active even when EIM is in low power down mode to provide clock for lpack/lpmd registers, IP registers and IP to AXI sync registers.
- IPG_CLK_S: IPG clock for IP accesses. IP registers are activated by ACLK_SLOW clock. BCLK is a clock created from EIM clock for External device usage. Integer division by 1, 2, 3 and 4 of the clock can be use with BCD bit field configuration,

according to external devices demands. EIM clock frequency may be reduced for lower frequency support which cannot be achieved via BCD bit field.
- FB_CLK: BCLK after pads delays.

23.6.2 Bus Sizing Configuration

The EIM supports byte, half word and word operands allowing access to x8, x16, x32 ports. It can be address/data multiplexed in x16, x32 ports. The port size is programmable via the DSZ bit field in the corresponding Chip Select Configuration Register. An 8-bit port can reside in each one of the bytes of the data bus. A 16-bit port can reside on the lower 16 bits of the data bus, DATA_IN/OUT[15:0] or on the higher 16 bits of the data bus, DATA_IN/OUT[31:16] (see Table 23-1).

In the case of a multi-cycle transfer, the lower two address bits (ADDR[1:0]) are incremented appropriately. The EIM address bus is configured according to DSZ bit field and AUS bits. There is either one bit (for x16 port size) or two bits (for x32 port size) right shift of the address bits (only when AUS=0) and no bit shift when AUS = 1 or DSZ[2] = 1.

The EIM has a data multiplexer which takes the four bytes of the AXI data bus and routes them to their required positions to properly interface to memory.

NOTE

A word access to or from a x16 port requires two external bus cycles to complete the transfer.

A word access to or from a x8 port requires four external bus cycles to complete the transfer.

23.6.2.1 8 BIT PORT SUPPORT

EIM has limited support for mot68000 & intel 386 protocols.

23.6.2.1.1 MOTOROLA 68000

EIM has limited support for mot68000 protocol. Only basic read or write asynchronous operations are supported.

The following operations are not supported:

- Read modify write
- Sync access

- All special accesses (ARM platform space, bus arbitration, bus control, bus error & reset operations)
- FC outputs

23.6.2.1.2 INTEL 386

EIM has limited support for intel 386 protocol. Only basic read or write async non-pipelined operations are supported.

The following operations are not supported:

- Other bus cycles (interrupt, halt & refresh)
- Bus lock
- M/IO, DC, LBA, NA, REFRESH & BS8 signals

23.6.3 EIM Operational Modes

EIM has the following main operational modes selected by control bit fields settings. For details, see the bit field descriptions of SWR / SRD / MUM. All modes are supported in with 8-, 16- or 32-bit port configuration, according to DSZ bit field.

Table 23-7. EIM Operation Modes Field Settings

Control bit fields			Brief mode description
MUM	SRD	SWR	
0	0	0	Asynchronous write / Asynchronous read for APR=0 / Asynchronous page read for APR=1, none multiplexed
		1	Synchronous write/ Asynchronous read or APR=0 / Asynchronous page read for APR=1,none multiplexed
	1	0	Asynchronous write/Synchronous read none multiplexed
		1	Synchronous write/read none multiplexed
1	0	0	Asynchronous write/read multiplexed
		1	Synchronous write/ Asynchronous read multiplexed
	1	0	Asynchronous write/Synchronous read multiplexed
		1	Synchronous write/read multiplexed

23.6.4 Burst Mode (Synchronous) Memory Operation

This mode is enabled for read or write access. Bit SWR sets the burst mode for write operations at the corresponding chip select and bit SRD sets it for read operation.

When this mode is set, the controller attempts to translate the Master burst accesses to memory burst accesses, being limited by the memory burst length, predefined by BL value, or memory and Master WRAP/INCR boundary crossing non-matching. Only the first address accessed is displayed by the controller on the external address bus in a memory burst sequence.

EIM may translate from some Master sequential accesses to one or few memory bursts, but not from two Master individual accesses to one memory burst.

For the first access in a memory burst sequence, the EIM asserts $\overline{\text{ADV}}$, causing the external burst device to latch the starting burst address; then toggle the burst clock (BCLK) a predefined number of cycles in order to latch the first unit of data. Subsequent accessed data units can then be burst in fewer clock cycles, realizing an overall increase in bus bandwidth.

NOTE

The BCLK signal toggles only when burst access is executed toward the external device (BCM=1'b0 for normal mode use). It runs with a 50% duty cycle until the end of access is reached. When access is terminated, BCLK stops toggling.

Memory burst accesses are terminated by the EIM whenever it detects the following:

- The specific burst length has executed completely (end of access)
- Write access - missing data in write buffer (Master is delaying the data transfer toward the EIM)
- Next sequential access crosses boundary with unequal condition (wrap/increment, burst length) on the Master and memory
- Current memory burst length reached

23.6.5 Burst Clock Divisor (BCD)

In some cases, it may be necessary to slow the external bus in relation to the internal bus to allow accesses to burst devices that have a maximum operating frequency less than the operating frequency of the internal bus.

The internal bus frequency can be divided by one, two, three or four for presentation on the external bus in burst mode operation.

To achieve frequency of BCLK other then an integer division, the cycle time of the EIM clock entering the controller must be changed accordingly.

By programming the BCD bit field to various values, two signals on the external bus are affected; $\overline{ADV}$ and BCLK. The $\overline{ADV}$ signal is asserted according to RADVA or WADVA bit fields programming, and is negated according to the formula mentioned in RADVN and WADVN bit fields description. The BCLK signal runs with a 50% duty cycle until the end of access is reached.

If BCM = 1, the BCLK runs at frequency according to GBCD bit field settings on every async memory access, regardless of the SWR and SRD bits configuration. Caution should be exercised when using BCM bit; GBCD bit field should be updated once and should not change when BCLK is toggling. The BCM bit is used mainly for system debug mode. It has no functional use of the EIM in normal mode.

23.6.6 Burst Clock Start (BCS)

In an effort to allow greater flexibility in achieving the minimum number of wait states on burst accesses, you can determine when you want the BCLK to start toggling after the start of access. This allows the BCLK to be skewed from point of data capture on the EIM clock by any number of EIM clock cycles.

Care must be exercised when setting BCS bit field in conjunction with the BCD and RWSC/WWSC bit fields. See the external timing diagrams in "Burst Memory Accesses Timing Diagrams" for examples of how to use the BCS, BCD and RWSC/WWSC bit fields together.

23.6.7 Multiplexed Address/Data Mode Support

The control bit MUM allows support memory with multiplexed address/data bus both in asynchronous and in synchronous modes.

Caution should be exercised for using OEA/WEA & ADH bit fields. They should be configured according to the external device requirements, as it determines the time point of end of address phase and start of data phase.

23.6.8 Mixed Master/Memory Burst Modes Support

To provide mixed sequential/wrap accesses with different length, EIM interprets burst signal and generate additional $\overline{ADV}$ signals whenever there appear unequal address or burst boundary crossing condition.

BL bit field is used to notify EIM about current memory burst and wrap condition for properly external address generation. In case of non-matching boundaries in both the memory and Master access, EIM starts a new memory burst access by updating address from Master on address bus and generating $\overline{\text{ADV}}$ signal.

23.6.9 AXI (Master) Bus Cycles Support

The EIM uses an ARM AXI slave interface. It has a 32-bit bus and supports one access (one ID) at a time. No out of order or parallel accesses are supported.

The following AXI protocol signals are not supported:

- AWLOCK
- AWCACHE
- ARLOCK
- ARCACHE

ARID bus is sampled when:

- new read access is valid on the read address channel and is reflected on the RID bus output toward the master.

AWID bus is sampled when:

- new write access is valid on the write address channel and is reflected on the WID/ BID bus output toward the master.

ARPROT and AWPROT signal are partially used. ARPROT[0] and AWPROT[0] bits are used for normal/privileged access detection. ARPROT[2:1] and AWPROT[2:1] are not used.

When sampling a valid access on both of the address channels, the read access will be performed first while write access is pending. After last data transfer completed, the pending write will be executed.

A new access may be executed one cycle after sampling a valid access on the read or write address channels. Assuming there is no current access (back to back) which can cause a recovery or end of access penalty cycles; for write access, data should be present at write buffer for fast execution.

NOTE

- Only 32-bit word size accesses are supported for burst mode accesses.
- Only 8-bit (1 byte), 16-bit (2 byte) and 32-bit (4 byte) word size supported for single access.

- Maximum number of burst length is 16.
- According to AXI protocol, burst access should not cross 4 KB blocks. In case EIM gets an access that crosses the 4 KB, memory address calculation is invalid.

AXI transfers shown in the table below are also supported. These AXI cycles will be translated into the necessary cycles on the memory side. For example, for optimal operation in case ARM cache is configured to 8 beat burst with wrap, a synchronous flash and cellular RAM memory should be configured in 16 word wrap burst mode when using a 16-bit data port, and in 8 word wrap burst mode when using a 32-bit data port. EIM uses BL bit field to support different memory configurations. The controller splits the transaction when needed in some cases. See Table 23-9.

Table 23-8. AXI Burst Cycles Supported

Burst Length - Number of data transfers	Burst size - Bytes in transfer	Burst type	Description
1	1	INCR	Single transfer
1	2	INCR	Single transfer
1	4	INCR	Single transfer
2	4	WRAP	2-beat wrapping burst
4	4	WRAP	4-beat wrapping burst
8	4	WRAP	8-beat wrapping burst
16	4	WRAP	16-beat wrapping burst
2	4	INCR	2-beat incrementing burst
3	4	INCR	3-beat incrementing burst
4	4	INCR	4-beat incrementing burst
5	4	INCR	5-beat incrementing burst
6	4	INCR	6-beat incrementing burst
7	4	INCR	7-beat incrementing burst
8	4	INCR	8-beat incrementing burst
9	4	INCR	9-beat incrementing burst
10	4	INCR	10-beat incrementing burst
11	4	INCR	11-beat incrementing burst
12	4	INCR	12-beat incrementing burst
13	4	INCR	13-beat incrementing burst
14	4	INCR	14-beat incrementing burst
15	4	INCR	15-beat incrementing burst
16	4	INCR	16-beat incrementing burst

Table 23-9. AXI to Memory Burst Splits Number

AXI Burst Type	Memory Burst Type Config.	# of accesses to X8 Memory Port size	# of accesses to X16 Memory Port size	# of accesses to X32 Memory Port size
INC16 Aligned Addr.	WRAP4	16	8	4
	Cont.	1	1	1
INC16 Unaligned Addr.	WRAP4	17	9	5
	Cont.	1	1	1
WRAP16 Aligned Addr.	WRAP16	4	2	1
	Cont.	1	1	1
WRAP16 Unaligned Addr.	WRAP16	5	3	1
	Cont.	2	2	2
INC8 Aligned Addr.	WRAP8	4	2	1
	WRAP16	2	1	1
INC8 Unaligned Addr.	WRAP8	4 or 5	2 or 3	2
	WRAP16	2 or 3	2	1 or 2
WRAP8 Aligned Addr.	WRAP16	2	1	1
	Cont.	1	1	1
WRAP8 Unaligned Addr.	WRAP16	2 or 3	1	2
	Cont.	2	2	2

23.6.10 WAIT_B Signal, RWSC and WWSC bit fields Usage

Most of the external devices supporting burst mode for write or read accesses providing signal which indicates for data valid on the memory bus (a.k.a. handshake mode). For this mode, RFL and WFL bits should be cleared and RWSC/ WWSC bit fields indicate when the controller should start sampling this signal from the external device or, in other words, how many BCLK cycles should be masked.

For devices which are not using this signal or have a fixed latency ability, the RFL and WFL bits may be set for internal calculation regarding BCLK cycles penalty until data is valid (memory initial access time). For this mode, RWSC/ WWSC indicates when the data is ready for sampling by the controller (read access) or the external device (write access). There is separation between read and write accesses wait-state control. For read access, RWSC bit field is valid and WWSC bit field is ignored; for write access, WWSC is valid and RWSC is ignored.

23.6.11 IPS Register Interface

Access to the registers of the EIM, read or write, is made with IPS protocol signals. The system should avoid changing the registers while master/memory transaction is valid, as this can cause an unknown behavior of the controller.

Register access size is 32-bit as the register size definition, other size of access (byte or half word) is not supported.

23.6.12 MRS Set for PSRAM

Memory registers of PSRAM devices can be configured according to external signal, which indicates whether the access is to a memory array or memory register domain.

When the CRE bit is set, the following transactions to the external device will assert the CRE signal. The polarity of this signal is determined by the CREP bit for active low or active high assertion of the signal.

23.6.13 EIM Access Termination

EIM is monitoring the corresponding CSx control signal every time variable latency access or dtack access is performed toward the external device.

In variable latency accesses, the Watchdog Timer (WDOG-1) counts BCLK cycles. If it reaches the wdog_limit (according to the WDOG_LIMIT bit field in the WCR) before the device signals can drive/sample new data, the controller will terminate the access and generate an error response transfer toward the Master.

In dtack access, WDOG-1 counts ACLK cycles instead of BCLK and it reaches the wdog_limit before the device asserts the dtack signal, the controller will terminate the access and generate an error response transfer toward the Master.

WDOG-1 can by disabled by WDOG_EN bit in the WCR.

23.6.14 Error Conditions

The following conditions cause an error (AXI error or IPS error) response signal:

- AXI errors
  - Access to a disabled chip select - access to a mapped chip select address space where the CSEN bit in the corresponding chip select Configuration Register is clear

- Access to a non mapped address - access to an address that is not mapped to any CS.
- User access to a supervisor-protected chip select address space (the SP bit in the corresponding chip select Configuration Register is set)
- User access in fixed mode access
- User preform write access to write protected chip select
- First write data ID and write address ID do not match. (No data is written to the memory.)
- First Write Data ID and write address ID match but one or more of the other Write data IDs does not match the First Write data ID (data is written to memory according)
- Access duration to external device from CSx signal assertion is 128/256/512/1024 cycles (access is terminated by the controller) - This error can be disabled be software.

- IPS errors
  - User read or write access to a reserved/non-valid address in the EIM Configuration Register

23.6.15 DTACK Mode

In DTACK mode, the EIM uses DTACK signal as an indication of when to end the access.

DTACK is an asynchronous edge/level sensitive signal. DTACK polarity is configurable by the DAP bit in CsxGCR2 (default value is 0).

In this case, EIM begins the access and after a few cycles (according DAPS field) and waits until DTACK (after synchronization) becomes asserted, then samples the data in read access and completes the current data access (see Figure 23-13, Figure 23-14 & Figure 23-15).

If more than one data is needed, CS will be negated between access (csrec field is not zero) and the AXI burst access will be split into single accesses (see Figure 23-17).

23.6.16 RDY_INT Signal as Interrupt

The EIM has an external interrupt support. When INTEN bit in the WCR is set, signal RDY_INT is used as interrupt; its status is being reflected by INT bit and output signal.

It is cleared by writing one to the INT bit. When INTEN is cleared, the interrupt is disabled. This interrupt is a level interrupt and its polarity can be configured by the INTPOL bit in the WCR.

23.6.17 RDY_INT Signal as Ready After Reset Indication

This feature is used for boot propose from external devices based on NANDFlash array memory with NORFlash interface.

When ERRST bit is set, RDY_INT signal is monitored to determine ready after reset of the external device located on CS0.

The monitoring is taking place when CS0 is accessed for the first time. The access will be pending until assertion of the signal is detected. When detection occurs, ERRST bit is self-cleared and pending access is executed to the external device on CS0.

23.6.18 EIM_GRANT / EIM_BUSY Handshake Description

Prior to executing command to one of the external device (chip select), EIM assert EIM_BUSY signal (1'b1) and checks the EIM_GRANT signal status.

If EIM_GRANT signal is high, it indicates external data bus is not used by other slaves (NAND Flash Controller) and EIM may start to execute the access. If EIM_GRANT is low, EIM waits until it is set (1'b1) before executing the access.

EIM keeps EIM_BUSY signal set until it completes the access toward the external device.

Once EIM_GRANT signal is set, it can not be reset until EIM_BUSY signal is cleared by EIM.

NOTE

In 16-bit Muxed EIM doesn't use the data bus, therefore there is no sharing of the data bus with NFC. EIM doesn't wait for EIM_GRANT signal from NFC and doesn't assert the EIM_BUSY signal.

23.6.19 LPMD / LPACK Handshake Description

These signals are used for frequency and/or voltage change, and for entering low power mode during normal operation of the EIM. Before any change can take place, the controller and all the relevant external devices should be in idle state, which means no access or data transfer is in process.

LPMD input signal is asserted once EIM detects the assertion of LPMD, all ready signals of the AXI channels are negated, and EIM is not sampling new accesses. It finishes all the ongoing accesses and already pending ones. When EIM is in idle state, the LPACK output signal is asserted. EIM will stay in idle state and the LPACK signal will stay asserted until the LPMD signal is negated.

23.6.20 Endianness

Big and Little endianness are supported by the controller according to the following table.

Table 23-10. EIM Out/in Data in Case AXI Out/in Data is 0xB3B2B1B0

Endian mode	AXI access	AXI address [1:0]	Port size and used bits								
			Word port				Half word port			Byte port	
			[31:24]	[23:16]	[15:8]	[7:0]	External address [0]	[31:24] ([15:8])	[23:16] ([7:0])	External address [1:0]	[31:24] ([23:16]) ([15:8]) ([7:0])
Big	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB3	0xB2	0	0xB3
										1	0xB2
							1	0xB1	0xB0	2	0xB1
										3	0xB0
	Half Word	0			0xB1	0xB0	0	0xB3	0xB2	0	0xB3
										1	0xB2
		2	0xB3	0xB2			1	0xB1	0xB0	2	0xB1
										3	0xB0
	Byte	0				0xB0	0		0xB3	0	0xB3
		1			0xB1			0xB2		1	0xB2
		2		0xB2			1		0xB1	2	0xB1
		3	0xB3					0xB0		3	0xB0

Table continues on the next page...

Table 23-10. EIM Out/in Data in Case AXI Out/in Data is 0xB3B2B1B0 (continued)

Endian mode	AXI access	AXI address [1:0]	Port size and used bits								
			Word port				Half word port			Byte port	
			[31:24]	[23:16]	[15:8]	[7:0]	External address [0]	[31:24] ([15:8])	[23:16] ([7:0])	External address [1:0]	[31:24] ([23:16]) ([15:8]) ([7:0])
Little	Word	0	0xB3	0xB2	0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
							1	0xB3	0xB2	2	0xB2
										3	0xB3
	Half Word	0			0xB1	0xB0	0	0xB1	0xB0	0	0xB0
										1	0xB1
		2	0xB3	0xB2			1	0xB3	0xB2	2	0xB2
										3	0xB3
	Byte	0				0xB0	0		0xB0	0	0xB0
		1			0xB1			0xB1		1	0xB1
		2		0xB2			1		0xB2	2	0xB2
		3	0xB3					0xB3		3	0xB3

23.6.21 Strobe Signal Use

The strobe signal is toggling according to address/data valid condition on the external bus for read and write accesses, and for both synchronous and asynchronous modes.

At any time point when address/data is valid on the external bus, the strobe signal will generate a positive edge, which can be used to sample the external data and control signal.

NOTE

Strobe signal for read data is active (RL + 1) cycles after data on external bus is valid.

23.7 Initialization Information

23.7.1 Active Chip Selects and Address Space Configuration

Each chip select activation is set using VIA port configuration.

Refer to the SoC chapter for the specific EIM chip select configuration.

Chip select is determine when the corresponding bit of the ACT_CS[5:0] input is set to one, else it is in none active state.

Each one of the four chip selects address space can be determined by ADDRS0[1:0] through ADDRS5[1:0] EIM inputs. The Minimum address space size per chip select is 32MByte, maximum size is 256MByte. Overall address space for all chip select is 512MByte. Base address is {EIM_NFC_BASE,28'h0}, topmost base address is CS0 base address + 0x1FFFFFFF.

Address space order is determined from the largest space on CS0 to the smallest one on CS3. Not all the chip selects must be used (determined by the active bit per CS inputs). Equal size for several chip selects may apply. It is the user's responsibility to ensure the total space size configured is within the 512 MByte dedicated to the EIM.

NOTE

If the total address space is smaller then 512MByte, it is referred as reserved space and can not be used.

Table 23-11. Examples of Address Space Configuration

chip select #	Exmple1	Exmple2	Exmple3	Exmple4
CS0	256 MByte	128 MByte	128 MByte	256 MByte
CS1	128 MByte	128 MByte	64 MByte	64 MByte
CS2	64 MByte	128 MByte	64 MByte	64 MByte
CS3	64 Mbyte	64 Mbyte	32 Mbyte	32 Mbyte

The address space is set with the matching ADDRSx[1:0] input bus according to the table below.

Table 23-12. Address Space Configuration

ADDRSx[1:0]	Size
2'b00	32 MByte
2'b01	64 MByte
2'b10	128 MByte
2'b11	256 Mbyte

23.7.2 Booting from EIM

EIM is ready to work with CS0 after the hardware reset, but it has been configured for very slow access (for boot purposes), with additional setup and hold time.

Other CSs are disabled by hardware reset. Therefore, all CSs must be properly initialized before use in writing values to the corresponding chip select configuration registers.

DSZ[1:0] and MUM fields are set according to EIM_BOOT [2:0] block inputs.

23.8 Application Note

Application note uses next functions to illustrate EIM and memory accesses:

- WR16(address, data) is a 16 bit write access
- WR32(address, data) is a 32 bit write access
- RD16(address, data) is a 16 bit read access
- RD32(address, data) is a 32 bit read access
- WR_I(address, data, delta, counter) is a write data sequence, there data(i+1) = data(i) + delta
- COMMAND_SEQUENCE as described on page 1-53
- CHECK_STATUS as described on page 1-53

All addresses are byte address. 'CS0 is a Chip Select 0 base address. 'EIM_ is a prefix of EIM's registers. 'h is a prefix of hexadecimal constant. // is a comment beginning. csba[cs] is a dimension of CS base addresses. "addr" means an address offset in current CS address space. Examples use CS0 address space, but it may be any except boot mode functionality.

Configuration examples were checked with some below defined memory models and may require some adjustment for other family members.

23.8.1 Access to AMD Flash

Below mentioned configurations intended for AMD Simultaneous Flash Memory w/ VersatileIO^{TM} Control.

23.8.1.1 AMD Flash Asynchronous Mode Configuration

Next EIM register configuration used for AMD flash asynchronous access:

- WR32('EIM_CS0RCR1,'h0a018000);

- WR32('EIM_CS0WCR1,'h0704a240);
- WR32('EIM_CS0GCR1,'h00430081); // for 32 bit memory or
- WR32('EIM_CS0GCR1,'h00410081); // for 16 bit memory

23.8.1.2 AMD Flash Utility

```
       // Single data word programming to addr, 32-bit memory
           port_size = 32;
           WR32('CS0+('h0555<<2),'h00aa); // command sequence
           WR32('CS0+('h02aa<<2),'h0055);
           WR32('CS0+('h0555<<2),'ha0); // command code
           WR32('CS0+addr, data); // addr & data
// Polling end of programming
          edata = data; // expected data
          data = ~edata;
          errst = 0;
      while(!(data == edata) &&!errst)
             begin: BR_EN
             RD16('CS0+addr, data);          // Read status
             if(data[7]!== edata[7])
                begin
                if(data[5] == 1)
                    begin
                    RD16('CS0+addr, data3);
                    RD16('CS0+addr, data);
                    if(data[6] !== data3[6])
                        begin
                        $display("CHECK_STATUS: Error timout on single data program");
                        errst = 1;
                        disable BR_EN;
                        end
                    end
               end
             else
                begin
                RD16('CS0+addr, data3);
                if(port_size == 32)
                    RD32('CS0+addr, data);
                else
                    begin
                    RD16('CS0+addr, data);
                    edata[31:16] = 16'h0;
                    end
                if(data !== edata)
                    begin
                    $display("CHECK_STATUS: Error in data write on single data program");
                      errst =2;
                    disable BR_EN;
                    end
                end
             end
      // Single data word programming to addr, 16-bit memory
          port_size = 16;
          WR32('CS0+('h0555<<1),'h00aa); // command sequence
          WR32('CS0+('h02aa<<1),'h0055);
          WR32('CS0+('h0555<<1),'ha0); // commund code
          WR32('CS0+addr, data); // addr & data
      // Polling end of programming - the same as for 32 bit
```

23.8.2 Access to Intel Sibley Flash

Below mentioned configurations intended to Sibley family muxed and non-muxed devices.

23.8.2.1 Intel Sibley Flash Asynchronous Mode Configuration

- WR32('EIM_CS0GCR1,'h00210081);
- WR32('EIM_CS0RCR1,'h0e020000);
- WR32('EIM_CS0RCR2,'h00000000);
- WR32('EIM_CS0WCR1,'h0704a040);

23.8.2.2 Intel Sibley Flash Synchronous Mode Configuration

Next configuration used for 133 MHz synchronous access to flash:

```
// Set memory to synchronous read mode
     WR16('CS0+('h5903<<1),'h0060);
     WR16('CS0+('h5903<<1),'h0003);
     WR16('CS0+('h0000<<1),'h00ff);
// Set EIM configuration to synchronous timing
     WR32('EIM_CS0GCR1,'h50214225);                    // 133 MHz
     WR32('EIM_CS0RCR1,'h0c000000);                    // 12 cycles on memory
```

Next configuration used for 66 MHz synchronous access to muxed flash:

```
// Set memory to synchronous read mode
     WR16('CS0+('h3103<<1),'h0060);
     WR16('CS0+('h3103<<1),'h0003);
     WR16('CS0+('h0000<<1),'h00ff);
//-------------------------------------------------------------------------
// Set EIM configuration to synchronous timing
     WR32('EIM_CS0GCR1,'h5021122d);                    // 66 MHz
     WR32('EIM_CS0RCR1,'h07000000);                    // 7cycles on memory
```

23.8.2.3 Intel Sibley Flash Utility

```
// Single data word programming to addr
     WR16('CS0+addr,'h0060);                 // Unlock
     WR16('CS0+addr,'h00d0);
     WR16('CS0+addr, 'h0041);
     WR16('CS0+addr, data);
     WR16('CS0+caddr, 'h0070);                 // Read Status command
     while('CS0+data[7] == 0)                                  // Wait / Polling
         RD16('CS0+addr, data);              // Read status
     RD16('CS0+addr, data);                  // Read status
     WR16('CS0+'h0000,'h00ff);
// Write buffer programming
     WR16('CS0+addr,'h0060);                 // Unlock
     WR16('CS0+addr,'h00d0);
     data    = 0;
     WR16('CS0+addr, 'h0070);                // Read Status command
     while(data[7] == 0)                           // Wait
         RD16('CS0+addr, data);              // Read status
```

```
WR16('CS0+'h0000,'h00ff);
           WR16('CS0+addr,'h00e9);           // Write Buffer command
           WR16('CS0+addr,255);              // Word counter (<256)
           for(i=0; i<'h200; i = i + 'h40)
              WR_I('CS0+addr+i, data+((i>2)*'h0010_0001), 'h0010_0001, 16); // Data
           WR16('CS0+addr,'h00d0);           // Write Confirm command
           data  = 0;
           while(data[7] == 0)                      // Wait
              RD16('CS0+addr, data);         // Read status
           RD16('CS0+addr, data);            // Read status
           WR16('CS0+'h0000,'h00ff);
```

23.8.3 Access to MDOC Device

Below mentioned configurations intended to MDOC H3 device.

23.8.3.1 MDOC Device Boot

To boot from the MDOC device the ERRST bit should be configured to 1, so that EIM will hold the first read access to CS0 until the MDOC asserts the RDY signal.

23.8.3.2 MDOC Device Asynchronous Mode Configuration

```
// Non-muxed mode
WR32('EIM_CS0GCR1,'h00410081);
WR32('EIM_CS0RCR1,'h0e121010);
WR32('EIM_CS0RCR2,'h00000000);
WR32('EIM_CS0WCR1,'h12092492);
// Muxed mode
WR32('EIM_CS0GCR1,'h00410081);
WR32('EIM_CS0RCR1,'h0e121010);
WR32('EIM_CS0RCR2,'h00000000);
WR32('EIM_CS0WCR1,'h12092492);
```

23.8.3.3 MDOC Device Utility

```
// Read Manufacturer ID and Device ID
    RE16('CS0+'h9400,'h4833);
    RE16('CS0+'h9422,'hb7cc);
```

23.8.4 Access to Micron PSRAM

Below mentioned configurations intended to mt45w4mw16bfb_706.

23.8.4.1 Micron PSRAM Asynchronous Mode Configuration

```
// 16 bit memory
    WR32('EIM_CS0GCR1,'h403104b1);
    WR32('EIM_CS0RCR1,'h0b010000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h0b040040);
// 32 bit memory
    WR32('EIM_CS0GCR1,'h403304b1);
    WR32('EIM_CS0RCR1,'h0f010000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h0f040040);
```

23.8.4.2 Micron PSRAM Synchronous Mode Configuration

```
// 16 bit memory
    WR32('EIM_CS0GCR1,'h403104b1);
    WR32('EIM_CS0WCR1,'h0b040000);
    WR16('CS0+('h85947<<1),'h0040); // memory configuration
    WR32('EIM_CS0GCR1,'h4021_5487);// fixed latency memory wrap 4
    WR32('EIM_CS0RCR1,'h04000000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h04000000);
// 32 bit memory
    WR32('EIM_CS0GCR1,'h6003_04f1);
    WR32('EIM_CS0WCR1,'h0b04_0000);
    WR32('CS0+('h85947<<2),'h0040); // memory configuration
    WR32('EIM_CS0GCR1,'h4003_1487); // var latency memory inc. page size 128
    WR32('EIM_CS0RCR1,'h04000000);
    WR32('EIM_CS0RCR2,'h00000008);
    WR32('EIM_CS0WCR1,'h04000000);
```

23.8.5 Access to Samsung OneNAND

Mentioned below are the configurations intended for Samsung OneNAND muxed and non-muxed devices.

23.8.5.1 Samsung OneNAND Boot

There are two ways to boot from Samsung OneNAND. In the first way, the ERRST bit is set to 0 and the user has to poll the interrupt status in the OneNAND interrupt register (or set interrupt handler there). In the second way, the ERRST bit is set to 1 and the user should enable the device interrupt output before the first read from CS0 access is issued.

Load sectors 2,3 to DataRAM, page 0 done in the next example:

- WR16('CS0+('hF241<<1),'h0); // Clear interrupt status
- WR16('CS0+('hF100<<1),'h0); // block[8:0] address
- WR16('CS0+('hF107<<1),'h2); // sector[1:0] and page[7:2] addresses
- WR16('CS0+('hF200<<1),'h802); // buffer[11:8] address and counter[1:0]

- WR16('CS0+('hF101<<1),'h0); // DDP choose
- WR16('CS0+('hF220<<1),'h0); // Set command

23.8.5.2 Samsung OneNAND Asynchronous Mode Configuration

```
// Non-muxed memory
    WR32('EIM_CS0GCR1,'h00410081);
    WR32('EIM_CS0RCR1,'h0b010000);
    WR32('EIM_CS0RCR2,'h00000000);
    WR32('EIM_CS0WCR1,'h0c092480);
// Muxed memory
    WR32('EIM_CS0GCR1,'h00410089);
    WR32('EIM_CS0RCR1,'h0b010000);
    WR32('EIM_CS0RCR2,'h00000000);
    WR32('EIM_CS0WCR1,'h0c092480);
```

23.8.5.3 Samsung OneNAND Synchronous Mode Configuration

Set memory and EIM to synchronous read mode is shown in the next example:

```
    WR16('CS0+('hF221<<1),'hc0e0); // Synchronous read, 4 clk latency
    WR32('EIM_CS0GCR1,'h50412405); // 44 MHz (non-muxed)
    WR32('EIM_CS0RCR1,'h05010000);
```

The muxed Samsung OneNAND supports synchronous write, too:

```
// Set memory & EIM to synchronous read and write mode
    WR16('CS0+('hF221<<1),'hc0f2); // Sync. read and write, 4 clk latency
    WR32('EIM_CS0GCR1,'h5041240f); // 44 MHz
    WR32('EIM_CS0RCR1,'h05010000);
    WR32('EIM_CS0WCR1,'h05040000);
```

23.8.5.4 Samsung OneNAND Utility

The following utility algorithms are used on the Samsung OneNAND:

```
// Unlock Block command
    WR16('CS0+('hF100<<1),'h0);        // DFS
    WR16('CS0+('hF100<<1),'h0);        // DBS
    WR16('CS0+('hF24c<<1),'h2);        // SBA - block number (2)
    WR16('CS0+('hF241<<1),'h0);        // Clear interrupt status
    WR16('CS0+('hF220<<1),'h23);       // Unlock command
    data ='h0;
    while(!(data &'h0004))             // Polling
       RD32('WIAR, data); // Read status
// Erase block command
    WR16('CS0+('hF100<<1),'h2);        // DFS and block ([8:0]) address
    WR16('CS0+('hF101<<1),'h0);        // DBS
    WR16('CS0+('hF241<<1),'h0);        // Clear interrupt status
              WR16('CS0+('hF220<<1),'h94);            // Erase command
    data ='h0;
    while(!(data &'h0004))                  // Wait
        RD32('WIAR, data); // Read status
// Program page command
    WR16('CS0+('hF100<<1),'h2);             // DFS and block[8:0] address
    WR16('CS0+('hF107<<1),'h0);             // sector[1:0] and page[7:2] addresses
```

```
          WR16('CS0+('hF200<<1),'h800);           // buffer[11:8] address and counter[1:0]
          WR16('CS0+('hF241<<1),'h0);             // Clear interrupt status
          WR16('CS0+('hF220<<1),'h80);            // Program command
          data ='h0;
          while(!(data &'h0004))                  // Wait
              RD32('WIAR, data); // Read status
```

23.8.6 Access to Samsung UtRAM

Below mentioned configurations are intended for Samsung UtRAM.

23.8.6.1 Samsung UtRAM Asynchronous Mode Configuration

```
          WR32('EIM_CS0GCR1,'h400104b1);
          WR32('EIM_CS0RCR1,'h0a010000);
          WR32('EIM_CS0RCR2,'h00000008);
          WR32('EIM_CS0WCR1,'h0b040040);
```

23.8.6.2 Samsung UtRAM Synchronous Mode Configuration

```
          RD16('CS0+('hff_ffff<<1),data);             // command sequence
          RD16('CS0+('hff_ffff<<1),data);
          RD16('CS0+('hff_ffff<<1),data);
          RD16('CS0+('hff_feff<<1),data);
          RD16('CS0+('h00_82a0<<1),data);             // memory sync. configuration
          WR32('EIM_CS0GCR1,'h4021_53b7);            // fixed latency memory wrap 32
          WR32('EIM_CS0RCR1,'h0500_0000);
          WR32('EIM_CS0WCR1,'h0300_0000);
```

23.8.7 Access to Spansion Flash

Below mentioned configurations are intended for Spansion Flash.

23.8.7.1 Spansion Flash Asynchronous Mode Configuration

```
          WR32('EIM_CS0GCR1,'h00410081);
          WR32('EIM_CS0RCR1,'h0a018000);
          WR32('EIM_CS0RCR2,'h00000000);
          WR32('EIM_CS0WCR1,'h0704a240);
WR16('CS0+('hF220<<1),'h94);            // Erase command
          data ='h0;
          while(!(data &'h0004))                  // Wait
              RD32('WIAR, data); // Read status
     // Program page command
          WR16('CS0+('hF100<<1),'h2);             // DFS and block[8:0] address
          WR16('CS0+('hF107<<1),'h0);             // sector[1:0] and page[7:2] addresses
          WR16('CS0+('hF200<<1),'h800);           // buffer[11:8] address and counter[1:0]
          WR16('CS0+('hF241<<1),'h0);             // Clear interrupt status
          WR16('CS0+('hF220<<1),'h80);            // Program command
          data ='h0;
```

```
            while(!(data &'h0004))                  // Wait
                RD32('WIAR, data); // Read status
```

23.8.7.2 Spansion Flash Synchronous Mode Configuration

```
            WR16('CS0+('h0555<<1),'h00aa);   // command sequence
            WR16('CS0+('h02aa<<1),'h0055);
            WR16('CS0+('h0555<<1),'hd0);
            WR16('CS0+('h0000<<1),'h1ec4);   // memory sync. configuration
            WR32('EIM_CS0GCR1,'h50411325); // 66 MHz
            WR32('EIM_CS0RCR1,'h05000000); // 5 cycles on memory
```

23.8.7.3 Spansion Flash Utility

```
        // Single word programming
            COMMAND_SEQUENCE(cs,16,'ha0);           // single word programming
            WR16('CS0+addr, data);
            CHECK_STATUS('CS0+addr,data,16,1,errst);
        // Write buffer programming
            COMMAND_SEQUENCE(0,16,'h25);              // write buffer programming
            WR16('CS0+addr,'h001f);                     // counter-1
            WR_I('CS0+addr, data, 'h0010_0001, 16); // data
            WR16('CS0+addr,'h0029);                     // write buffer to flash
            CHECK_STATUS('CS0+addr+'h3e,data[31:16]+'h00f0,16,1,errst);
```

There COMMAND_SEQUENCE and CHECK_STATUS are next functions:

```
        task COMMAND_SEQUENCE;
            input[2:0]    cs;
            input[7:0]    port_size;
            input[31:0]   code;
        begin
            if(port_size == 16)
               begin
               WR16(csba[cs]+('h0555<<1),'h00aa);
               WR16(csba[cs]+('h02aa<<1),'h0055);
               WR16(csba[cs]+('h0555<<1),code);
               end
            else
               begin
               WR32(csba[cs]+('h0555<<2),'h00aa);
               WR32(csba[cs]+('h02aa<<2),'h0055);
               WR32(csba[cs]+('h0555<<2),code);
               end
        end
        endtask
        task     CHECK_STATUS;
            input[31:0] addr;
            input[31:0] edata;
            input[7:0]   port_size;
            input[7:0]   opcode;
            output[7:0] errst;
            reg[31:0]    data;
            reg[31:0]    data3;
        begin
            errst = 0;
            data  = 0;
            data3 = 0;
while(!(data == edata) && !errst) // Wait operation
          begin: BR_EN
          RD16(addr, data);             // Read status
          if(data[7] !== edata[7])
```

```
            begin
            if(data[5] == 1)
                 begin
                 RD16(addr, data3);
                 RD16(addr, data);
                 if(data[6] !== data3[6])
                      begin
                      $display("CHECK_STATUS: Error timout on single data program");
                      errst = 1;
                      disable BR_EN;
                      end
                 end
            else
                 begin
                 if(opcode == 2)
                      if(data[1] == 1)
                          begin
                          RD16(addr, data3);
                          if(port_size == 32)
                             RD32(addr, data);
                          else
                             RD16(addr, data);
                          if(data[1] == 1 && data !== edata)
                             begin
                             $display("CHECK_STATUS: Error on write buffer");
                             errst =3;
                             disable BR_EN;
                             end
                          end
                 end
            end
        else
            begin
            RD16(addr, data3);
            if(port_size == 32)
                 RD32(addr, data);
            else
                 begin
                 RD16(addr, data);
                 edata[31:16] = 16'h0;
                 end
            if(data !== edata)
                 begin
                 $display("CHECK_STATUS: Error in data write on single data program");
                 errst =2;
                 disable BR_EN;
                 end
            end
        end
end
endtask
```

23.8.8 8 bit support

This section details the pin connections for Intel mode and Motorola mode.

Intel Mode - For intel mode use the following connection:

Table 23-13. Intel Mode pin connections

ARM platform Pin	EIM Pin	Notes
ADS#	IPP_DO_ADV_B	WAL = 1,RAL = 1

Table continues on the next page...

Table 23-13. Intel Mode pin connections (continued)

ARM platform Pin	EIM Pin	Notes
W/R	IPP_DO_BE_B	WBED = 1
WR#	WE#	
RD#	OE#	

Mot. Mode - For intel mode use the following connection:

Table 23-14. Motorola Mode pin connections

ARM platform Pin	EIM Pin	Notes
AS#	IPP_DO_CS_B	
R/W#	WE#	
LDS#	BE#	

23.9 Booting from OneNAND and NOR Flash devices

External Bus Timing Diagrams

The following timing diagrams show the timing of accesses to memory or a peripheral with different timing parameters. All examples done for CS0, but are valid for any others chip select. BE means one from current used BE[3:0].

23.9.1 Asynchronous Read Memory Accesses Timing Diagram

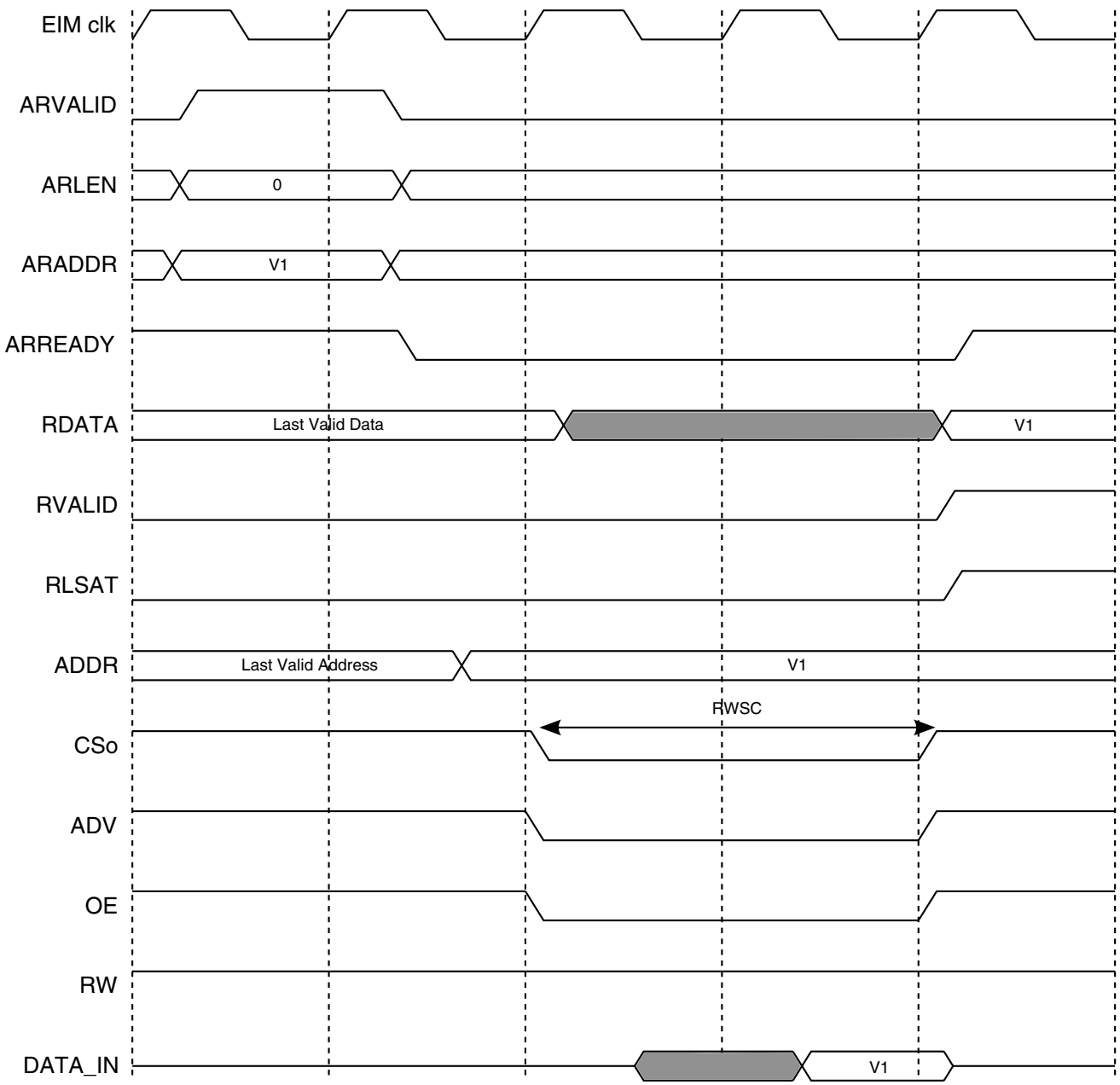


Figure 23-2. Read Access, RWSC=2,RCSA=0,OEA=0,RCSN=0,OEN=0, RAL=1

23.9.2 Asynchronous Write Memory Accesses Timing Diagram

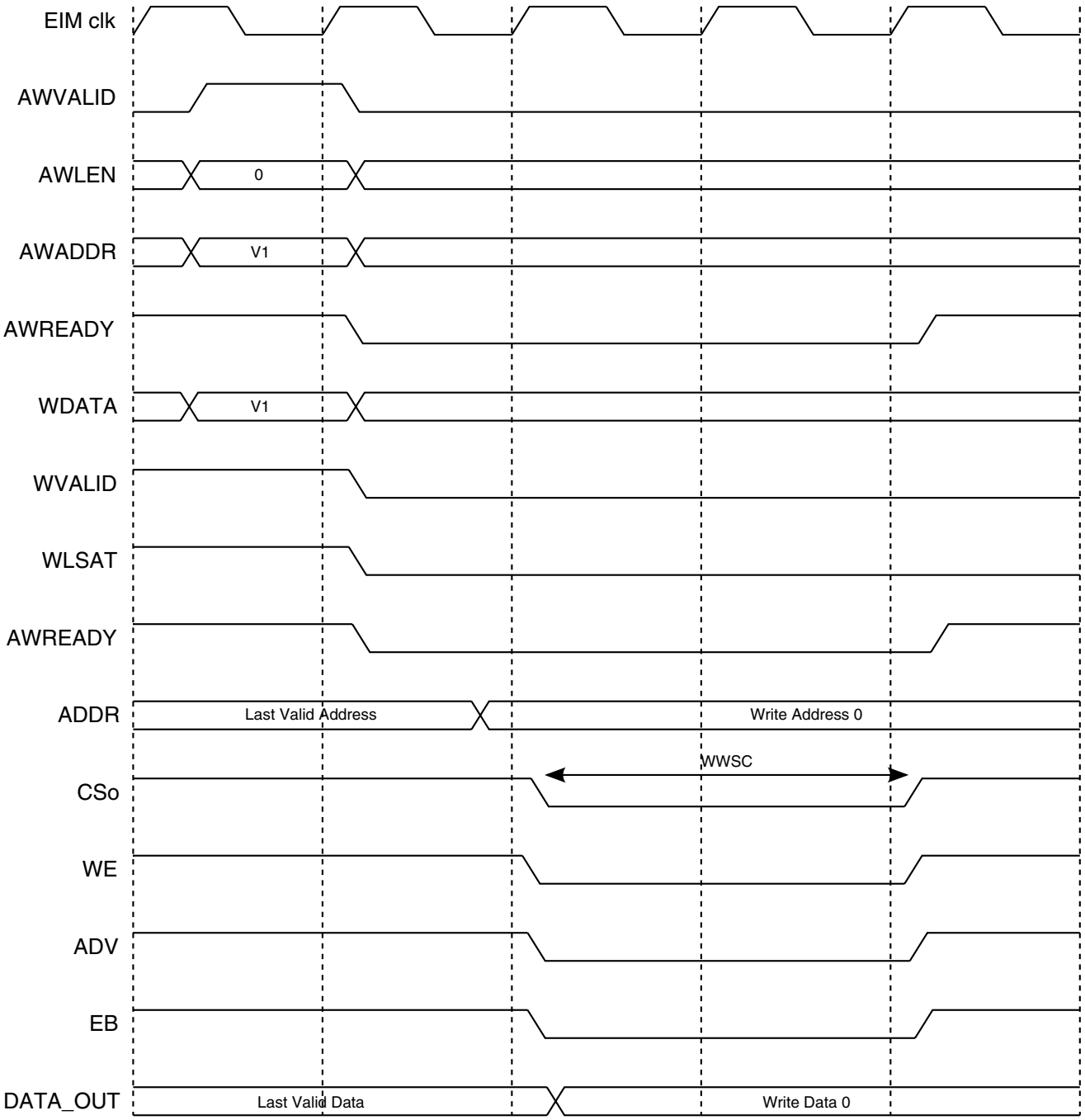


Figure 23-3. Write Access, WWSC=2,WCSA=0,WEA=0,WCSN=0,WEN=0,BEA=0, BEN=0, WAL=1

23.9.3 Asynchronous Read/Write Memory Accesses Timing Diagram

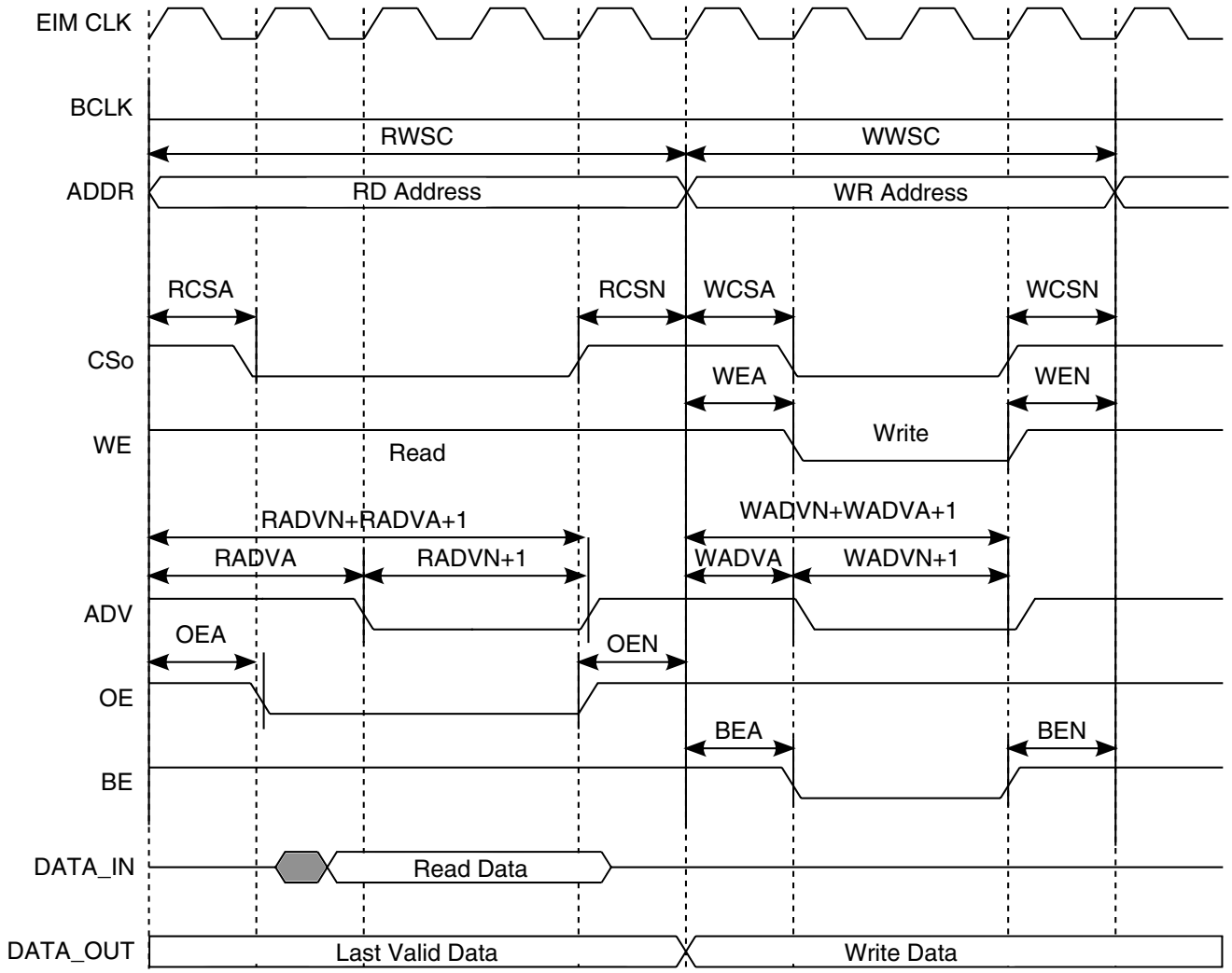


Figure 23-4.
RCSA=1,RADVA=2,OEA=1,RADVN=1,RCSN=1,OEN=1,WCSA=1,WEA=1,WADVA=1,BEA=1,WADVN=1,WCSN=1,WEN=1,BEN=1

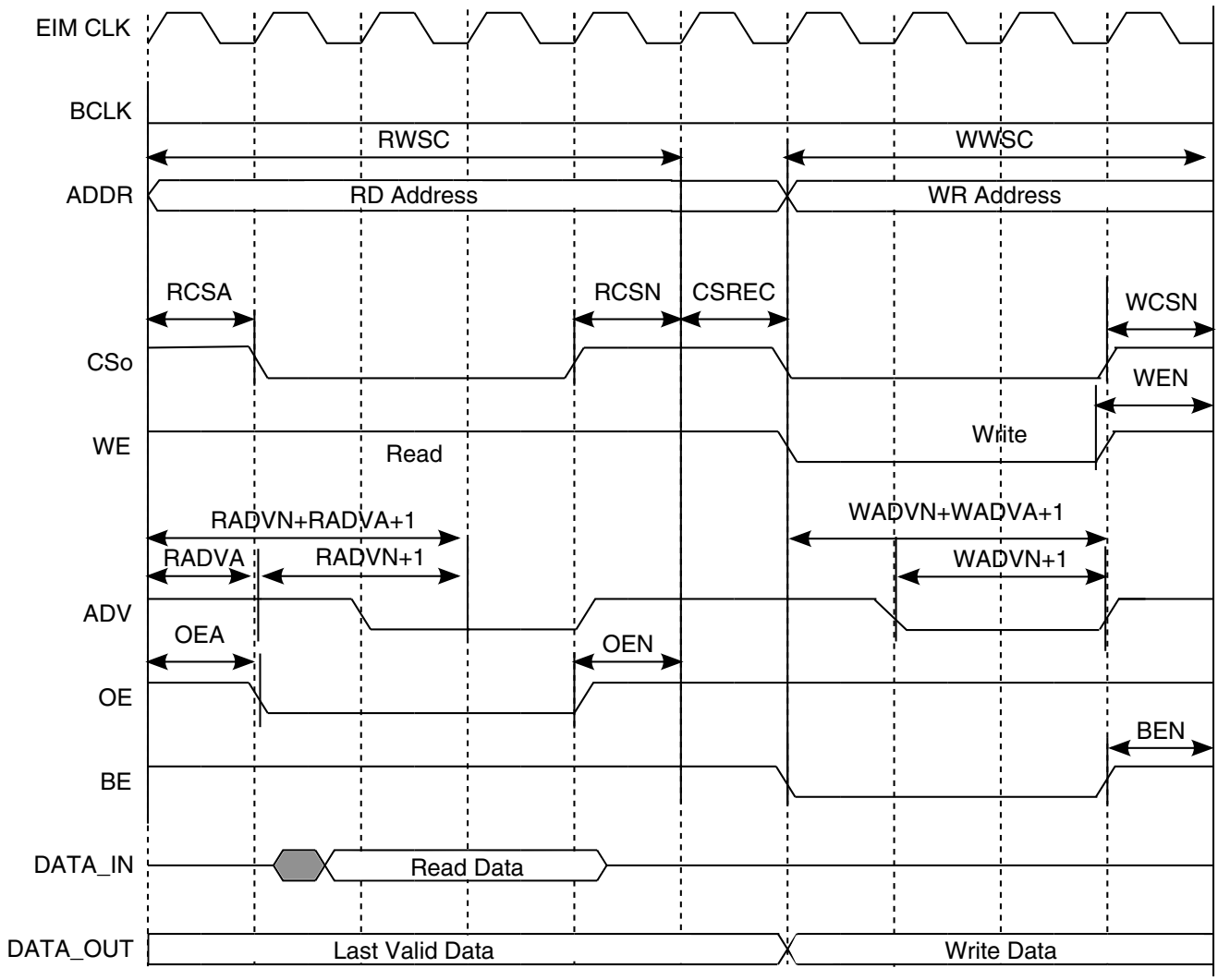


Figure 23-5.
RWSC=5,RCSA=1,RCSN=1,RADVA=1,RADVN=1,OEA=1,OEN=1,WWSC=4,WCSA=0,WCSN=1,WEA=0,WEN=1,WADVA=1,WADVN=1,BEA=0,BEN=1,CSREC=1

23.9.4 Asynchronous Read/Write Using RAL, WAL and CSREC

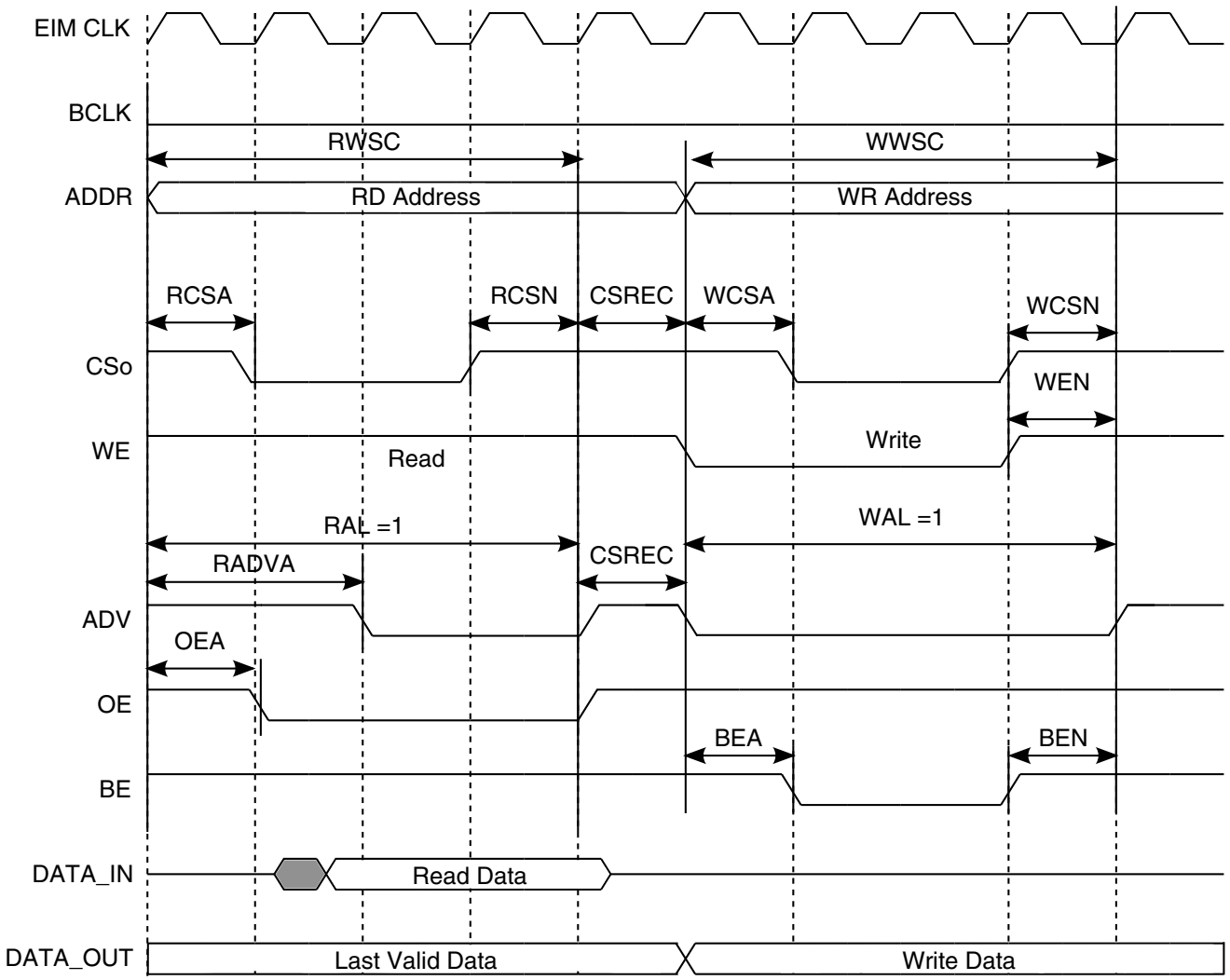


Figure 23-6.
RAL=1,RCSN=1,RADVA=2,OEA=1,RCSN=1,CSREC=1,WCSA=1,WEA=0,WADVA=0,BEA=1,WAL=1,WCSN=1,WEN=1,BEN=1

23.9.5 Consecutive Asynchronous Write Memory Accesses Timing Diagram

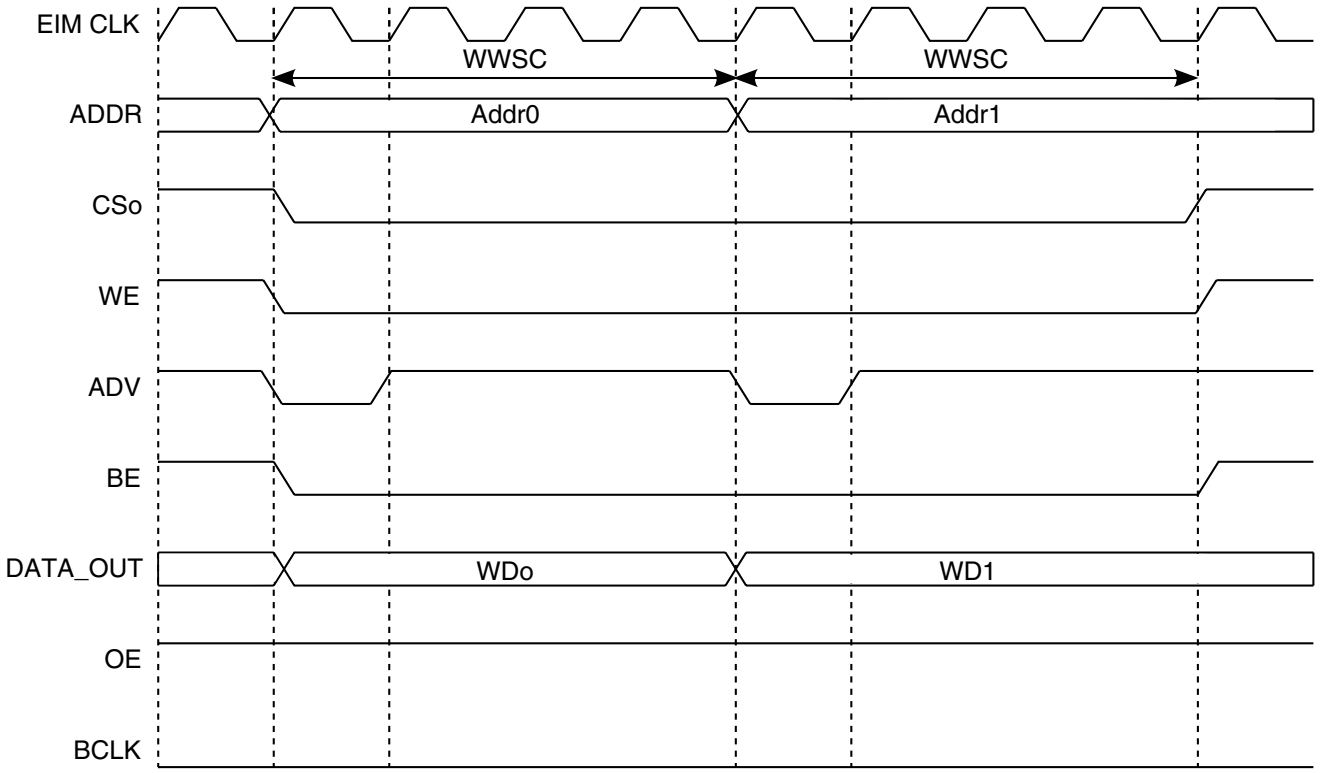


Figure 23-7. WWSC=4,WCSA=0,WEA=0,WADVA=0,BEA=0,WCSN=0,WEN=0,WADVN=0,BEN=0,CSREC=0

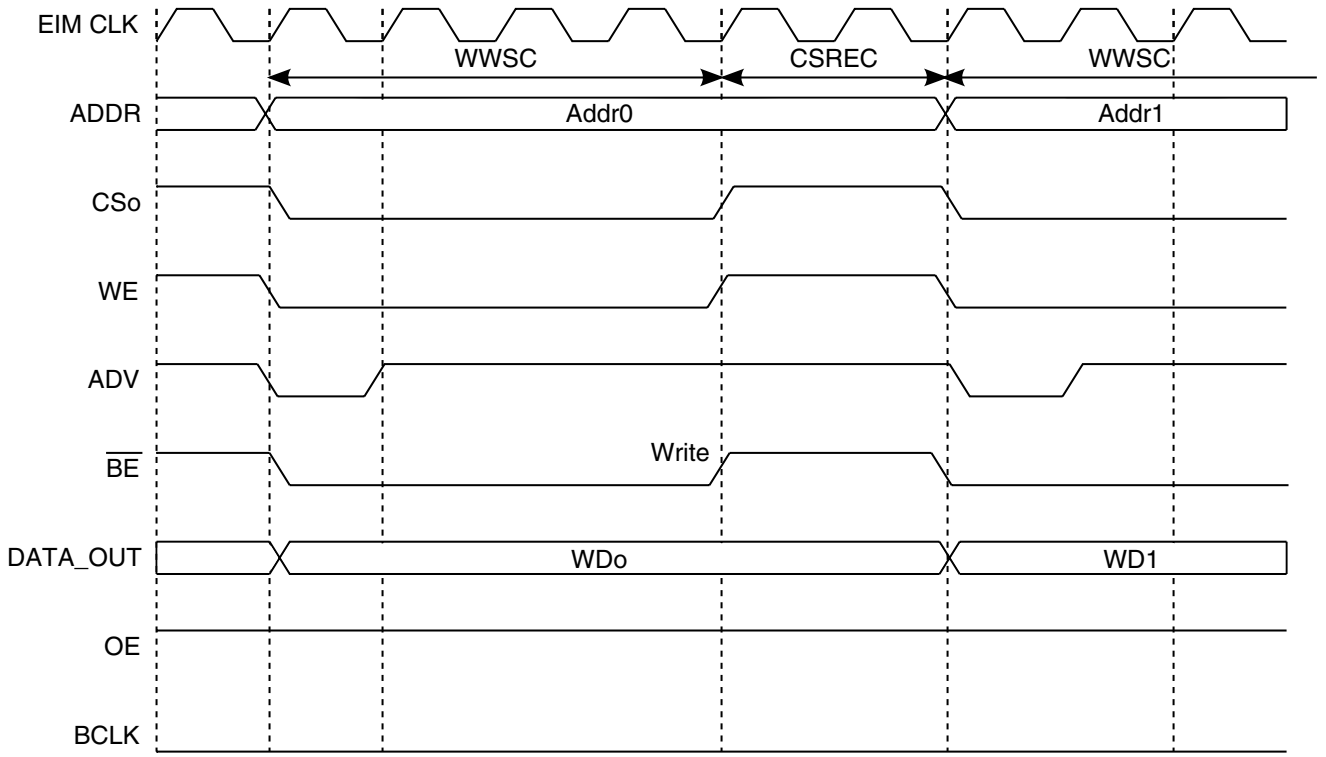


Figure 23-8. WWSC=4,WCSA=0,WEA=0,WADVA=0,BEA=0,WCSN=0,WEN=0,WADVN=0,BEN=0,CSREC=2

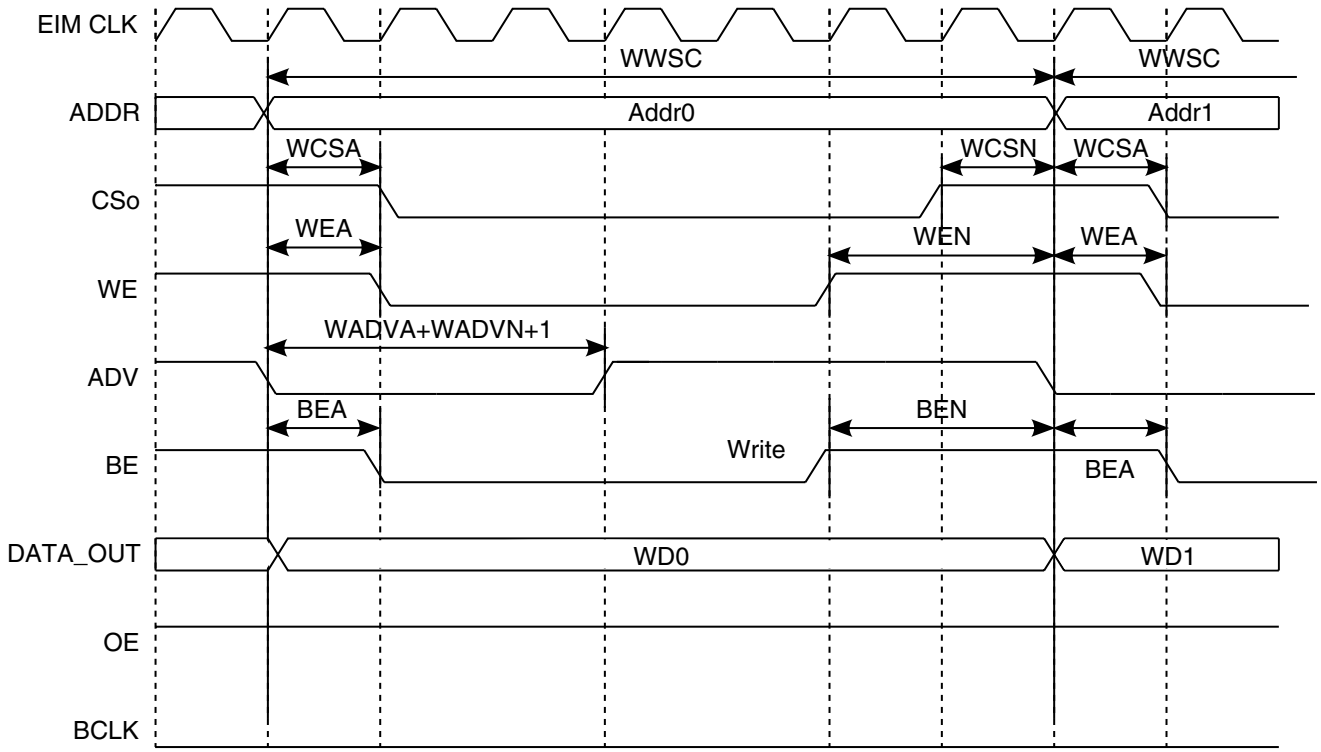


Figure 23-9.
WWSC=7,WCSA=1,WCSN=1,WEA=1,WEN=2,WADVA=0,WADVN=2,BEA=1,BEN=2

23.9.6 Consecutive Asynchronous Read Memory Accesses Timing Diagram

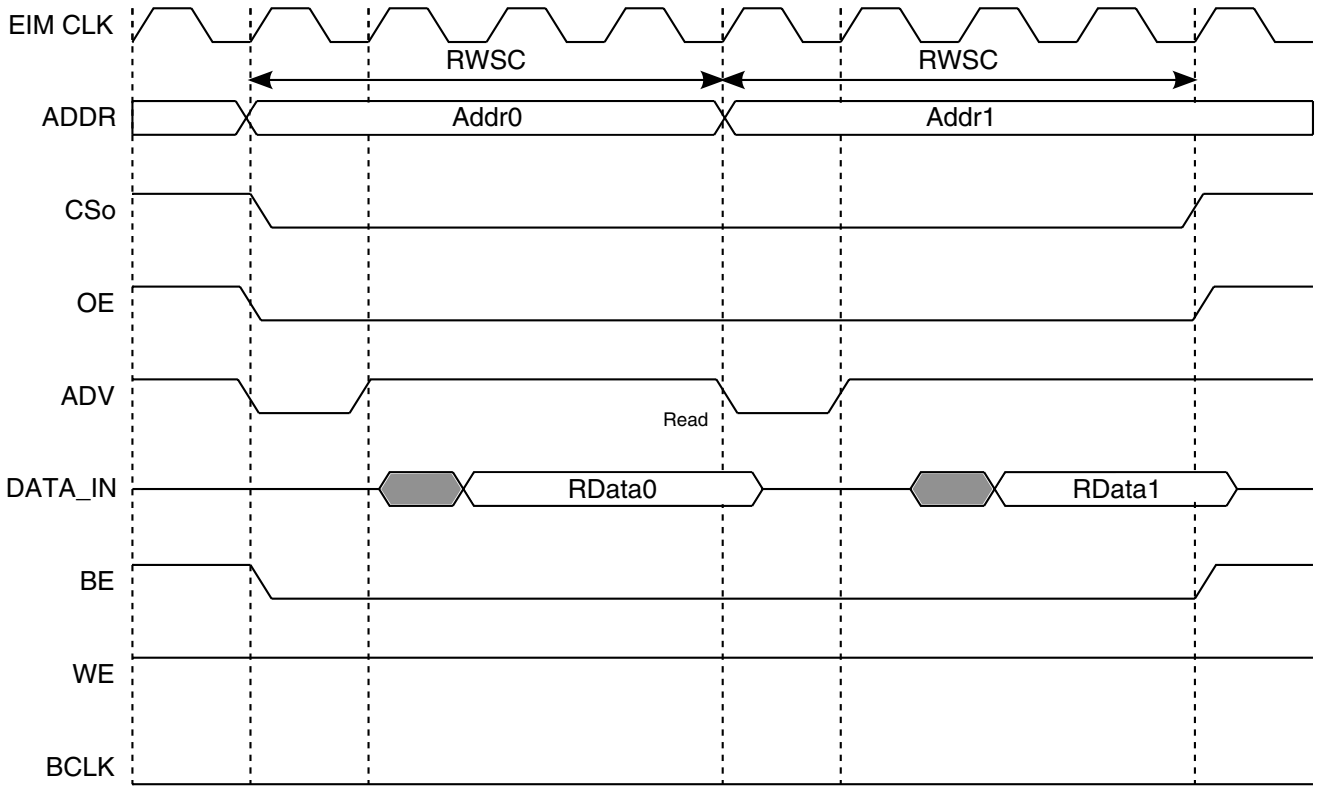


Figure 23-10. RWSC=4,RCSA=0,OEA=0,RADVA=0,RCSN=0,OEN=0,RADVN=0,CSREC=0

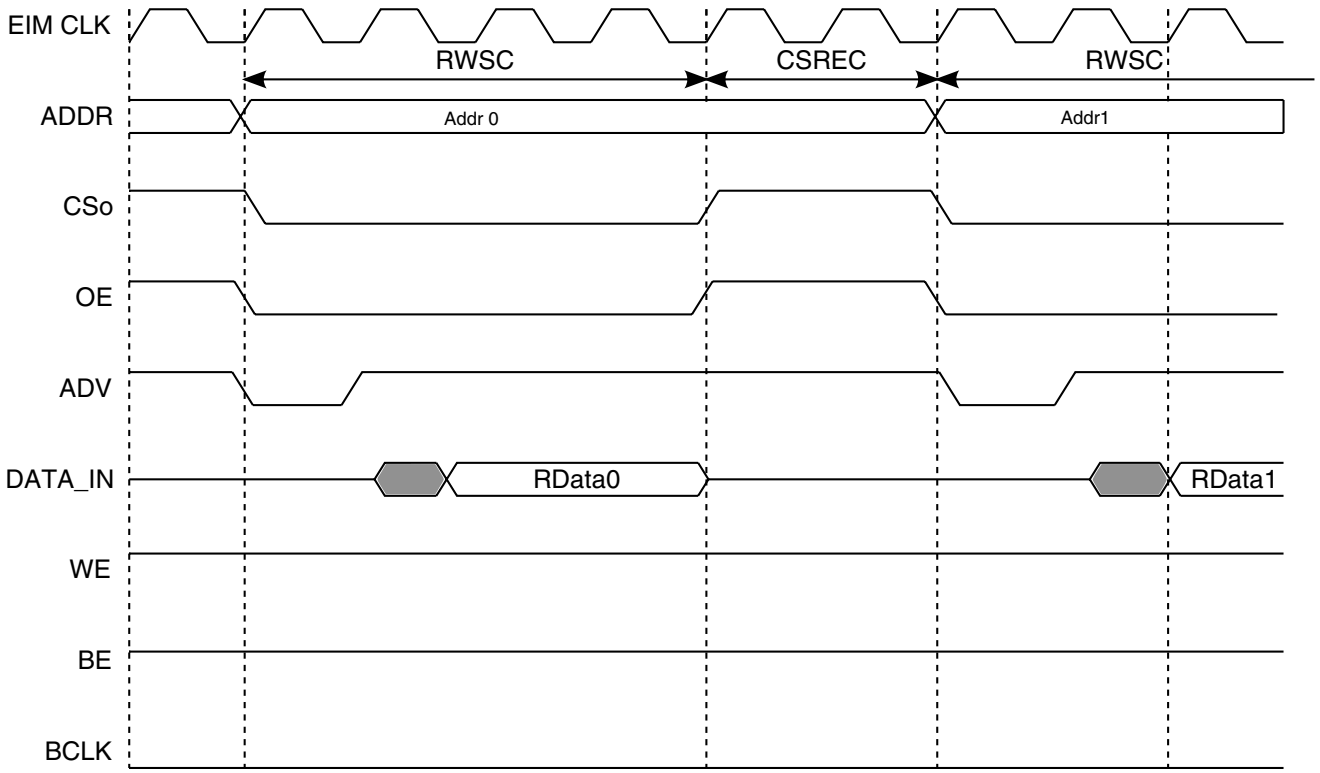


Figure 23-11. RWSC=4,RCSA=0,OEA=0,RADVA=0,RCSN=0,OEN=0,RADVN=0,CSREC=2

23.9.7 Async. Page Mode Access

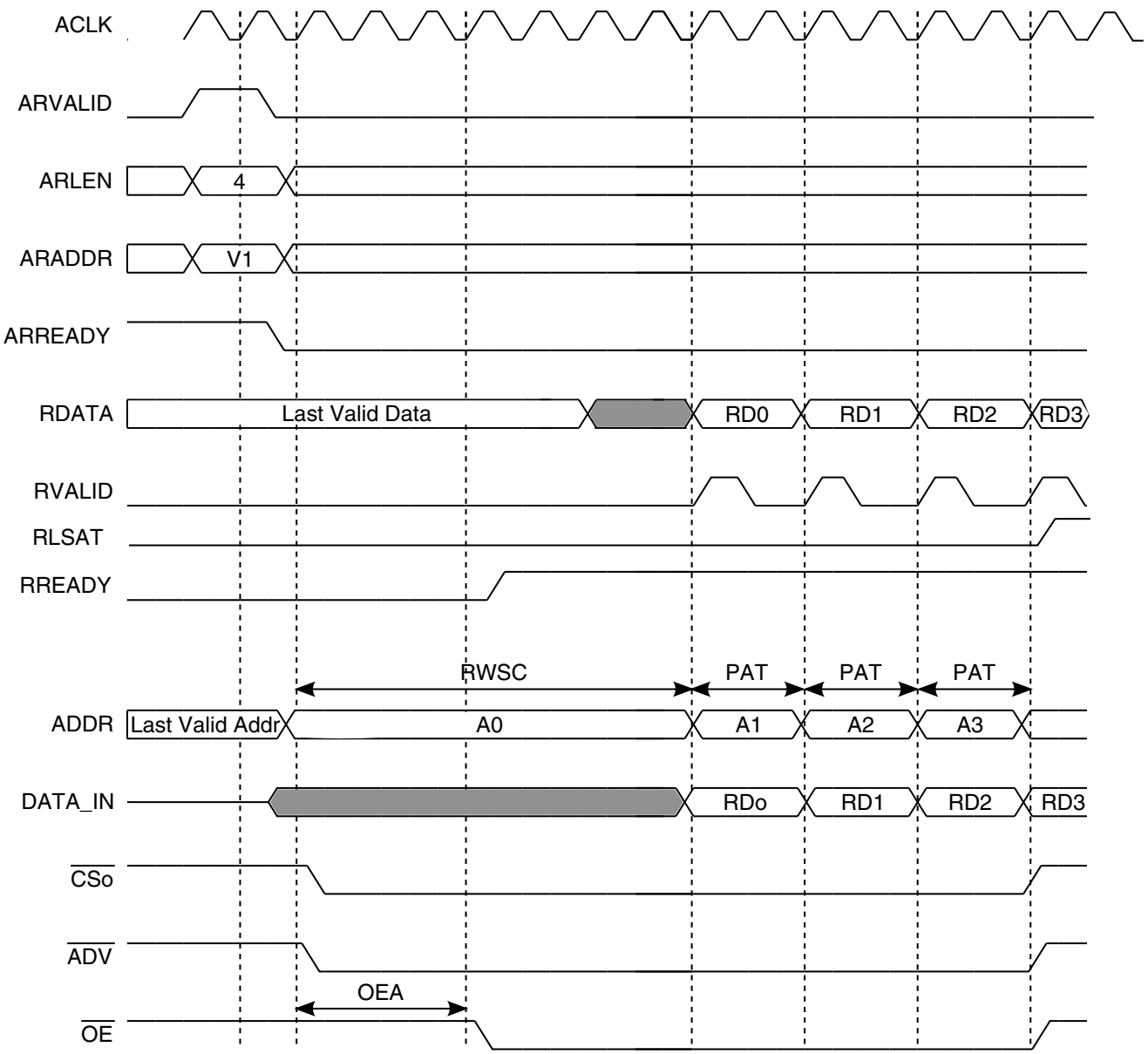


Figure 23-12. PAT = 2

23.9.8 DTACK Mode - AXI Single Access

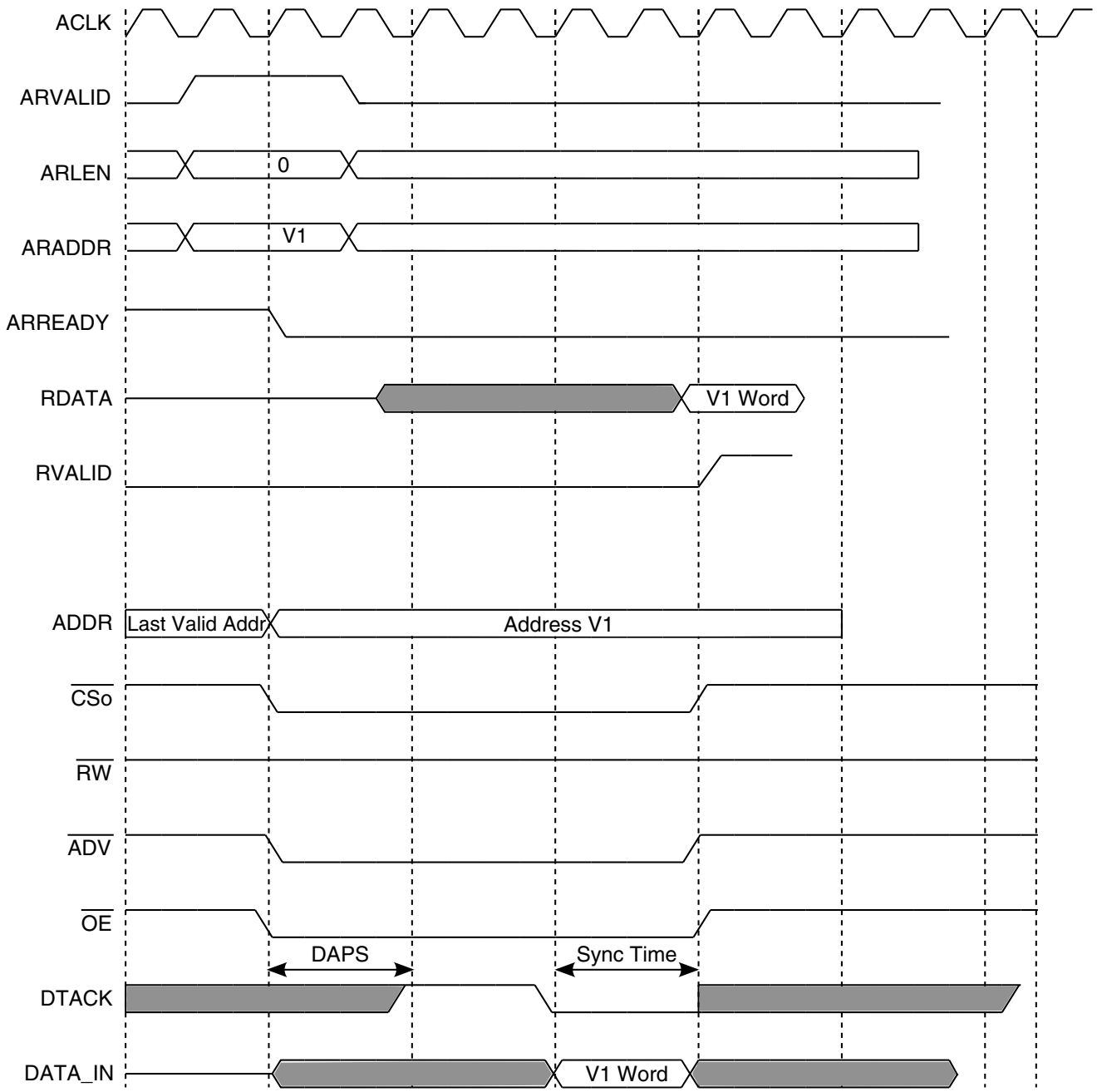


Figure 23-13. DAPS = 2

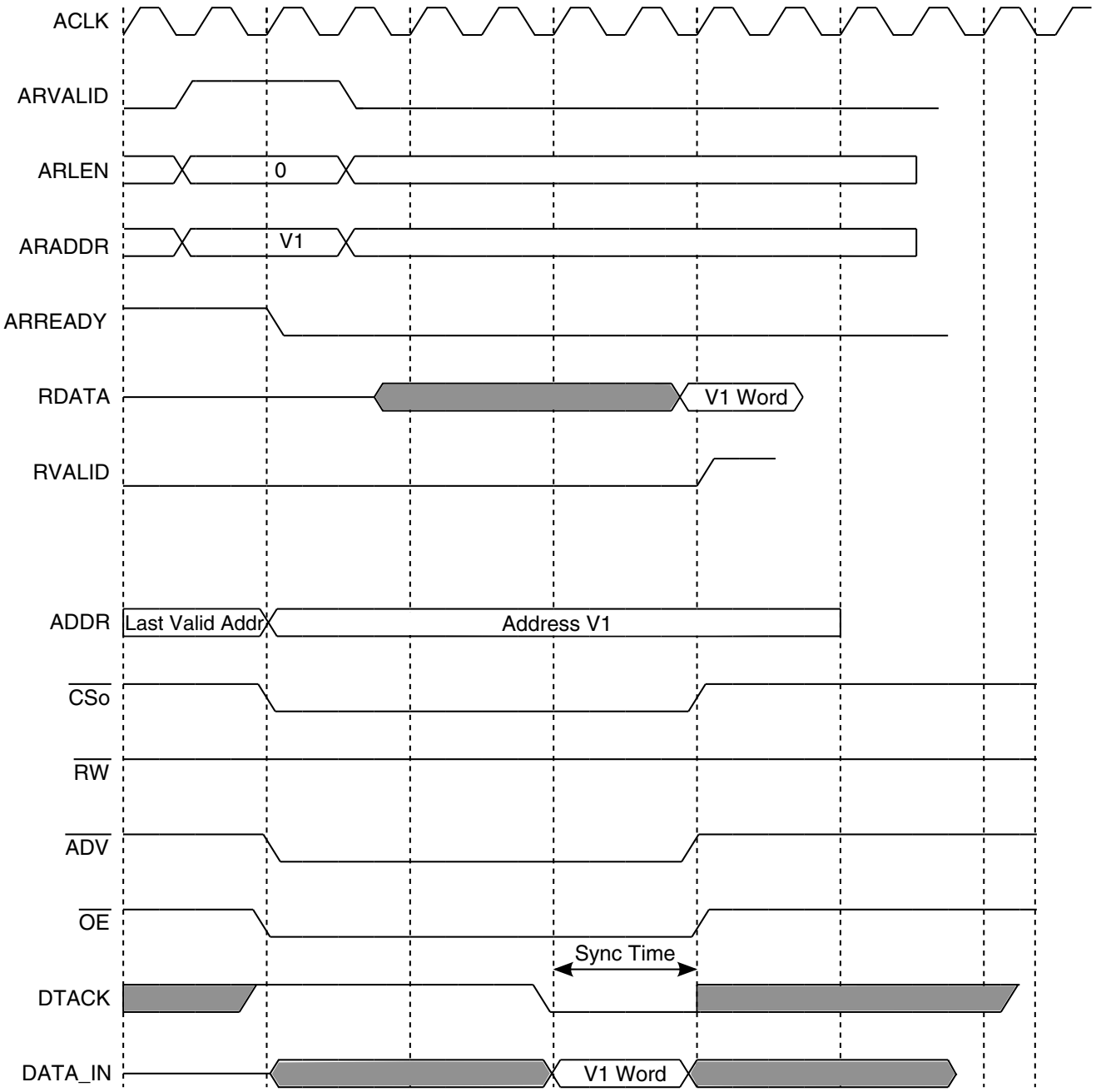


Figure 23-14. DAPS = 0

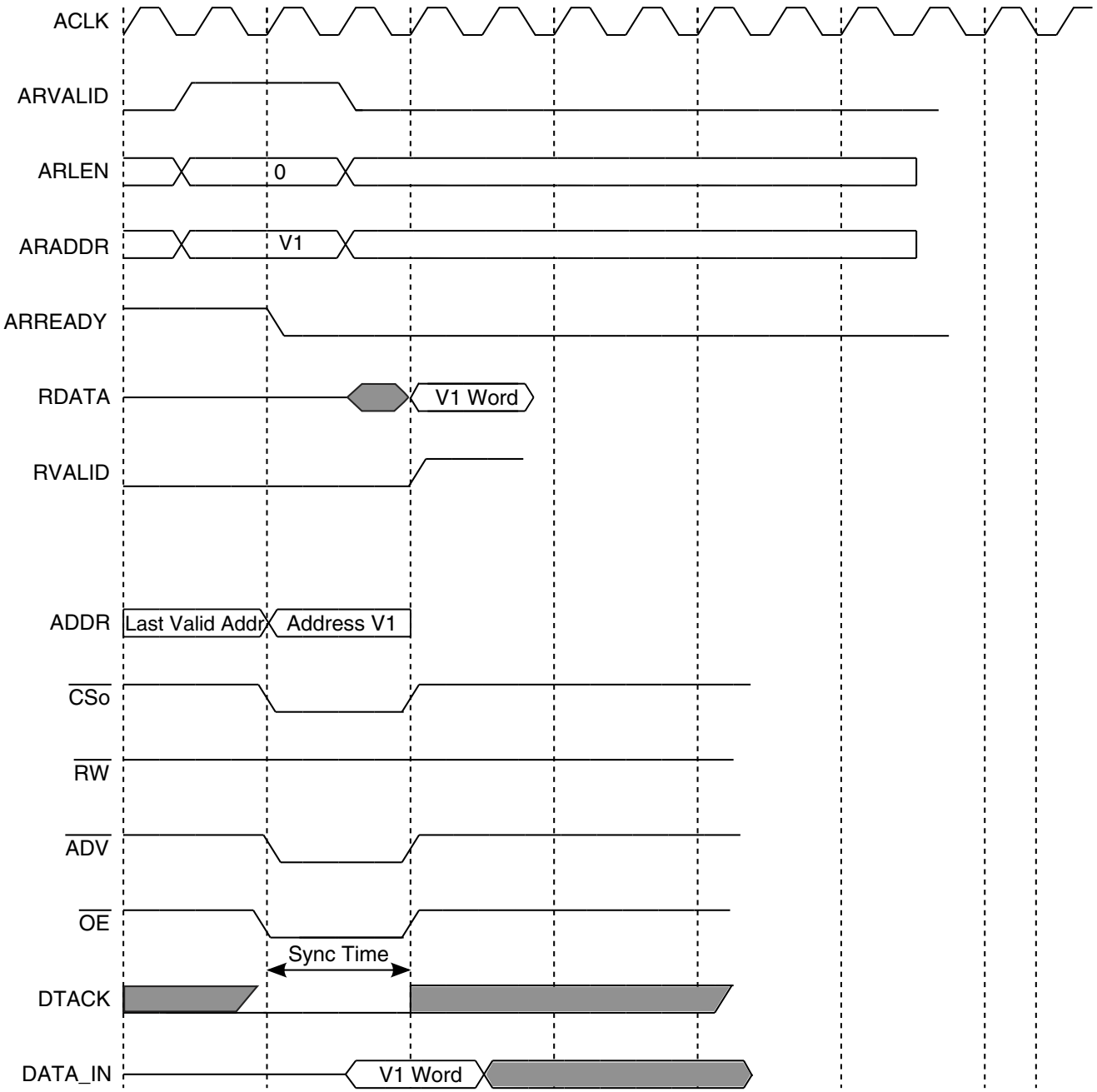


Figure 23-15. DAPS = 0

23.9.9 DTACK Mode - AXI Single Write Access

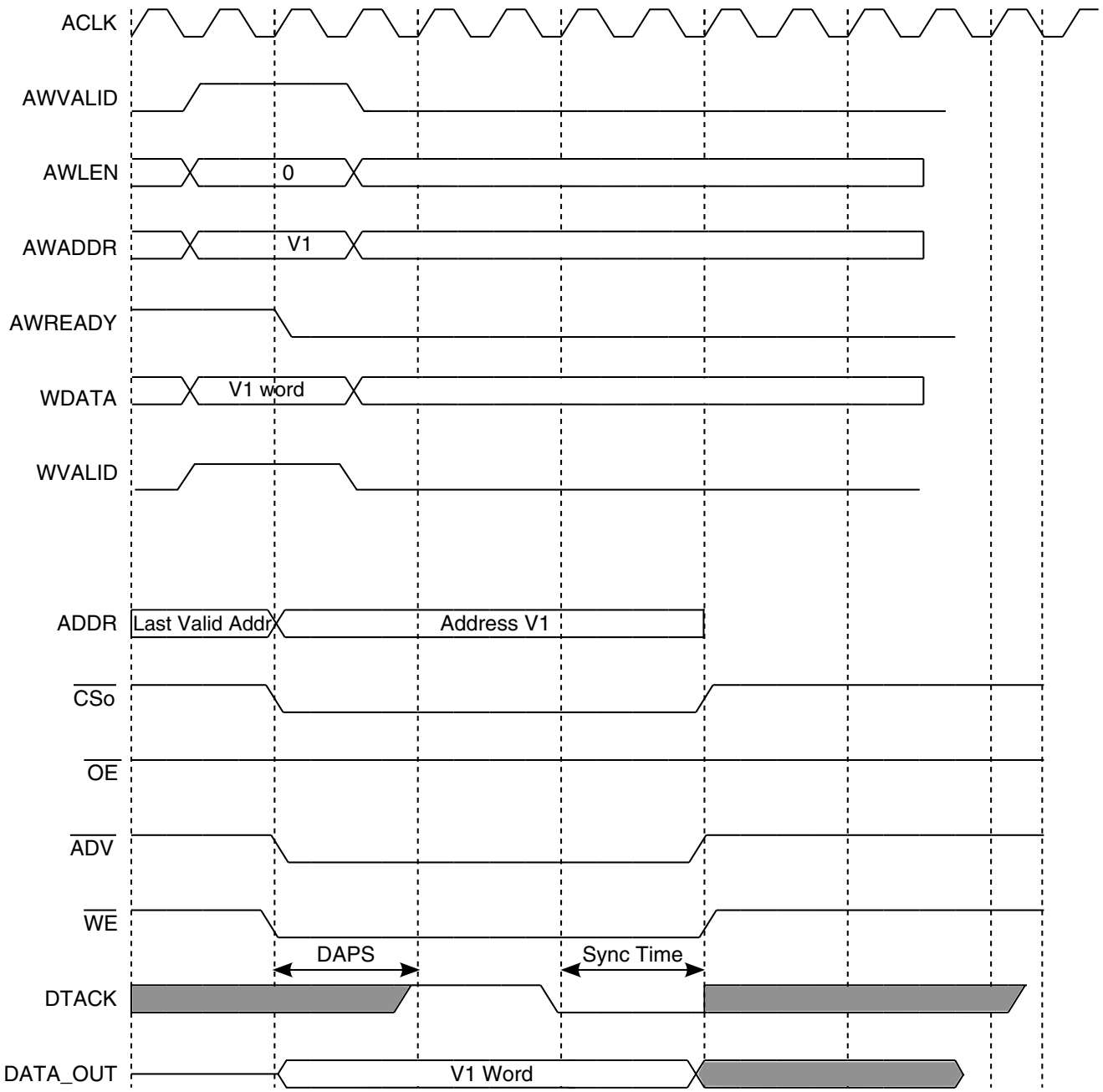


Figure 23-16. DAPS = 2

23.9.10 DTACK Mode - AXI Burst Access

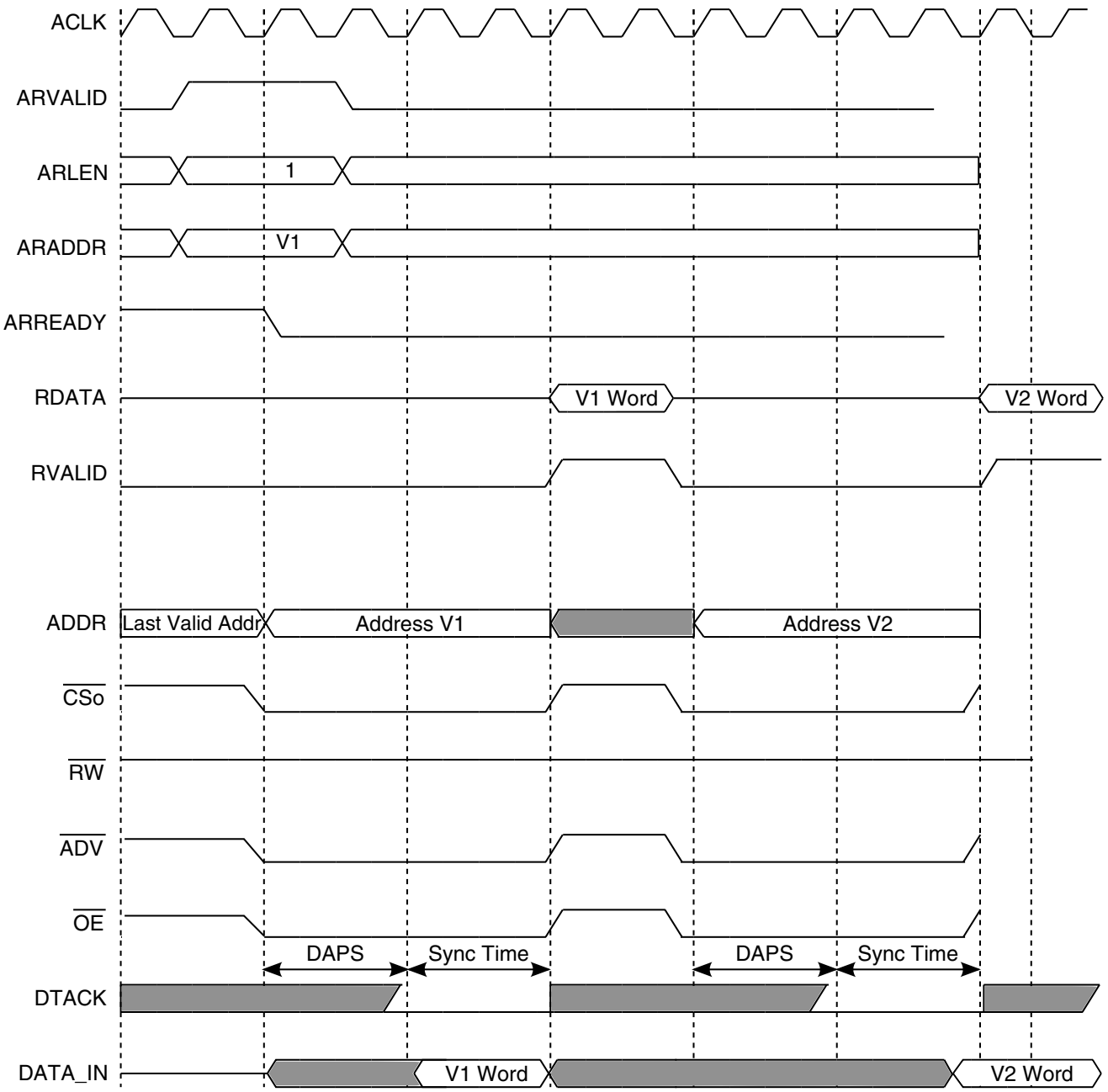


Figure 23-17. DAPS = 2 CSREC = 2

23.10 Programmable Registers

23.10.1 EIM Memory Map/Register Definition

The EIM includes 33 user-accessible 32-bit registers. The the EIM Configuration Register (EIM_WCR) contains control bits that configure the EIM for certain operation modes.

The 160 bits used to control Individual Chip Select are divided into five registers:

- Chip Select x General Configuration Register 1 (EIM_CSxGCR1)
- Chip Select x General Configuration Register 2 (EIM_CSxGCR2)
- Chip Select x Read Configuration Register 1 (EIM_CSxRCR1)
- Chip Select x Read Configuration Register 2 (EIM_CSxRCR2)
- Chip Select x Write Configuration Register (EIM_CSxWCR)

In addition there are 3 general registers: EIM_WCR, EIM_WIAR & EIM_EAR.

NOTE

- All EIM registers are sampled by IPG_CLK_S, therefore IPG_CLK_S must be active when accessing through IP bus.
- Read access from all registers (except EIM_WIAR & EIM_EAR) will generate one IPG_XFR_WAIT cycle.
- Read access from EIM_WIAR & EIM_EAR will generate six IPG_XFR_WAIT cycles.
- Write access to all registers (except EIM_EAR) will generate three IPG_XFR_WAIT cycles.
- Write access to EIM_EAR will generate six IPG_XFR_WAIT cycles.

EIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FD_8000	Chip Select n General Configuration Register 1 (EIM_CS0GCR1)	32	R/W	See section	23.101.1/ 1135
63FD_8004	Chip Select n General Configuration Register 2 (EIM_CS0GCR2)	32	R/W	0000_1000h	23.101.2/ 1139

Table continues on the next page...

EIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FD_8008	Chip Select n Read Configuration Register 1 (EIM_CS0RCR1)	32	R/W	See section	23.101.3/ 1141
63FD_800C	Chip Select n Read Configuration Register 2 (EIM_CS0RCR2)	32	R/W	0000_0000h	23.101.4/ 1144
63FD_8010	Chip Select n Write Configuration Register 1 (EIM_CS0WCR1)	32	R/W	See section	23.101.5/ 1145
63FD_8014	Chip Select n Write Configuration Register 2 (EIM_CS0WCR2)	32	R/W	0000_0000h	23.101.6/ 1148
63FD_8018	Chip Select n General Configuration Register 1 (EIM_CS1GCR1)	32	R/W	See section	23.101.1/ 1135
63FD_801C	Chip Select n General Configuration Register 2 (EIM_CS1GCR2)	32	R/W	0000_1000h	23.101.2/ 1139
63FD_8020	Chip Select n Read Configuration Register 1 (EIM_CS1RCR1)	32	R/W	See section	23.101.3/ 1141
63FD_8024	Chip Select n Read Configuration Register 2 (EIM_CS1RCR2)	32	R/W	0000_0000h	23.101.4/ 1144
63FD_8028	Chip Select n Write Configuration Register 1 (EIM_CS1WCR1)	32	R/W	See section	23.101.5/ 1145
63FD_802C	Chip Select n Write Configuration Register 2 (EIM_CS1WCR2)	32	R/W	0000_0000h	23.101.6/ 1148
63FD_8030	Chip Select n General Configuration Register 1 (EIM_CS2GCR1)	32	R/W	See section	23.101.1/ 1135
63FD_8034	Chip Select n General Configuration Register 2 (EIM_CS2GCR2)	32	R/W	0000_1000h	23.101.2/ 1139
63FD_8038	Chip Select n Read Configuration Register 1 (EIM_CS2RCR1)	32	R/W	See section	23.101.3/ 1141
63FD_803C	Chip Select n Read Configuration Register 2 (EIM_CS2RCR2)	32	R/W	0000_0000h	23.101.4/ 1144
63FD_8040	Chip Select n Write Configuration Register 1 (EIM_CS2WCR1)	32	R/W	See section	23.101.5/ 1145
63FD_8044	Chip Select n Write Configuration Register 2 (EIM_CS2WCR2)	32	R/W	0000_0000h	23.101.6/ 1148
63FD_8048	Chip Select n General Configuration Register 1 (EIM_CS3GCR1)	32	R/W	See section	23.101.1/ 1135
63FD_804C	Chip Select n General Configuration Register 2 (EIM_CS3GCR2)	32	R/W	0000_1000h	23.101.2/ 1139
63FD_8050	Chip Select n Read Configuration Register 1 (EIM_CS3RCR1)	32	R/W	See section	23.101.3/ 1141
63FD_8054	Chip Select n Read Configuration Register 2 (EIM_CS3RCR2)	32	R/W	0000_0000h	23.101.4/ 1144

Table continues on the next page...

EIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FD_8058	Chip Select n Write Configuration Register 1 (EIM_CS3WCR1)	32	R/W	See section	23.101.5/ 1145
63FD_805C	Chip Select n Write Configuration Register 2 (EIM_CS3WCR2)	32	R/W	0000_0000h	23.101.6/ 1148
63FD_8090	EIM Configuration Register (EIM_WCR)	32	R/W	0000_0020h	23.101.7/ 1149
63FD_8094	EIM IP Access Register (EIM_WIAR)	32	R/W	0000_0010h	23.101.8/ 1150
63FD_8098	Error Address Register (EIM_EAR)	32	R/W	0000_0000h	23.101.9/ 1151

23.101.1 Chip Select n General Configuration Register 1 (EIM_CSGCR1)

Addresses: EIM_CS0GCR1 is 63FD_8000h base + 0h offset = 63FD_8000h
EIM_CS1GCR1 is 63FD_8000h base + 18h offset = 63FD_8018h
EIM_CS2GCR1 is 63FD_8000h base + 30h offset = 63FD_8030h
EIM_CS3GCR1 is 63FD_8000h base + 48h offset = 63FD_8048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PSZ				WP	GBC			AUS	CSREC			SP	DSZ			BCS		BCD		WC	BL			CREP	CRE	RFL	WFL	MUM	SRD	SWR	CSEN
Reset	0	0	0	0	0	0	0	0	0	n*	n*	n*	0	n*	n*	n*	0	0	0	0	0	0	0	0	1	0	0	0	n*	0	0	0

* Notes:
- CSREC bitfield: See field descriptions for the reset values.
- DSZ bitfield: See field descriptions for the reset values.
- MUM bitfield: See field descriptions for the reset values.

EIM_CS*n*GCR1 field descriptions

Field	Description
31–28 PSZ	Page Size. This bit field indicates memory page size in words (word is defined by the DSZ field). PSZ is used when fix latency mode is applied, WFL=1 for sync. write accesses, RFL=1 for sync. Read accesses. When working in fix latency mode WAIT signal from the external device is not being monitored, PSZ is used to determine if page boundary is reached and renewal of access is preformed. This bit field is ignored when sync. Mode is disabled or fix latency mode is not being used for write or read access separately. It can be valid for both access type, read or write, or only for one type, according to configuration. PSZ is cleared by a hardware reset. 0000 8 words page size 0001 16 words page size 0010 32 words page size

Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
	0011 64 words page size 0100 128 words page size 0101 256 words page size 0110 512 words page size 0111 1024 (1k) words page size 1000 2048 (2k) words page size 1001 - 1111 Reserved
27 WP	Write Protect. This bit prevents writes to the address range defined by the corresponding chip select. WP is cleared by a hardware reset. 0 Writes are allowed in the memory range defined by chip. 1 Writes are prohibited. All attempts to write to an address mapped by this chip select result in a error response and no assertion of the chip select output.
26–24 GBC	Gap Between Chip Selects. This bit field, according to the settings shown below, determines the minimum time between end of access to the current chip select and start of access to different chip select. GBC is cleared by a hardware reset. Example settings: 000 minimum of 0 EIM clock cycles before next access from different chip select (async. mode only) 001 minimum of 1 EIM clock cycles before next access from different chip select 010 minimum of 2 EIM clock cycles before next access from different chip select 111 minimum of 7 EIM clock cycles before next access from different chip select
23 AUS	Address UnShifted. This bit indicates an unshifted mode for address assertion for the relevant chip select accesses. AUS bit is cleared by hardware reset. **NOTE:** The reset value for EIM_CS0GCR1[AUS] = EIM_BOOT[10]. For EIM_CS1GCR1 - EIM_CS5GCR1, the reset value of AUS is 0. 0 Address shifted according to port size (DSZ config.) 1 Address unshifted
22–20 CSREC	CS Recovery. This bit field, according to the settings shown below, determines the minimum pulse width of CS, OE, and WE control signals before executing a new back to back access to the same chip select. CSREC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0GCR1, CSREC[2:1] is EIM_BOOT[9:8], for CSREC[0] is 0. For EIM_CS1GCR1 - EIM_CS5GCR, the reset value is 0b000. Example settings: 000 0 EIM clock cycles minimum width of CS, OE and WE signals (read async. mode only) 001 1 EIM clock cycles minimum width of CS, OE and WE signals 010 2 EIM clock cycles minimum width of CS, OE and WE signals 111 7 EIM clock cycles minimum width of CS, OE and WE signals
19 SP	Supervisor Protect. This bit prevents accesses to the address range defined by the corresponding chip select when the access is attempted in the User mode. SP is cleared by a hardware reset. 0 User mode accesses are allowed in the memory range defined by chip select. 1 User mode accesses are prohibited. All attempts to access an address mapped by this chip select in User mode results in an error response and no assertion of the chip select output.

Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
18–16 DSZ	Data Port Size. This bit field defines the width of an external device's data port as shown below. **NOTE:** Only async. access supported for 8 bit port. **NOTE:** The reset value for EIM_CS0GCR1[DSZ] = {EIM_BOOT[11], EIM_BOOT[1:0]}. For EIM_CS1GCR1 - EIM_CS5GCR1, the reset value is 0b001. 000 Reserved. 001 16 bit port resides on DATA[15:0] 010 16 bit port resides on DATA[31:16] 011 32 bit port resides on DATA[31:0] 100 8 bit port resides on DATA[7:0] 101 8 bit port resides on DATA[15:8] 110 8 bit port resides on DATA[23:16] 111 8 bit port resides on DATA[31:24]
15–14 BCS	Burst Clock Start. When SRD=1 or SWR=1,this bit field determines the number of EIM clock cycles delay from start of access before the first rising edge of BCLK is generated. When BCD=0 value of BCS=0 results in a half clock delay after the start of access. For other values of BCD a one clock delay after the start of access is applied, not an immediate assertion. BCS is cleared by a hardware reset. 00 0 EIM clock cycle additional delay 01 1 EIM clock cycle additional delay 10 2 EIM clock cycle additional delay 11 3 EIM clock cycle additional delay
13–12 BCD	Burst Clock Divisor. This bit field contains the value used to program the burst clock divisor for BCLK generation. It is used to divide the internal EIMbus frequency. BCD is cleared by a hardware reset. **NOTE:** For other then the mentioned below frequency such as 104 MHz, EIM clock (input clock) should be adjust accordingly. 00 Divide EIM clock by 1 01 Divide EIM clock by 2 10 Divide EIM clock by 3 11 Divide EIM clock by 4
11 WC	Write Continuous. The WI bit indicates that write access to the memory are always continuous accesses regardless of the BL field value. WI is cleared by hardware reset. 0 Write access burst length occurs according to BL value. 1 Write access burst length is continuous.
10–8 BL	Burst Length. The BL bit field indicates memory burst length in words (word is defined by the DSZ field) and should be properly initialized for mixed wrap/increment accesses support. Continuous BL value corresponds to continuous burst length setting of the external memory device. For fix memory burst size, type is always wrap. In case not matching wrap boundaries in both the memory (BL field) and Master access on the current address, EIM update address on the external device address bus and regenerates the access. BL is cleared by a hardware reset. When APR=1, Page Read Mode is applied, BL determine the number of words within the read page burst. BL is cleared by a hardware reset for EIM_CS0GCR1 - EIM_CS5GCR1.

Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
	000 4 words Memory wrap burst length (read page burst size when APR = 1) 001 8 words Memory wrap burst length (read page burst size when APR = 1) 010 16 words Memory wrap burst length (read page burst size when APR = 1) 011 32 words Memory wrap burst length (read page burst size when APR = 1) 100 Continuous burst length (2 words read page burst size when APR = 1) 101 Reserved 110 Reserved 111 Reserved
7 CREP	Configuration Register Enable Polarity. This bit indicates CRE memory pin assertion state, active-low or active-high, while executing a memory register set command to the external device (PSRAM memory type). CREP is set by a hardware reset. **NOTE:** Whenever PSRAM is connected the CREP value must be correct also for accesses where CRE is disabled. For Non-PSRAM memory CREP value should be 1. 0 CRE signal is active low 1 CRE signal is active high
6 CRE	Configuration Register Enable. This bit indicates CRE memory pin state while executing a memory register set command to PSRAM external device. CRE is cleared by a hardware reset. 0 CRE signal use is disable 1 CRE signal use is enable
5 RFL	Read Fix Latency. This bit field determine if the controller is monitoring the WAIT signal from the External device connected to the chip select (handshake mode - fix or variable data latency) or if it start sampling data according to RWSC field, it only valid in synchronous mode. RFL is cleared by a hardware reset. When RFL=1 Burst access is terminated on page boundary and resume on the following page according to BL bit field configuration, because WAIT signal is not monitored from the external device. 0 the External device WAIT signal is being monitored, and it reflect the external data bus state 1 the state of the External devices is determined internally (Fix latency mode only)
4 WFL	Write Fix Latency. This bit field determine if the controller is monitoring the WAIT signal from the External device connected to the chip select (handshake mode - fix or variable data latency) or if it start data transfer according to WWSC field, it only valid in synchronous mode. WFL is cleared by a hardware reset. When WFL=1 Burst access is terminated on page boundary and resume on the following page according to BL bit field configuration, because WAIT signal is not monitored from the external device 0 the External device WAIT signal is being monitored, and it reflect the external data bus state 1 the state of the External devices is determined internally (Fix latency mode only)
3 MUM	Multiplexed Mode. This bit determines the address/data multiplexed mode for asynchronous and synchronous accesses for 8 bit, 16 bit or 32 bit devices (DSZ config. dependent). **NOTE:** The reset value for EIM_CS0GCR1[MUM] = EIM_BOOT[2]. For EIM_CS1GCR1 - EIM_CS5GCR1 the reset value is 0. 0 Multiplexed Mode disable 1 Multiplexed Mode enable

Table continues on the next page...

EIM_CS*n*GCR1 field descriptions (continued)

Field	Description
2 SRD	Synchronous Read Data. This bit field determine the read accesses mode to the External device of the chip select. The External device should be configured to the same mode as this bit implicates. SRD is cleared by a hardware reset. **NOTE:** Sync. accesses supported only for 16/32 bit port. 0 read accesses are in Asynchronous mode 1 read accesses are in Synchronous mode
1 SWR	Synchronous Write Data. This bit field determine the write accesses mode to the External device of the chip select. The External device should be configured to the same mode as this bit implicates. SWR is cleared by a hardware reset. **NOTE:** Sync. accesses supported only for 16/32 bit port. 0 write accesses are in Asynchronous mode 1 write accesses are in Synchronous mode
0 CSEN	CS Enable. This bit controls the operation of the chip select pin. CSEN is set by a hardware reset for CSGCR0 to allow external boot operation. CSEN is cleared by a hardware reset to CSGCR1-CSGCR5. **NOTE:** Reset value for EIM_CS0GCR1 for CSEN is 1. For EIM_CS1GCR1-CS1GCR5 reset value is 0. 0 Chip select function is disabled; attempts to access an address mapped by this chip select results in an error respond and no assertion of the chip select output 1 Chip select is enabled, and is asserted when presented with a valid access.

23.101.2 Chip Select n General Configuration Register 2 (EIM_CSGCR2)

Addresses: EIM_CS0GCR2 is 63FD_8000h base + 4h offset = 63FD_8004h
EIM_CS1GCR2 is 63FD_8000h base + 1Ch offset = 63FD_801Ch
EIM_CS2GCR2 is 63FD_8000h base + 34h offset = 63FD_8034h
EIM_CS3GCR2 is 63FD_8000h base + 4Ch offset = 63FD_804Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			MUX16_BYP_GRANT	0		DAP	DAE	DAPS				0		ADH	
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*GCR2 field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value zero. Reserved
12 MUX16_BYP_GRANT	Muxed 16 bypass grant. This bit when asserted causes EIM to bypass the grant/ack. arbitration with NFC (only for 16 bit muxed mode accesses). **NOTE:** The reset value for EIM_CS0GCR2[MUX16_BYP_GRANT] = EIM_BOOT[12]. For EIM_CS1GCR2 - EIM_CS5GCR2, MUX16_BYP_GRANT reset value is 1. 0 EIM waits for grant before driving a 16 bit muxed mode access to the memory. 1 EIM ignores the grant signal and immediately drives a 16 bit muxed mode access to the memory.
11–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9 DAP	Data Acknowledge Polarity. This bit indicates DTACK memory pin assertion state, active-low or active-high, while executing an async access using DTACK signal from the external device. DAP is cleared by a hardware reset. 0 DTACK signal is active high 1 DTACK signal is active low
8 DAE	Data Acknowledge Enable. This bit indicates external device is using DTACK pin as strobe/terminator of an async. access. DTACK signal may be used only in asynchronous single read (APR=0) or write accesses. DTACK poling start point is set by DAPS bit field. polarity of DTACK is set by DAP bit field. DAE is cleared by a hardware reset. 0 DTACK signal use is disable 1 DTACK signal use is enable
7–4 DAPS	Data Acknowledge Poling Start. This bit field determine the starting point of DTACK input signal polling. DAPS is used only in asynchronous single read or write accesses. **NOTE:** Since DTACK is an async. signal the start point of DTACK signal polling is at least 3 cycles after the start of access. DAPS is cleared by a hardware reset. Example settings: 0000 3 EIM clk cycle between start of access and first $\overline{\text{DTACK}}$ check 0001 4 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 0010 5 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 0111 10 EIM clk cycles between start of access and first $\overline{\text{DTACK}}$ check 1011 14 EIM clk cycles between start of access and first DTACK check 1111 18 EIM clk cycles between start of access and first DTACK check
3–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1–0 ADH	Address hold time - This bit field determine the address hold time after ADV negation when mum = 1 (muxed mode). When mum = 0 this bit has no effect. For read accesses the field determines when the pads direction will be switched. **NOTE:** Reset value for EIM_CS0GCR2 for ADH is 10. For EIM_CS1GCR2-EIM_CS5GCR2 reset value is 00.

Table continues on the next page...

EIM_CS*n*GCR2 field descriptions (continued)

Field	Description
	00 0 cycle after ADV negation 01 1 cycle after ADV negation 10 2 cycle after ADV negation 11 Reserved

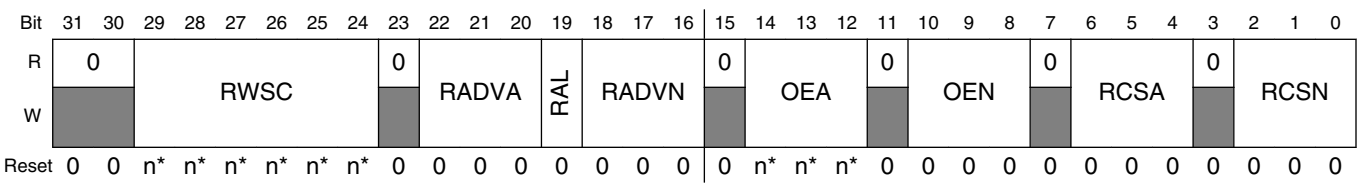
23.101.3 Chip Select n Read Configuration Register 1 (EIM_CSRCR1)

Addresses: EIM_CS0RCR1 is 63FD_8000h base + 8h offset = 63FD_8008h

EIM_CS1RCR1 is 63FD_8000h base + 20h offset = 63FD_8020h

EIM_CS2RCR1 is 63FD_8000h base + 38h offset = 63FD_8038h

EIM_CS3RCR1 is 63FD_8000h base + 50h offset = 63FD_8050h



* Notes:

- RWSC bitfield: See field descriptions for the reset values.
- OEA bitfield: See field descriptions for the reset values.

EIM_CS*n*RCR1 field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value zero. Reserved
29–24 RWSC	Read Wait State Control. This bit field programs the number of wait-states, according to the settings shown below, for synchronous or asynchronous read access to the external device connected to the chip select. When SRD=1 and RFL=0, RWSC indicates the number of burst clock (BCLK) cycles from the start of an access, before the controller can start sample data.Since WAIT signal can be asserted one cycle before the first data can be sampled, the controller starts evaluating the WAIT signal state one cycle before, this is referred as handshake mode or variable latency mode. When SRD=1 and RFL=1, RWSC indicates the number of burst clock (BCLK) cycles from the start of an access, until the external device is ready for data transfer, this is referred as fix latency mode. When SRD=0, RFL bit is ignored, RWSC indicates the asynchronous access length and the number of EIM clock cycles from the start of access until the external device is ready for data transfer. RWSC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0RCR1[RWSC[4:2]] = EIM_BOOT [7:5]. For {RWSC[5], RWSC[1:0]} the reset value is 0b000. For CG1RCR1 - CS1RCR5 the reset value is 0b000000. Example settings: 000000 Reserved

Table continues on the next page...

EIM_CS*n*RCR1 field descriptions (continued)

Field	Description
	000001 RWSC value is 1 000010 RWSC value is 2 111101 RWSC value is 61 111110 RWSC value is 62 111111 RWSC value is 63
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22–20 RADVA	ADV Assertion. This bit field determines when ADV signal is asserted for synchronous or asynchronous read modes according to the settings shown below. RADVA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of access and ADV assertion 001 1 EIM clock cycles between beginning of access and ADV assertion 010 2 EIM clock cycles between beginning of access and ADV assertion 111 7 EIM clock cycles between beginning of access and ADV assertion
19 RAL	Read ADV Low. This bit field determine ADV signal negation time. When RAL=1, RADVN bit field is ignored and ADV signal will stay asserted until end of access. When RAL=0 negation of ADV signal is according to RADVN bit field configuration. **NOTE:** The reset value of EIM_CS0RCR1[RAL] = EIM_BOOT[3]. RAL is cleared by a hardware reset for EIM_CS1RCR1 - EIM_CS5RCR1.
18–16 RADVN	ADV Negation. This bit field determines when ADV signal to memory is negated during read accesses. When SRD=1 (synchronous read mode), ADV negation occurs according to the following formula: (RADVN + RADVA + BCD + BCS + 1) EIM clock cycles from start of access. When asynchronous read mode is applied (SRD=0) and RAL=0 ADV negation occurs according to the following formula: (RADVN + RADVA + 1) EIM clock cycles from start of access. RADVN is cleared by a hardware reset. **NOTE:** the reset value for EIM_CS0RCR1[RADVN] = 2. For EIM_CS1RCR1 - EIM_CS5RCR1, the reset value is 0b000. **NOTE:** This field should be configured so ADV negation will occur before the end of access. For ADV negation at the same time with the end of access user should RAL bit.
15 Reserved	This read-only field is reserved and always has the value zero. Reserved
14–12 OEA	OE Assertion. This bit field determines when OE signal are asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. OEA is cleared by a hardware reset. In muxed mode OE assertion occurs (OEA + RADVN + RADVA + ADH +1) EIM clock cycles from start of access. **NOTE:** The reset value for EIM_CS0RCR1[OEA] is 0b000 if EIM_BOOT[2] = 0. If EIM_BOOT[2] is 1, the reset value for EIM_CS0RCR1 is 0b010. The reset value of this field for EIM_CS1RCR1 - EIM_CS5RCR1 is 0b000. Example settings: 000 0 EIM clock cycles between beginning of access and OE assertion 001 1 EIM clock cycles between beginning of access and OE assertion

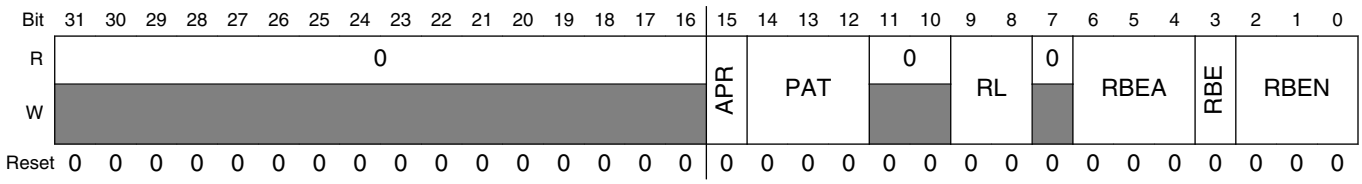
Table continues on the next page...

EIM_CS*n*RCR1 field descriptions (continued)

Field	Description
	010 2 EIM clock cycles between beginning of access and OE assertion 111 7 EIM clock cycles between beginning of access and OE assertion
11 Reserved	This read-only field is reserved and always has the value zero. Reserved
10–8 OEN	OE Negation. This bit field determines when OE signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR = 0), according to the settings shown below. This bit field is ignored when SRD=1. OEN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of access and OE negation 001 1 EIM clock cycles between end of access and OE negation 010 2 EIM clock cycles between end of access and OE negation 111 7 EIM clock cycles between end of access and OE negation
7 Reserved	This read-only field is reserved and always has the value zero. Reserved
6–4 RCSA	Read CS Assertion. This bit field determines when CS signal is asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. RCSA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of read access and CS assertion 001 1 EIM clock cycles between beginning of read access and CS assertion 010 2 EIM clock cycles between beginning of read access and CS assertion 111 7 EIM clock cycles between beginning of read access and CS assertion
3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2–0 RCSN	Read CS Negation. This bit field determines when CS signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR = 0), according to the settings shown below. This bit field is ignored when SRD=1. RCSN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and CS negation 001 1 EIM clock cycles between end of read access and CS negation 010 2 EIM clock cycles between end of read access and CS negation 111 7 EIM clock cycles between end of read access and CS negation

23.101.4 Chip Select n Read Configuration Register 2 (EIM_CSRCR2)

Addresses: EIM_CS0RCR2 is 63FD_8000h base + Ch offset = 63FD_800Ch
EIM_CS1RCR2 is 63FD_8000h base + 24h offset = 63FD_8024h
EIM_CS2RCR2 is 63FD_8000h base + 3Ch offset = 63FD_803Ch
EIM_CS3RCR2 is 63FD_8000h base + 54h offset = 63FD_8054h



EIM_CS*n*RCR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 APR	Asynchronous Page Read. This bit field determine the asynchronous read mode to the external device. When APR=0, the async. read access is done as single word (where word is defined by the DSZ field). when APR=1, the async. read access executed as page read. page size is according to BL field config., RCSN,RBEN,OEN and RADVN are being ignored. APR is cleared by a hardware reset for EIM_CS1GCR1 - EIM_CS5GCR1. **NOTE:** SRD=0 and MUM=0 must apply when APR=1
14–12 PAT	Page Access Time. This bit field is used in Asynchronous Page Read mode only (APR=1). the initial access is set by RWSC as in regular asynchronous mode. the consecutive address assertions width determine by PAT field according to the settings shown below. when APR=0 this field is ignored. PAT is cleared by a hardware reset for EIM_CS1GCR1 - EIM_CS5GCR1. 000 Address width is 2 EIM clock cycles 001 Address width is 3 EIM clock cycles 010 Address width is 4 EIM clock cycles 011 Address width is 5 EIM clock cycles 100 Address width is 6 EIM clock cycles 101 Address width is 7 EIM clock cycles 110 Address width is 8 EIM clock cycles 111 Address width is 9 EIM clock cycles
11–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9–8 RL	Read Latency. This bit field indicates cycle latency when executing a synchronous read operation. The fields holds the feedback clock loop delay in aclk cycle units. This field is cleared by a hardware reset. 00 Feedback clock loop delay is up to 1 cycle for BCD = 0 or 1.5 cycles for BCD != 0 01 Feedback clock loop delay is up to 2 cycles for BCD = 0 or 2.5 cycles for BCD != 0 10 Feedback clock loop delay is up to 3 cycles for BCD = 0 or 3.5 cycles for BCD != 0 11 Feedback clock loop delay is up to 4 cycles for BCD = 0 or 4.5 cycles for BCD != 0

Table continues on the next page...

EIM_CS*n*RCR2 field descriptions (continued)

Field	Description
7 Reserved	This read-only field is reserved and always has the value zero. Reserved
6–4 RBEA	Read BE Assertion. This bit field determines when BE signal is asserted during read cycles (synchronous or asynchronous mode), according to the settings shown below. RBEA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of read access and BE assertion 001 1 EIM clock cycles between beginning of read access and BE assertion 010 2 EIM clock cycles between beginning of read access and BE assertion 111 7 EIM clock cycles between beginning of read access and BE assertion
3 RBE	Read BE enable. This bit field determines if BE will be asserted during read access. 0 - BE are disabled during read access. 1- BE are enable during read access according to value of RBEA & RBEN bit fields.
2–0 RBEN	Read BE Negation. This bit field determines when BE signal is negated during read cycles in asynchronous single mode only (SRD=0 & APR=0), according to the settings shown below. This bit field is ignored when SRD=1. RBEN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and BE negation 001 1 EIM clock cycles between end of read access and BE negation 010 2 EIM clock cycles between end of read access and BE negation 111 7 EIM clock cycles between end of read access and BE negation

23.101.5 Chip Select n Write Configuration Register 1 (EIM_CSWCR1)

Addresses: EIM_CS0WCR1 is 63FD_8000h base + 10h offset = 63FD_8010h
EIM_CS1WCR1 is 63FD_8000h base + 28h offset = 63FD_8028h
EIM_CS2WCR1 is 63FD_8000h base + 40h offset = 63FD_8040h
EIM_CS3WCR1 is 63FD_8000h base + 58h offset = 63FD_8058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WAL	WBED	WWSC						WADVA			WADVN			WBEA		WBEN				WEA			WEN			WCSA			WCSN		
Reset	0	0	n*	n*	n*	n*	n*	n*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- WWSC bitfield: See field descriptions for the reset values.

EIM_CS*n*WCR1 field descriptions

Field	Description
31 WAL	Write ADV Low. This bit field determine ADV signal negation time in write accesses. When WAL=1, WADVN bit field is ignored and ADV signal will stay asserted until end of access. When WAL=0 negation of ADV signal is according to WADVN bit field configuration. **NOTE:** The reset value of CS0WCR1[WAL] = EIM_BOOT[3]. This field is cleared by a hardware reset for CS1WCR1 - CS5WCR1.
30 WBED	Write Byte Enable Disable. When asserted this bit prevent from IPP_DO_BE_B[x] to be asserted during write accesses.This bit is cleared by hardware reset.
29–24 WWSC	Write Wait State Control. This bit field programs the number of wait-states, according to the settings shown below, for synchronous or asynchronous write access to the external device connected to the chip select. When SWR=1 and WFL=0, WWSC indicates the number of burst clock (BCLK) cycles from the start of an access, before the memory can sample the first data.Since WAIT signal can be asserted one cycle before the first data can be sampled, the controller starts evaluating the WAIT signal state one cycle before, this is referred as handshake mode or variable latency mode. When SWR=1 and WFL=1, WWSC indicates the number of burst clock (BCLK) cycles from the start of an access, until the external device is ready for data transfer, this is referred as fix latency mode. When SWR=0, WFL bit is ignored, WWSC indicates the asynchronous access length and the number of EIM clock cycles from the start of access until the external device is ready for data transfer. WWSC is cleared by a hardware reset. **NOTE:** The reset value for EIM_CS0WCR1[WWSC[4:2]] = EIM_BOOT [7:5], {WWSC[5], WWSC[1:0]} = 0b000. For EIM_CS1WCR1 - EIM_CS5WCR1, the reset value of this field is 0b000000. Example settings: 000000 Reserved 000001 WWSC value is 1 000010 WWSC value is 2 000011 WWSC value is 3 111111 WWSC value is 63
23–21 WADVA	ADV Assertion. This bit field determines when ADV signal is asserted for synchronous or asynchronous write modes according to the settings shown below. WADVA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of access and ADV assertion 001 1 EIM clock cycles between beginning of access and ADV assertion 010 2 EIM clock cycles between beginning of access and ADV assertion 111 7 EIM clock cycles between beginning of access and ADV assertion
20–18 WADVN	ADV Negation. This bit field determines when ADV signal to memory is negated during write accesses. When SWR=1 (synchronous write mode), ADV negation occurs according to the following formula: (WADVN + WADVA + BCD + BCS + 1) EIM clock cycles. When asynchronous read mode is applied (SWR=0) ADV negation occurs according to the following formula: (WADVN + WADVA + 1) EIM clock cycles. **NOTE:** Reset value for EIM_CS0WCR for WADVN is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. **NOTE:** This field should be configured so ADV negation will occur before the end of access. For ADV negation at the same time as the end of access, S/W should set the WAL bit.

Table continues on the next page...

EIM_CS*n*WCR1 field descriptions (continued)

Field	Description
17–15 WBEA	BE Assertion. This bit field determines when BE signal is asserted during write cycles in async. mode only (SWR=0), according to the settings shown below. BEA is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WBEA is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. Example settings: 000 0 EIM clock cycles between beginning of access and BE assertion 001 1 EIM clock cycles between beginning of access and BE assertion 010 2 EIM clock cycles between beginning of access and BE assertion 111 7 EIM clock cycles between beginning of access and BE assertion
14–12 WBEN	BE[3:0] Negation. This bit field determines when BE[3:0] bus signal is negated during write cycles in async. mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. BEN is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WBEN is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. Example settings: 000 0 EIM clock cycles between end of access and WE negation 001 1 EIM clock cycles between end of access and WE negation 010 2 EIM clock cycles between end of access and WE negation 111 7 EIM clock cycles between end of access and WE negation
11–9 WEA	WE Assertion. This bit field determines when WE signal is asserted during write cycles (synchronous or asynchronous mode), according to the settings shown below. This bit field is ignored when executing a read access to the external device. WEA is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WEA is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. Example settings: 000 0 EIM clock cycles between beginning of access and WE assertion 001 1 EIM clock cycles between beginning of access and WE assertion 010 2 EIM clock cycles between beginning of access and WE assertion 111 7 EIMclock cycles between beginning of access and WE assertion
8–6 WEN	WE Negation. This bit field determines when WE signal is negated during write cycles in asynchronous mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. WEN is cleared by a hardware reset. **NOTE:** Reset value for EIM_CS0WCR for WEN is 2. For EIM_CS1WCR - EIM_CS5WCR reset value is 000. Example settings: 000 0 EIM clock cycles between beginning of access and WE assertion 001 1 EIM clock cycles between beginning of access and WE assertion 010 2 EIM clock cycles between beginning of access and WE assertion 111 7 EIM clock cycles between beginning of access and WE assertion

Table continues on the next page...

EIM_CS*n*WCR1 field descriptions (continued)

Field	Description
5–3 WCSA	Write CS Assertion. This bit field determines when CS signal is asserted during write cycles (synchronous or asynchronous mode), according to the settings shown below.this bit field is ignored when executing a read access to the external device. WCSA is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between beginning of write access and CS assertion 001 1 EIM clock cycles between beginning of write access and CS assertion 010 2 EIM clock cycles between beginning of write access and CS assertion 111 7 EIMclock cycles between beginning of write access and CS assertion
2–0 WCSN	Write CS Negation. This bit field determines when CS signal is negated during write cycles in asynchronous mode only (SWR=0), according to the settings shown below. This bit field is ignored when SWR=1. WCSN is cleared by a hardware reset. Example settings: 000 0 EIM clock cycles between end of read access and CS negation 001 1 EIM clock cycles between end of read access and CS negation 010 2 EIM clock cycles between end of read access and CS negation 111 7 EIM clock cycles between end of read access and CS negation

23.101.6 Chip Select n Write Configuration Register 2 (EIM_CSWCR2)

Addresses: EIM_CS0WCR2 is 63FD_8000h base + 14h offset = 63FD_8014h
EIM_CS1WCR2 is 63FD_8000h base + 2Ch offset = 63FD_802Ch
EIM_CS2WCR2 is 63FD_8000h base + 44h offset = 63FD_8044h
EIM_CS3WCR2 is 63FD_8000h base + 5Ch offset = 63FD_805Ch

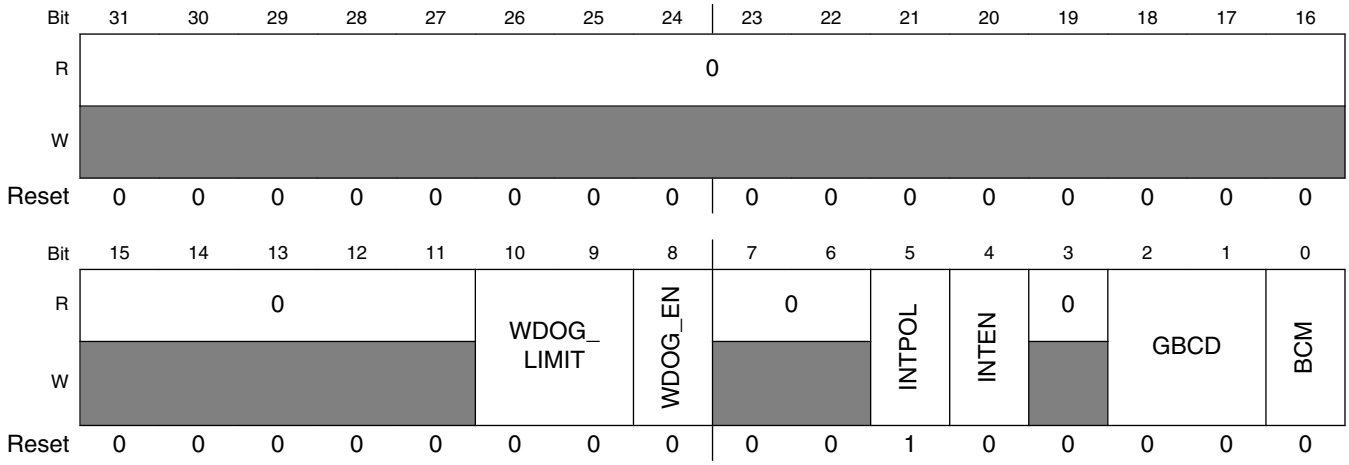
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																															WBCDD
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_CS*n*WCR2 field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 WBCDD	Write Burst Clock Divisor Decrement. If this bit is asserted and BCD value is 0 sync. write access will be preformed as if BCD value is 1.When this bit is negated or BCD value is not 0 this bit has no affect. This bit is cleared by hardware reset.

23.101.7 EIM Configuration Register (EIM_WCR)

Address: EIM_WCR is 63FD_8000h base + 90h offset = 63FD_8090h



EIM_WCR field descriptions

Field	Description
31–11 Reserved	This read-only field is reserved and always has the value zero. Reserved
10–9 WDOG_LIMIT	Memory Watch Dog (WDog) cycle limit. This bit field determines the number of BCLK cycles (ACLK cycles in dtack mode) before the wdog counter terminates the access and send an error response to the master. 00 128 BCLK cycles 01 256 BCLK cycles 10 512 BCLK cycles 11 1024 BCLK cycles
8 WDOG_EN	Memory WDog enable. This bit controls the operation of the wdog counter that terminates the EIM access. 0 Memory WDog is Disabled 1 Memory WDog is Enabled
7–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5 INTPOL	Interrupt Polarity. This bit field determines the polarity of the external device interrupt. 0 External interrupt polarity is active low 1 External interrupt polarity is active high
4 INTEN	Interrupt Enable. When this bit is set the External signal RDY_INT as active interrupt. When interrupt occurs, INT bit at the WCR will be set and t EIM_EXT_INT signal will be asserted correspondingly. This bit is cleared by a hardware reset. 0 External interrupt Disable 1 External interrupt Enable

Table continues on the next page...

EIM_WCR field descriptions (continued)

Field	Description
3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2–1 GBCD	General Burst Clock Divisor. When BCM bit is set, this bit field contains the value used to program the burst clock divisor for Continuos BCLK generation. The other BCD bit fields for each chip select are ignored. It is used to divide the internal AXI bus frequency. When BCM=0 GBCD bit field has no influence. GBCD is cleared by a hardware reset. 00 Divide EIM clock by 1 01 Divide EIM clock by 2 10 Divide EIM clock by 3 11 Divide EIM clock by 4
0 BCM	Burst Clock Mode. This bit selects the burst clock mode of operation. It is used for system debug mode. BCM is cleared by a hardware reset. **NOTE:** The BCLK frequency in this mode is according to GBCD bit field. **NOTE:** The BCLK phase is opposite to the EIM clock in this mode if GBCD is 0. **NOTE:** This bit should be used only in async. accesses. No sync access can be executed if this bit is set. **NOTE:** When this bit is set bcd field shouldn't be configured to 0. 0 The burst clock runs only when accessing a chip select range with the SWR/SRD bits set. When the burst clock is not running it remains in a logic 0 state. When the burst clock is running it is configured by the BCD and BCS bit fields in the chip select Configuration Register. 1 The burst clock runs whenever ACLK is active (independent of chip select configuration)

23.101.8 EIM IP Access Register (EIM_WIAR)

Address: EIM_WIAR is 63FD_8000h base + 94h offset = 63FD_8094h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											ACLK_EN	ERRST	INT	IPS_ACK	IPS_REQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

EIM_WIAR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4 ACLK_EN	ACLK enable. This bit gates the ACLK for the EIM except from FFs that get ipg_aclk_s. After reset ACLK is enabled. 0 ACLK is disabled 1 ACLK is enabled
3 ERRST	READY After Reset. This bit controls the initial ready/busy status for external devices on CS0 immediately after hardware reset. This is a sticky bit which is cleared once the RDY_INT signal is asserted by the external device. When ERRST = 1 the first fetch access from EIM to the external device located on CS0 will be pending until RDY_INT signal indicates that the external device is ready, then EIM will execute the access. **NOTE:** Reset value for ERRST is EIM_BOOT[4]. 0 RDY_INT After Reset Disable 1 RDY_INT After Reset Enable
2 INT	Interrupt. This bit indicates interrupt assertion by an external device according to RDY_INT signal. When polling this bit, INT=0 indicates interrupt not occurred and INT=1 indicates assertion of the external device interrupt. This bit is cleared by a hardware reset.
1 IPS_ACK	IPS ACK. The EIM is ready for ips access. There is no active AXI access and no new AXI access is accepted till this bit is cleared. This bit is cleared by the master after it completes the ips accesses. 0 Master cannot access ips. 1 Master can access ips.
0 IPS_REQ	IPS request. The Master requests to access one of the IPS registers.Dutring such access the EIM should not preform any AXI/memory accesses. The EIM finishes the AXI accesses that already starts and asserts the IPS_ACK bit. 0 No Master requests ips access 1 Master requests ips access

23.101.9 Error Address Register (EIM_EAR)

Address: EIM_EAR is 63FD_8000h base + 98h offset = 63FD_8098h

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

EIM_EAR field descriptions

Field	Description
31–0 Error-ADDR	Error Address. This bit field holds the AXI address of the last access that caused error. This register is read only register.

Chapter 24
Electrophoretic Display Controller (EPDC)

24.1 Overview

This chapter describes the detailed architecture of the EPDC. It provides a detailed description of the the block for digital design and software driver development.

The EPDC is a feature-rich, low power and high-performance direct-drive active matrix EPD controller. It is specifically designed to drive E•INK^{TM} EPD panels supporting a wide variety of TFT backplanes. The goal of the EPDC is to provide an efficient SoC integration of this functionality for e-paper applications, allowing a significant BOM cost saving over an external solution while reaching much higher levels of performance and lower power. The EPDC module is defined in the context of an optimized HW/SW partitioning and works in conjunction with the ePXP IP module to form a complete display processing solution.

The key features of the EPDC are as follows:

- TFT resolutions up to 4096 x 4096 pixels with 20 Hz refresh (programmable up to 8191 x 8191)
- TFT resolutions up to 2048 x 1536 pixels at 106 Hz refresh
- Industry standard bus interfaces (AMBA AXI and APB)
- Up to 5-bit pixel representation for up to 32 pixel levels
- Up to 16 concurrent updates with partial update support
- Automatic collision handling when used in conjunction with the i.MX driver
- Dual-scan TFT drive mode to support ultra-high resolution/refresh-rate displays
- Flexible direct-drive TFT interface supporting next generation source-drive, gate-driver and panel architectures, including LVDS, DDR and multi-level source drivers
- High-performance pixel pipeline architecture to guarantee refresh performance at high pixel-rates without the need for high internal clocking
- Ability to process multiple updates asynchronously to refresh/update operations with ability to intercept each frame scan will multiple update requests
- Performance tuning capabilities which can interface with SoC level memory arbitration mechanisms further guaranteeing frame refresh operation

- Decoupled clocking architecture allowing for independent and asynchronous clock sources for memory bus, peripheral bus and pixel-clock domains
- Full and partial update mode support
- Support for up to 256 waveform modes
- Low-power mode operation via architectural clock gating

24.2 Block Diagram and Overview

The top-level view of the EPDC is shown in Figure 24-1.

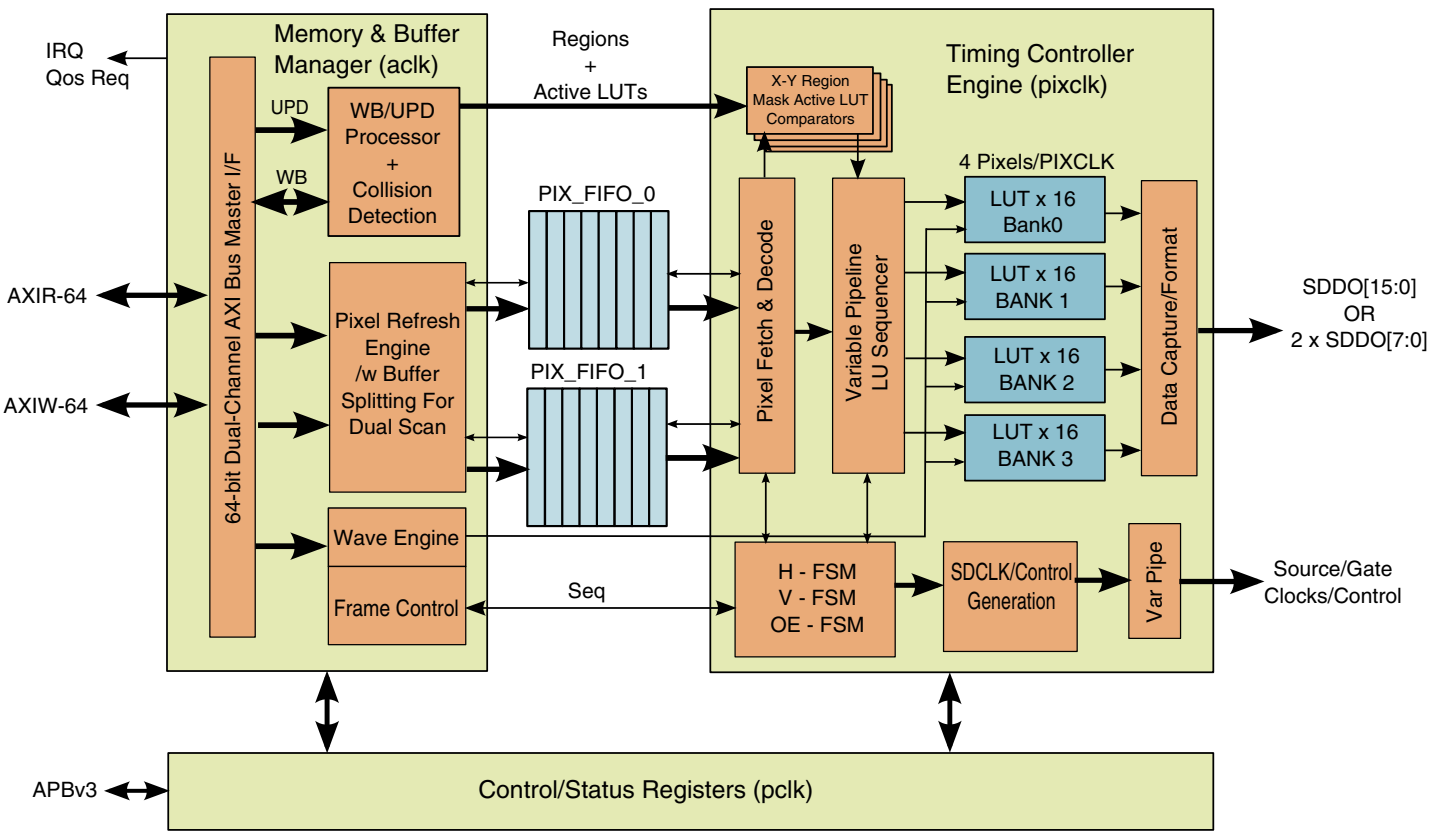


Figure 24-1. EPDC Top-Level Block Diagram

The EPD function can be separated into two major asynchronous processes. The first is the front-end portion which is responsible for processing display updates. The second is the function of driving waveforms to the TFT panel in order to update the screen contents (the refresh operation). The EPDC is comprised of two major submodules (MBM and TCE) to mirror this architectural split in the EPD algorithm. These submodules communicate through a number of control/sequencing signals and data (through the pixel FIFOs and LUT memories):

- Memory and Buffer Manager (MBM): This submodule is responsible for all memory related operations and LUT life cycle control, which involves the frame-count management for all updates. This module is clocked by both *aclk* and *pixclk*. It also encapsulates the APB register interface which is clocked by *apb_clk*.
- Timing Control Engine (TCE): This submodule is responsible for performing TFT scan frame refreshes. Its clocked by *pixclk*. It also houses the LUT memory system.

24.3 Programming Model

From an application perspective, the EPDC HW/SW solution aims to abstract much of the detail of driving an EPD display from the user space.

This is achieved through a blend of HW features and the kernel-level SW driver implementation of the EPD frame buffer. The scope of this chapter mostly pertains to EPDC HW but in order to correctly define the context of the programming model, certain assumptions about the driver architecture (including use of the ePXP IP) are described below.

24.3.1 Assumptions

The scope of the EPDC does not entail certain functions performed by the driver and application layer.

These features are as follows:

- Maintenance of update and collision lists. This includes, but is not limited to, responsibility for managing update order. Updates are processed in the order they are received by the EPDC. Actual start times of multiple updates can occur on the same frame scan. All pixels contained within an update will always begin on the same frame scan.
- Border control. This function is assumed to be performed by a GPIO functionality.
- EPD Panel Power Management. It is assumed that the driver and application layers will control the panel PMIC functions including relevant interfaces such as I^2C and 4-wire power sequencing signals.
- Reading and decompression of the waveform file. This is assumed to be unpacked, reformatted and written to system memory into a form that is easily accessed by the EPDC.
- Waveform mode selection. The ePXP provides a mechanism through its histogram analysis feature (which can be run as part of any frame/region processing operations) to allow the SW driver to characterize the region/buffer in terms of gray-map content. For example, the ePXP can determine if the update region only contains 2, 4,

8 or 16 gray levels, allowing the SW driver to choose the appropriate waveform update mode. Alternatively, the driver will also allow the application level SW to manually choose an update mode (which may be the case for ghosting artifacts clean-up, for example).

24.3.2 Register Space (Write/Set/Clear/Toggle)

The EPDC registers utilize four word address locations per 32-bit register.

For control registers (especially interrupts), this provides for a unique address for the following operations:

- Write (0x0). The base address of the register allows full writes to the 32-bit register.
- Set (0x4). This address allows a set operation on the register or its fields; writing a 1 to a register bit/field with the set address will set that bit. Writing a 0 to the set address has no effect.
- Clear (0x8). This address allows a clear operation on the register or its fields; writing a 1 to a register bit/field with the clear address will clear that bit or field.

NOTE

Interrupt status bits (such as those contained in EPDC_IRQ) must be cleared with the clear address only. Writing to the EPDC_IRQ register with a 0 will not clear the interrupt status/source. Writing a 0 to the clear address has no effect.

- Toggle (0xc). This address allows a toggle operation on the register or its fields; writing a 1 to a register bit/field with the toggle address will invert the state of that bit/field. This address is especially useful because it provides the ability to perform a read-modify-write operation with a single write operation. Writing a 0 to the toggle address has no effect.

24.3.3 Interrupts

24.3.3.1 Interrupt Sources

The EPDC contains 22 unique interrupt sources. Each interrupt source has an interrupt status bit captured in EPDC_IRQ.

When an interrupt event happens, the appropriate bit within the EPDC_IRQ register is set by the EPDC. Each of the bits in EPDC_IRQ is first AND'ed with its enable bit (in EPDC_IRQ_MASK), and then all the masked bits are logically OR'ed together to generate the final output.

24.3.3.2 Enabling/Masking Interrupts

Each of the EPDC interrupt sources can be individually enabled through the bits in the EPDC_IRQ_MASK register.

Even if an interrupt source is masked out (IRQ_EN bit set to 0), its status will still be available in the EPDC_IRQ register. The difference is that it does not result in the interrupt line being asserted.

24.3.3.3 Handling/Clearing Interrupts

When the software driver is entered as a result of an EPDC interrupt it should read the EPDC_IRQ interrupt status register.

Because multiple interrupts may have fired, in the case of a collision for example, both WB_CMPLT_IRQ and LUT_COL IRQ fields of EPDC_IRQ will be set. Once the interrupt has been handled, the driver should clear the interrupt source by writing to the CLEAR address (see Register Space (Write/Set/Clear/Toggle) for details).

It should be noted that for cases where a collision has occurred and the LUT_COL_IRQ bit is set, once the collision is handled and LUT_COL_IRQ is cleared this also clears the entire EPDC_STATUS_COL register.

24.3.4 Controller Setup

Before screen update operations can be performed the EPDC should be configured appropriately.

These rudimentary setup operations include:

- Waveform data must be present in memory (in a format that is consistent with the method in which the EPDC accesses it). Because the format of the waveform data is supplier proprietary information, it is not discussed in this chapter.
- Configuring the panel architecture parameters (source and gate-driver configuration)
- Configuring line and frame timing parameters
- Initializing the working buffer and display

24.3.4.1 Memory Requirements

At any point in time, the EPDC requires access to the un-packed waveform data, the update-region buffer, and its working buffer.

These are all expected to reside in system memory (typically external DRAM). As such, each has particular requirements:

- The Working Buffer (WB), which is pointed to by EPDC_WB_ADDR, must have EPDC_RES[HORIZONTAL] x EPDC_RES[VERTICAL] x 2 bytes allocated. This buffer is used by the EPDC to process updates and perform TFT refresh operations.
  - EPDC_RES[HORIZONTAL] mod 4 must equal 0 for memory allocation calculations because the EPDC constructs each line of the working buffer at a 64-bit boundary.
- The Update Buffer, which is pointed to by EPDC_UPD_ADDR (and can be re-defined with a different address for each update), must have the following allocation in memory:
  a. when stride feature not enabled
     - EPDC_UPD_SIZE[HEIGHT] x EPDC_UPD_SIZE[WIDTH] bytes
     - Note that EPDC_UPD_SIZE[WIDTH] mod 8 must equal to 0 (for memory allocation purposes). This means that for the address size allocation, the WIDTH field must be rounded up to the nearest number so that it can be divided by 8. This is because the EPDC reads each line of the update buffer at a 64-bit boundary (it should be noted that this is consistent with the ePXP IP output). The actual value programmed for the update can be at the pixel granularity.
- The waveform data set size is a function of mode-count, number of temperature tables and number of frames required for each temperature compensated mode.

In summary, the memory requirements are as follows (in bytes):

- WB-roundup4(EPDC_RES[HORIZONTAL]) x EPDC_RES[VERTICAL] x 2
- UPD-N x roundup8(EPDC_UPD_SIZE[WIDTH]) x EPDC_UPD_SIZE[HEIGHT]
  - N = the number of update requests currently being managed by the i.MX EPD driver (that is, the driver may maintain a list of updates in memory which it uses to feed the EPDC).

24.3.4.2 Panel Architecture Configuration

The EPDC is designed to directly drive EPD panels which typically expose the source and gate driver IC interfaces (these are typically abstracted in traditional displays such as TFT-LCD).

The source and gate driver ICs are responsible for driving the TFT matrix. There are variations in both the panel architectures and the underlying source and gate driver IC functionality.

All the key configuration parameters are defined in EDPC_RES, EPDC_FORMAT, EPDC_TCE_CTRL, EPDC_TCE_SDCFG and EPDC_TCE_GDCFG registers as follows.

- EPDC_RES:
  - HORIZONTAL: The panel horizontal resolution (in pixels). It must be noted that horizontal resolution must be an integer multiple of the source driver PIXELS_PER_SDCLK value. It also must be an integer multiple of the PIXELS_PER_CE value.
  - VERTICAL: The panel vertical resolution (in pixels). This can be any integer value.
- EPDC_FORMAT:
  - DEFAULT_TFT_PIXEL: This field should almost always be left at 0 x 00. This register is used as the value for the TFT pixel during a frame scan when a pixel location is not being updated. This condition is commonplace especially when using partial updates or updates which do not encompass the entire screen resolution. This value must always correspond to the state the source driver should see when no voltage change is needed. Incorrectly setting this value will damage the EPD material.
  - TFT_PIXEL_FORMAT: This enumerated field defines the width of the TFT pixel. The TFT pixel is defined as the per-pixel voltage control value. The most common pixel format is 2B (2 bits per pixel). Panels which support source-driver voltage modulation (multi-level voltage control) should use the 4B format. For source drivers that support 7 levels, the 4B format can be used and the unused SDDO output pins should be left floating at the board level connection (specifically SDDO3, 7, 11, 15). It should be noted that the 2BV and 4BV are variations which require the waveform data to supply a 2-bit VCOM value for each pixel.
- EPDC_TCE_CTRL:
  - SDDO_WIDTH: This selects 8- or 16-bit SDDO operation using enumerations 8-bit and 16-bit. This field is used to describe useful data, that is, for panels which support LVDS, ipp_epdc_sddo[15:8] would be used to drive differential data. In these modes, SDDO_WIDTH would be set to 8-bit.

- VCOM_MODE, VCOM_VAL: This high-level mode bit allows the EPDC to either use the VCOM value supplied in the waveform or a software programmable value defined in the VCOM_VAL field, for panels which support VCOM modulation.
- DDR_MODE: This mode-bit should be set for panels which expect source-driver data to be available on both edges of the SDCLK. This is common place for panels that support differential signaling, but the feature is not limited to LVDS panels. For example, the EPDC supports DDR modes which make full use of SDDO[15:0]. In these modes, DDR_MODE = 1 and LVDS_MODE = 0.
- LVDS_MODE: This mode always requires DDR_MODE to be set. When this mode bit is set, differential signaling is enabled such that SDDO[15:8] always drives the inverse of the pixel data which is presented on SDDO[7:0]. Because LVDS source-drivers use 2 pins per data-bit, they are typically configured to work in DDR mode. Differential signaling is also supported on the SDCE pins (as an option selected by LVDS_MODE_CE). In this mode SDCE[9:5] are used to drive the inverse differential signals for SDCE[4:0].
- LVDS_MODE_CE: When LVDS_MODE is set, this bit can also be set to allow SDCE[9:5] to act as differential pairs for each SDCE[4:0].
- DUAL_SCAN: Setting this field enables the dual scan function. When this field is set, all other parameters (except resolution) should be programmed in reference to each half of the panel. Because DUAL_SCAN modes require 2x the pixel-generation capability, there are limitations to the available TFT modes supported in this configuration (see Table 24-1 for details).
- SCAN_DIR0, SCAN_DIR1: These fields determine the vertical scan direction of the panel.
  - SCAN_DIR0: Determines the vertical scan direction of the panel when DUAL_SCAN = 0. When DUAL_SCAN = 1, this field determines the vertical scan direction of the top-half of the panel (0 means scan down and 1 means scan up).
  - SCAN_DIR1 only has meaning when DUAL_SCAN = 1. It defines the vertical scan direction of the bottom-half of the panel.
- PIXELS_PER_SDCLK: This is a key configuration and also timing parameter. Each source-driver latches a number of TFT pixels per shift-clock period (SDCLK). This register defines how many pixels are driven per SDCLK period. It must be noted that when DDR_MODE = 1, pixels are driven on both edges of the clock, so this value must be adjusted accordingly. See Table 24-1 for the required values per mode.

- EPDC_TCE_SDCFG:
  - SDCLK_HOLD: This holds the SDCLK low during the LINE_BEGIN time. The purpose of this field is to allow the user to set a LINE_BEGIN time which is not an integer multiple of SDCLKs (for fine tuning the line-time). In order to avoid

shortened pulses on SDCLK, this can be used to hold SDCLK low during the LINE_BEGIN time.
    - SDSHR: This defines the value of the SDSHR output signal which determines the source-driver output order mapping.
    - NUM_CE: This defines the total number of source-driver chip-enables (must be 1-0). It should be noted that this number does not always correspond to the number of source drivers in the panel. Many panels, only require a single chip-enable signal which is active during the entire line-data time. This signal is often called SPH in such cases.
    - PIXELS_PER_CE: This register defines the span of each chip-enable expressed in pixels. Note that for panels which utilize a single global CE, this value should be set to the horizontal resolution of the panel.
    - SDDO_REFORMAT: This enumerated field allows for TFT-pixel level re-ordering within SDDO. For most panels it is recommended to set SDDO_REFORMAT to the REVERSE_PIXELS enumeration.
    - SDDO_INVERT: When this field is set, the values on SDDO are inverted relative to the values extracted from the waveform LUT operations.
- EPDC_TCE_GDCFG:
    - GDRL: This defines the value of the GDRL output signal which determines the gate-driver pulse shift direction.
    - GDOE_MODE: This field selects between two possible methods for driving the GDOE signal. When GDOE_MODE = 0, the GDOE output is always driven during the frame-scan time except during FRAME_SYNC when it is driven low. When GDOE_MODE = 1, the GDOE signal mimics the GDCLK signal but is only active during the frame-data time (FD). For most panels this is recommended to be set at 0.
    - GDSP_MODE: This field selects between two possible methods for driving the GDSP signal. When GDSP_MODE = 0, the GDSP signal is driven on the first line-clock time of the FRAME_BEGIN time. The signal is driven for one GDCLK period (same as line-time), and can be shifted using the GDSP_OFFSET control. When GDSP_MODE = 1, GDSP is active during the entire FRAME_SYNC time, and in this mode GDSP_OFFSET has not effect.

24.3.4.3 TFT Panel Timing Configuration

The EPDC provides a simple primary set of registers to define TFT line and frame (refresh) timing.

In addition to these, there are also low-level timing and polarity registers to provide additional flexibility in dealing with future TFT architectures.

The following figure shows how the horizontal and vertical timing is defined in terms of the various timing parameters. When configuring the TFT timing, the following goals should be met:

- Arrive at a refresh rate that matches the waveform requirement (for example, 50 Hz for 50 Hz waveform). The refresh rate is defined as the time between each Frame-Sync event which can be expressed as a multiple of the line timing and the total number of vertical lines.
- Meet the various timing requirements of the gate and source driver clock and control signals (the blanking period provide an infrastructure for controlling these).
- Stay below the maximum source-driver clock (usually referred to as CL) frequency.
- Arrive at a source-driver clock frequency which in turn defines the exact pixclk frequency. In most cases, E-INK™ shall provide the expected SDCLK frequency.
- Meet the line-timing (LT) requirements of the E-INK panels and their associated waveforms.

For proper waveform performance, it is critical to match the refresh frequency to that required by the waveform and associated panel. Deviation from this frequency results in short term inaccuracy of color and in the long term, complete loss of coherency between the EPDC's internal representation of color and the physical representation on the screen.

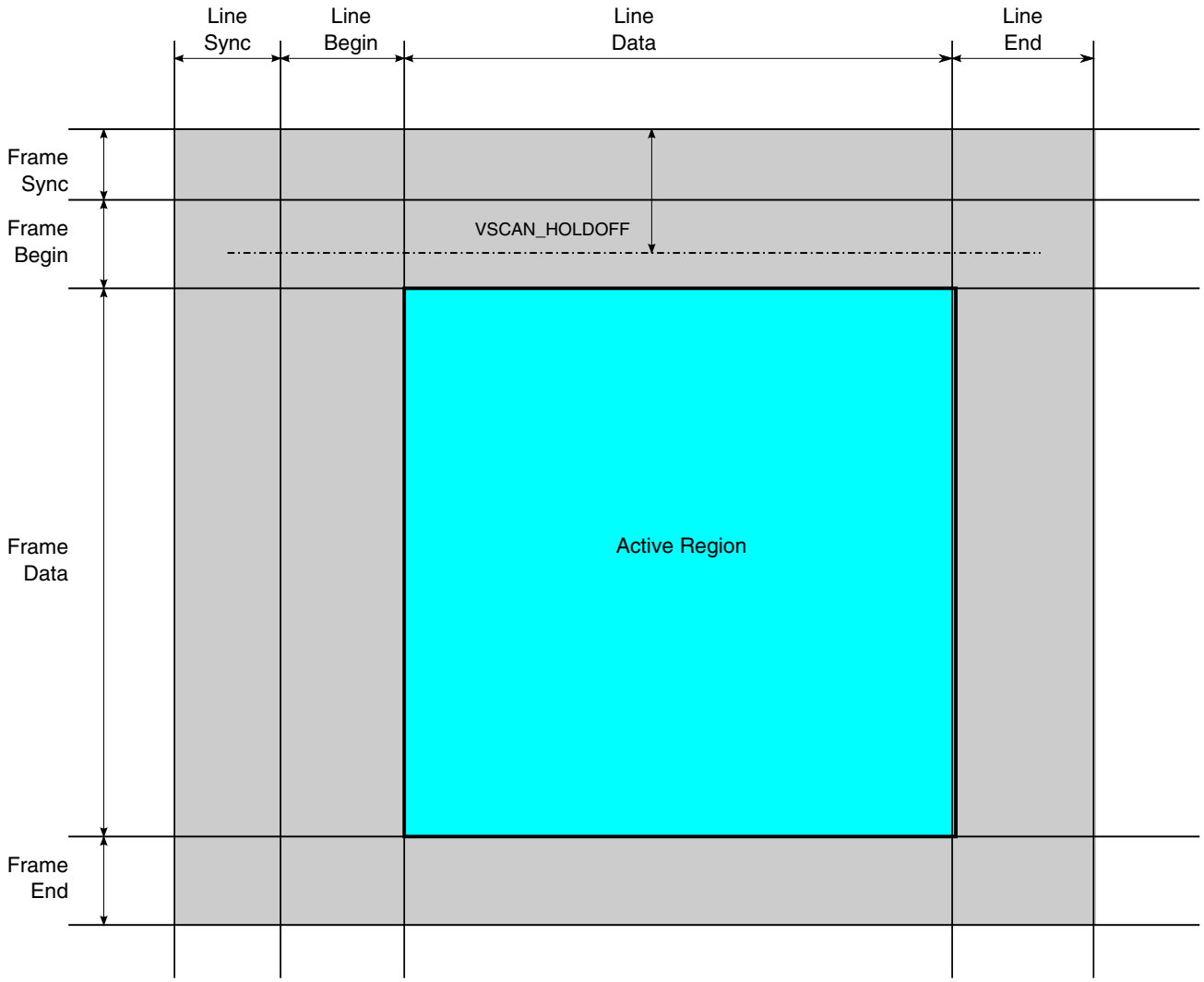


Figure 24-2. Frame Timing

The following equations define the timing of the panel scan frame (where LT = line-timing and FT = frame-timing):

$$LT = T_{PIXCLK}\,(LS + LB + (HRES \times \left(\frac{Ratio}{PIXELSPERSDCLK}\right) + LE)$$

$$FT = LT \times (FS + FB + VRES + LE)$$

Ratio is defined between the internal PIXCLK and the external source driver clock SDCLK. The ratio is automatically determined by the EPDC according to Table 24-1. A simple method to determine the ratio is as follows:

- When DDR_MODE = 1, RATIO = 4
- When DDR_MODE = 0
  - If PIXELS_PER_SDCLK = 8, RATIO = 4
  - Else, RATIO = 2

Typically it is recommended to seed these equations with an estimated (or provided) SDCLK frequency and adjust the various timings until the desired refresh rate is attained.

The following figure shows line timing for cases, where DDR_MODE = 0 (that is, those cases where data is only sampled by the source driver on the rising edge of SDCLK). Depending on the mode (see Table 24-1), the ratio is a function of TFT pixel width and PIXELS_PER_SDCLK, so the timings are shown generically, that is, for this diagram RATIO can either be 2 or 4.

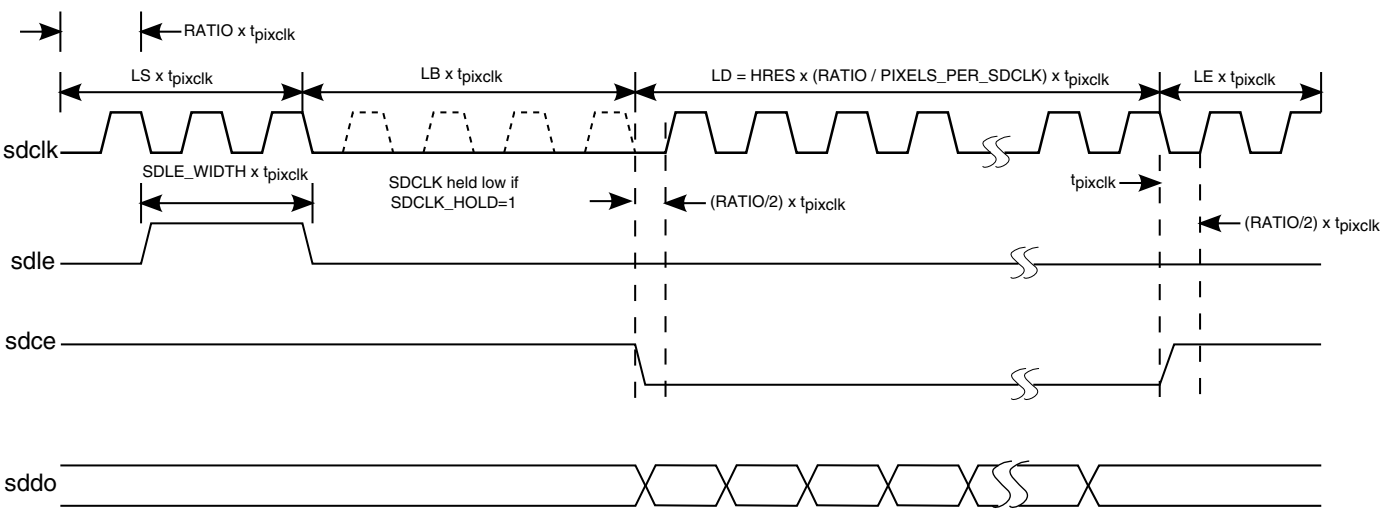


Figure 24-3. Line Timing (DDR_MODE = 0)

As per Table 24-1, when DDR_MODE=1 (for example, for source drivers that require data to be driven on each edge of SDCLK), the RATIO is always set to 4. The timing for such cases is shown in the figure below.

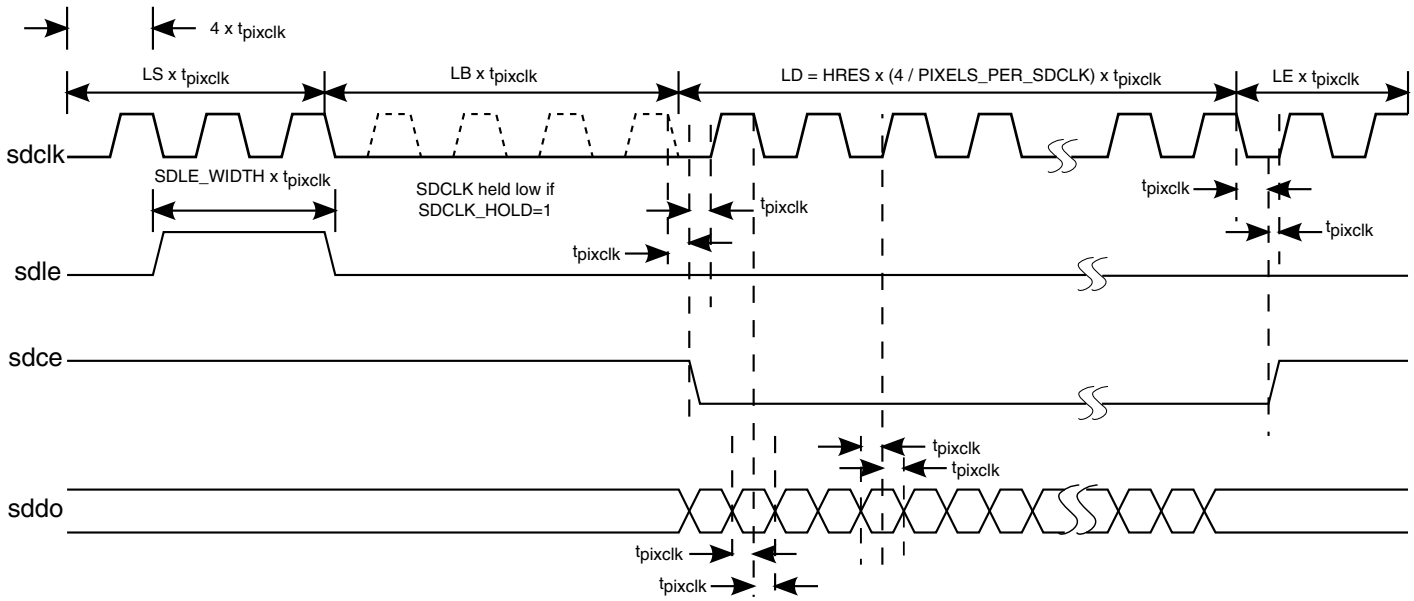


Figure 24-4. Line Timing (DDR_MODE = 1) - RATIO = 4

Vertical timing or frame timing (FT) is always expressed in terms of line timing (LT). This is shown in the following figure, followed by a detailed view of the GDCLK timing (which has programmable duty cycle to allow programming of gate-on/gate-off time) in Figure 24-6. The following important points should be noted in regard to gate-driver timing:

- All gate driver timing signals are independently referenced to the LINE_SYNC leading edge (this includes GDCLK_OFFSET, GDSP_OFFSET, and GDOE_OFFSET).
- GDSP_OFFSET is only supported for GDSP_MODE = 1
- For GDOE_MODE = 1, if GDOE is expected to be delayed relative to GDCLK, the GDOE_OFFSET value should be set at a value appropriately greater than GDCLK_OFFSET.
- All offset settings are programmed in terms of PIXCLK cycles.
- By default, if all offset values are set to 0, the gate-driver signals are always one PIXCLK cycle delayed from the LINE_SYNC leading edge.
- The LINE_SYNC leading edge does not depend on the LINE_SYNC_WIDTH value. In cases, where LINE_SYNC_WIDTH is set to some value that is less than LINE_SYNC, a virtual leading edge which determines the line start time still exists.

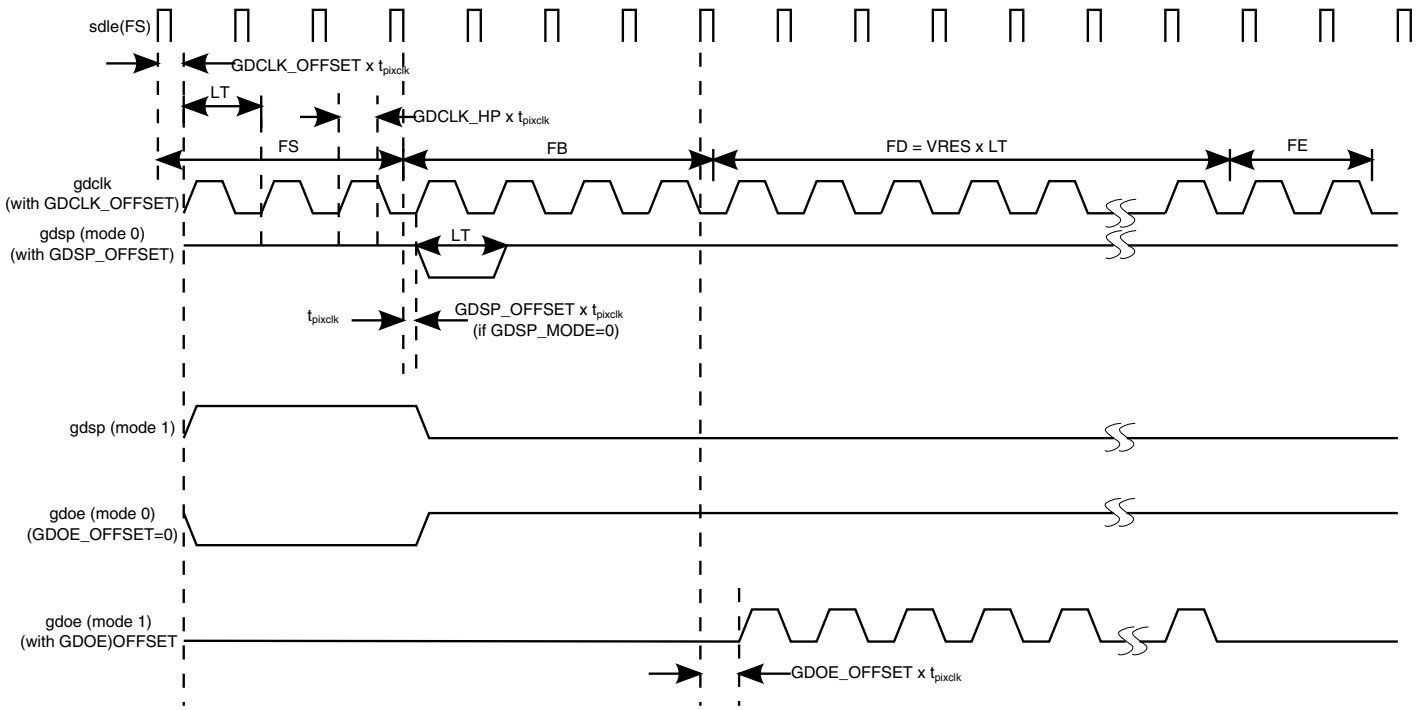


Figure 24-5. Frame Timing (Vertical)

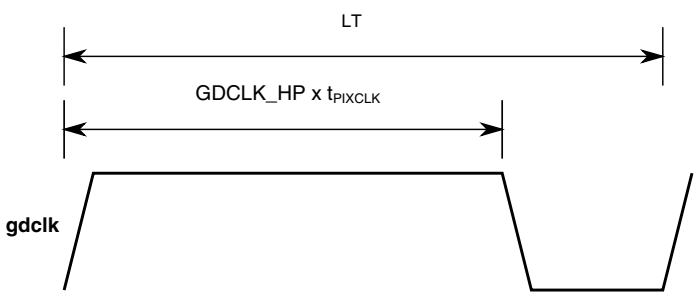


Figure 24-6. GDCLK Duty Cycle

An example calculation based on the 800 x 600 E-INK 6" panel with 50 Hz waveforms is shown be:

- PIXELS_PER_SDCLK = 4
- Ratio = 2 (2 bpp, 8-bit single-ended, SDR per Table 24-1)
- tPIXCLK = 17.64 MHz (assuming available chip clock frequency)
- LS = 20 x tPIXCLK
- LB = 8 x tPIXCLK
- LD = 800 x (2/4) x tPIXCLK = 400 x tPIXCLK
- LE = 142 x tPIXCLK
- LT = (20 + 8 + 400 + 142) x tPIXCLK = 32.31293 uS

It is seen that LINE_SYNC_WIDTH does not affect line-timing. It is recommended set the same value as LINE_SYNC, unless LB and LE are short and it is used to shorten the width of SDLE whilst still maintaining the same line-timing.

Note that it is important to meet the target line time of the panel which was 32.312 uS. Now, the frame-rate can be calculated, given the vertical timings:

- FS = 4 x LT
- FB = 4 x LT
- FD = 600 x LT
- FE = 10 x LT
- FT = (4 + 4 + 600 + 10) x LT = 50.07665 Hz

In addition to this, E-INK panels require a particular duty cycle for the GDCLK signal which is specified as gate-on-time and gate-off-time. The sum of gate-off and gate-on should equal LT and the value of gate-on can be programmed through HW_EPDC_TCE_TIMING2[GDCLK_HP], that is, GDCLK High-Pulse time.

24.3.4.4 Source-Driver and Pixel-Clock Configuration

The EPDC supports a number of source-driver architectures. Each architecture forces a particular shift-clock pixel rate.

In addition to this, the selection of the architecture requires a particular clock ratio between the internal pixel clock (pixclk) and external shift clock (ipp_epdc_sdclk). The table below outlines the various configurations and clock ratios. It also lists the relevant register settings for this configuration.

Table 24-1. Interface Modes

High-Level Mode		HW_EPDC_CTRL [DUAL_SCAN]	ipp_epdc_sddo pins used	HW_EPDC_FORMAT [TFT_PIXEL_FORMAT]	HW_EPDC_TCE_CTRL [PIXELS_PER_SDCLK]	HW_EPDC_TCE_CTRL [LVDS_MODE]	HW_EPDC_TCE_CTRL [DDR_MODE]	HW_EPDC_TCE_CTRL [SDDO_WIDTH]	Required PIXCLK (RATIO)
Single Scan	2bpp, 8-bit single-ended, SDR	0	[7:0]	2B[V]	FOUR	0	0	8BIT	SDCLK x 2
	2bpp, 16-bit single-ended, SDR	0	[15:0]	2B[V]	EIGHT	0	0	16BIT	SDCLK x 4
	2bpp, 8-bit, DDR (LVDS option)	0	[7:0],[15:8]	2B[V]	EIGHT	1|0	1	8BIT	SDCLK x 4
	4bpp, 8-bit single-ended, SDR	0	[7:0]	4B[V]	TWO	0	0	8BIT	SDCLK x 2
	4bpp, 16-bit, single-ended SDR	0	[15:0]	4B[V]	FOUR	0	0	16BIT	SDCLK x 2
	4bpp, 8-bit, DDR (LVDS option)	0	[7:0],[15:8]	4B[V]	FOUR	1|0	1	8BIT	SDCLK x 4
	4bpp, 16-bit, single-ended, DDR	0	[15:0]	4B[V]	EIGHT	0	1	16BIT	SDCLK x 4

Table continues on the next page...

Table 24-1. Interface Modes (continued)

High-Level Mode		HW_EPDC_CTRL [DUAL_SCAN]	ipp_epdc_sddo pins used	HW_EPDC_FORMAT [TFT_PIXEL_FORMAT]	HW_EPDC_TCE_CTRL [PIXELS_PER_SDCLK]	HW_EPDC_TCE_CTRL [LVDS_MODE]	HW_EPDC_TCE_CTRL [DDR_MODE]	HW_EPDC_TCE_CTRL [SDDO_WIDTH]	Required PIXCLK (RATIO)
Dual Scan	2bpp, 8-bit single-ended, SDR	1	[7:0],[15:8]	2B[V]	FOUR	0	0	8BIT	SDCLK x 2
	2bpp, 8-bit, DDR	1	[7:0],[15:8]	2B[V]	EIGHT	0	1	8BIT	SDCLK x 4
	4bpp, 8-bit single-ended, SDR	1	[7:0],[15:8]	4B[V]	TWO	0	0	8BIT	SDCLK x 2
	4bpp, 8-bit, DDR	1	[7:0],[15:8]	4B[V]	FOUR	0	1	8BIT	SDCLK x 4

24.3.4.5 Initializing the Display

Because of the nature of the EPD technology, a typical scenario would involve the EPDC's working buffer (WB) being maintained in system memory. In such power states (where memory is maintained), it is not required to initialize the display (even if the EPD panel had been powered off for example).

In low-power modes where system memory is not maintained, there are two possible methods to initialize the display.

The first method includes initialization from an unknown state and the second mode simply requires software to load the last state of the WB into memory and configure the EPDC to point to it.

24.3.4.5.1 Reset/Clocks and Buffer Preparation

In cases where the user wants to initialize the screen to a new known state, the following sequence should be followed. Assume that all relevant display power supplies are active.

First, the user must complete a soft-reset sequence which also includes enabling the main EPDC block-level clock-gate. This sequence is described in the example code below. It should be noted that setting SFTRST enables the CLKGATE. In order to correctly cycle reset values through pipeline registers, the CLKGATE bit stays asserted for some finite amount of time before it sets. After this time, it is safe to clear both the SFTRST and CLKGATE. The example below shows a typical use of these registers.

```
EPDC_CTRL_SET(BM_EPDC_CTRL_SFTRST);
while (!EPDC_CTRL.B.CLKGATE);
EPDC_CTRL_CLR(BM_EPDC_CTRL_SFTRST | BM_EPDC_CTRL_CLKGATE);
while (EPDC_CTRL_RD() & (BM_EPDC_CTRL_SFTRST | BM_EPDC_CTRL_CLKGATE));
```

Before any display update is performed, the waveform address and working buffer (WB) pointers must be set (both must be aligned to a 64-bit double long word address):

- EPDC_WVADDR-Must point to the base address of the waveform data (managed by the i.MX driver)
- EPDC_WB_ADDR-An area of memory must be assigned for the EPDC's working buffer (WB). Once assigned, this memory space should be reserved purely for the use of the EPDC. The requirements for the memory allocation are described in Memory Requirements.

After this, the various panel-configuration parameters must be set including resolutions, source and gate-driver formats, TFT and buffer pixel formats and so on.

24.3.4.5.2 Performing an Initialization Display Update

Once all the panel timing and format parameters are defined a command sequence must be sent to the EPDC with the following properties:

- Update size must be full screen resolution
- HW_EPDC_UPD_CTRL[UPDATE_MODE] must be set to FULL
- HW_EPDC_UPD_CTRL[WAVEFORM_MODE] must be set to the INIT waveform (typically 0x00)
- HW_EPDC_UPD_CTRL[USE_FIXED] must be set
- HW_EPDC_UPD_FIXED[FIXNP_EN], FIXCP_EN must be set to 1, and FIXNP and FIXCP must be set to 0xFF

The use of HW_EPDC_UPD_FIXED allows the EPDC working buffer to be primed to a state that matches the result of the initialization waveform.

If the application loads the previous WB of the EPDC, these steps are not necessary.

24.3.5 Dual-Scan Configuration

The EPDC supports ultra-high resolution/refresh rate EPD panels.

At certain resolutions/refresh-rates there exist electrical limitations of the TFT devices and TFT driver ICs such as source drivers. In order to still support driving such displays, the EPDC supports the method of the dual-scan driver method.

The specific method involves driving a TFT configuration where there are two sets of source drivers. All source driver control signals are expected to be shared across all source drivers (except the SDCE signals unless a unified CE is presented by the panel).

In this mode, ipp_epdc_sddo[7:0] is reserved to drive the upper-half of the display and ipp_epdc_sddo[15:8] is reserved to drive the lower-half of the panel. The update and working buffers do not have to be aware of the dual-scan operation since the MBM knows how to access the WB for refresh operations based on the settings of the EPDC_TCE_CTRL[SCAN_DIR_0,1]. The EPDC support four scan configurations for dual scan which are shown in the following figures. Each figure shows an example configuration (that is, the EPDC is not limited to supporting only 2 source drivers per side).

It should be noted that the dual-scan drive method allows the SDCLK frequency to be halved relative to a single-scan method for the same resolution and refresh rate. In addition to this, each TFT is only charged half as frequently (since for any given SDCLK cycle, twice as many pixels are being driven per frame scan).

Because all 16 SDDO signals must be utilized for dual-scan, there is no support for the LVDS_MODE. The supported source-driver modes for dual scan are shown in Table 24-1. From this table, it can be seen that for the 2bpp DDR mode, the EPDC is capable of generating a total of 16 pixels per SDCLK cycle.

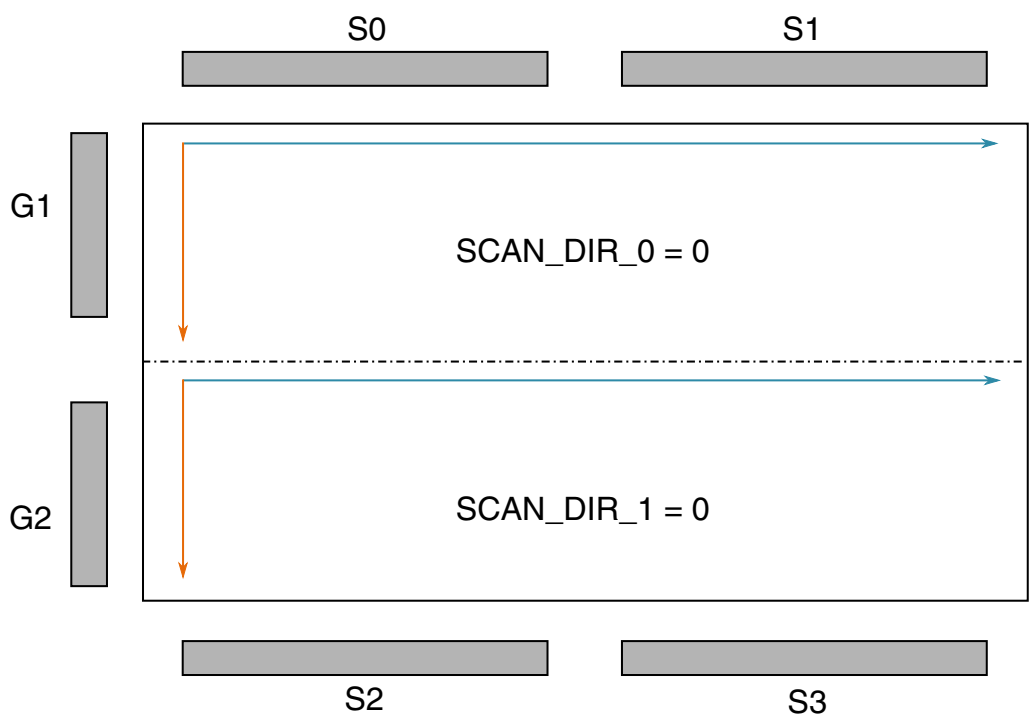


Figure 24-7. Dual-Scan Method (Scan = 00)

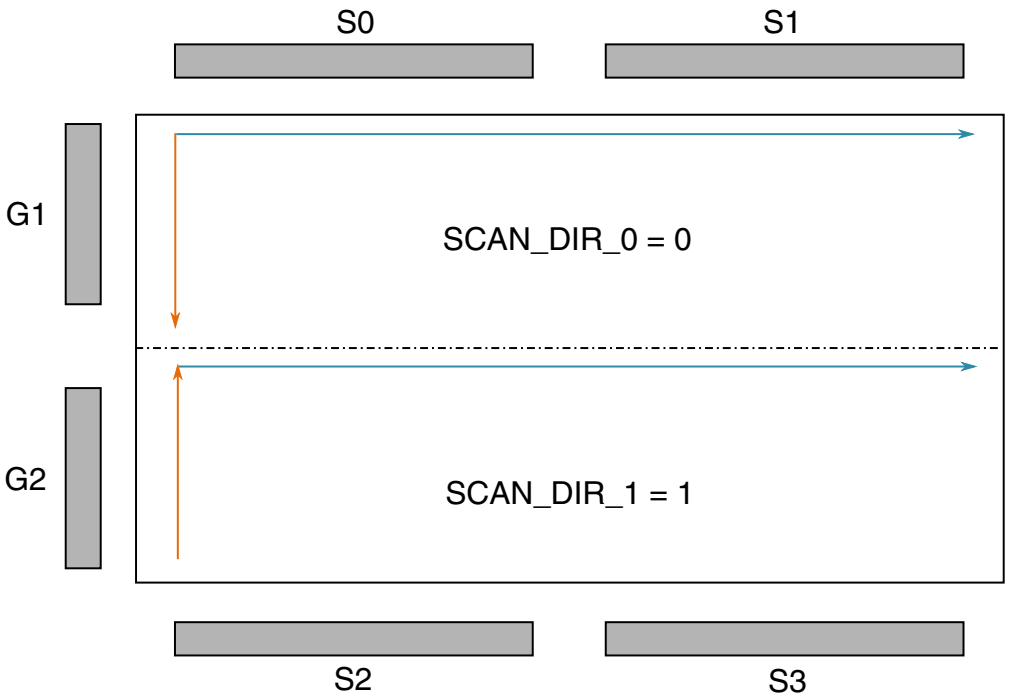


Figure 24-8. Dual-Scan Method (Scan = 01)

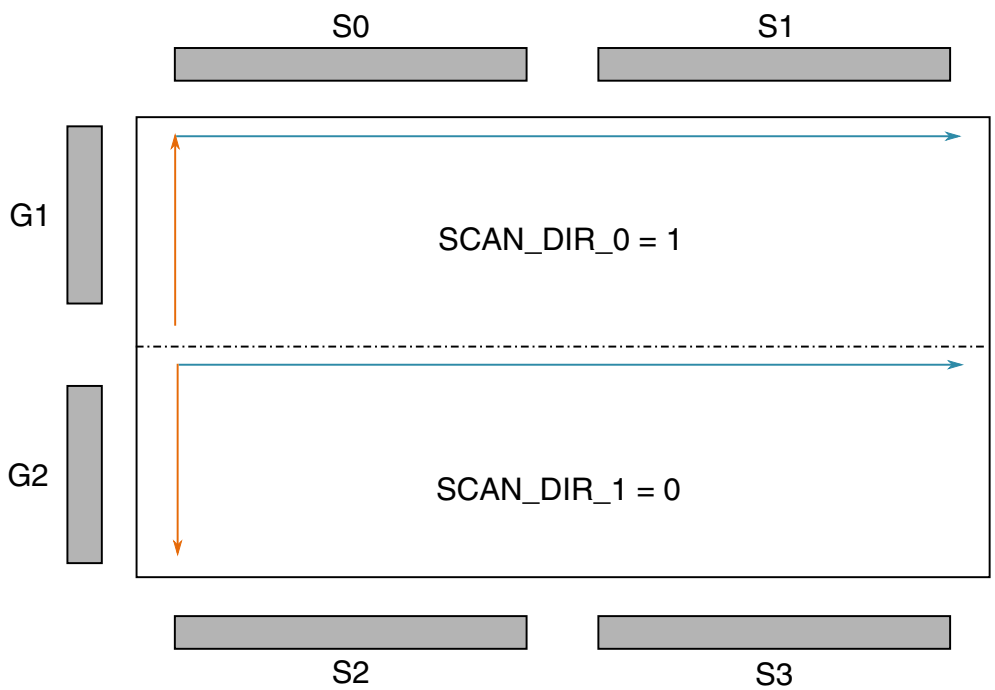


Figure 24-9. Dual-Scan Method (Scan = 10)

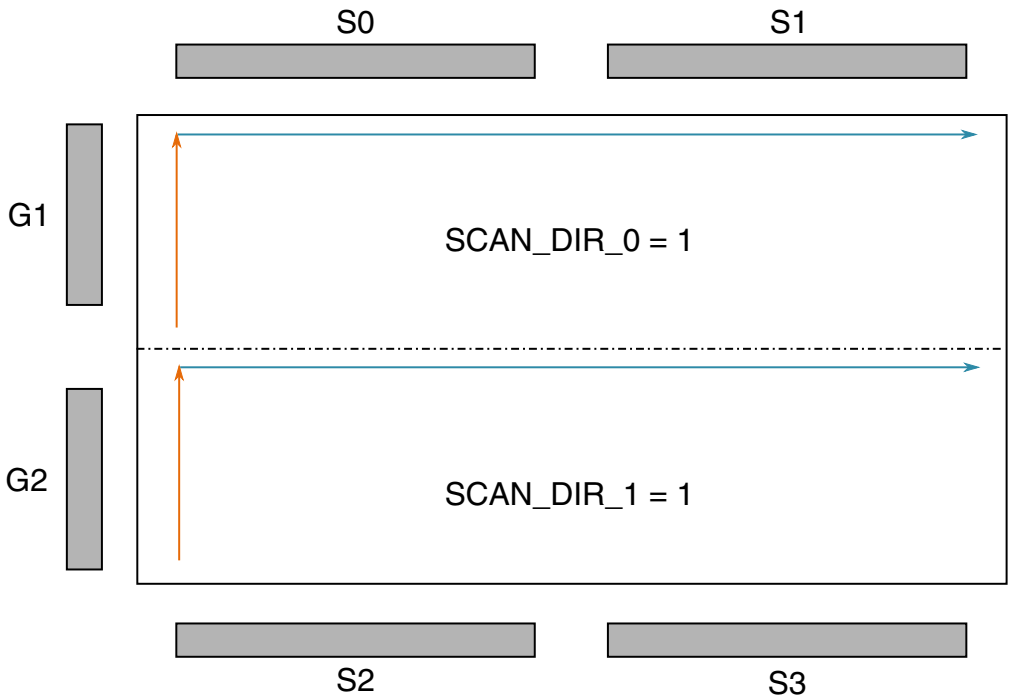


Figure 24-10. Dual-Scan Method (Scan = 11)

24.3.6 Display Update Programming

The EPDC is designed to communicate with the kernel driver through the interrupts and status registers.

The driver entry is always assumed to occur as a result of an interrupt.

24.3.6.1 Initiating a Display Update

The typical flow for in performing a display update involves the following:

- EPDC_STATUS[WB_BUSY] must be 0. The EPDC cannot process new updates if it is currently processing an update.
- EPDC_STATUS[LUTS_BUSY] must be 0. The EPDC cannot accept new updates if all LUT resources are currently active.

Assuming both conditions above are met, an update request is programmed in the following manner (it is important to note that the last step must be writing to EPDC_UPD_CTRL which initiates the display update).

The following steps can be performed before WB_BUSY and LUTS_BUSY are clear:

- At this time it is assumed that the correct temperature is selected and written to EPDC_TEMP.
- EPDC_UPD_ADDR must be set to the 8-bit buffer for the update. This buffer must be aligned to a 64-bit word address. In addition, each line address must begin at a 64-bit word address. Both criteria can be met buy utilizing the ePXP pixel processing engine which always aligns lines at a 64-bit raster (see Memory Requirements for details).
- EPDC_UPD_CORD-defines the insertion co-ordinate into the panel resolution.
- EPDC_UPD_SIZE-defines the dimension of the update (pixels).
- EPDC_UPD_FIXED-used for panel initialization or rectangular fill operations.

Writing the EPDC_UPD_CTRL register initiates a display update:

- USE_FIXED-This field is set when EPDC_UPD_FIXED is used.
- LUT_SEL-The driver should read EPDC_STATUS_NEXLUT[NEXT_LUT] (qualified by NEXT_LUT_VALID) to select an available LUT to assign this update.
- WAVEFORM_MODE-Selects the E-INK waveform mode (for example, INIT, DU, GC16 and so on). The number corresponds to the waveform number in the waveform specification document.
- UPDATE_MODE-This selects one of two enumerated updated modes.
  - FULL-In this mode, all pixels defined in the update rectangle have the waveform applied.
  - PARTIAL-In this mode, the EPDC shall only apply the waveform to pixels which are changing, otherwise the EPDC_FORMAT[DEFAULT_TFT_PIXEL] is applied to the pixels which are not changing.

24.3.6.2 Update Processing and Collisions

After the update is initiated (by writing EPDC_UPD_CTRL), the EPDC performs update-buffer processing. This involves processing this update region into it's working buffer (WB).

It is also during this time that the EPDC performs collision detection. At the end of the WB processing, the EPDC asserts the WB_CMPLT_IRQ interrupt. If a collision is detected, the LUT_COL_IRQ interrupt status bit will also be asserted.

The EPDC performs image updates to the EPD by applying waveforms to individual pixels. Waveforms are applied to groups of pixels (ranging from a single pixel to the entire display). Each individual update request is bound to a rectangle. Each pixel within this rectangle shall have a waveform applied to it using the waveform and update mode specified in EPDC_UPD_CTRL. After this process is initiated, a waveform take a finite

amount of time to complete (usually specified within the specification of the waveform mode). For proper display operation and optimum performance, a waveform must not be interrupted once it has begun

A collision is defined simply as the interruption of the application of waveform to a pixel or group of pixels. As such, it must be avoided. Because the waveforms take a finite amount of time to complete, the EPDC provides mechanism to automatically manage collisions. This allows the application to perform multiple updates resulting in a more rich user-experience without having to be cognizant of the panel and waveform characteristics.

Depending on the update mode (FULL or PARTIAL) a collision is defined as follows:

- For a full-update mode, a collision is defined as the temporal and spatial union of rectangles pertaining to unique update requests which are active. An active update is one which is still in the process of being updated (that is, a waveform is being applied).
- For a partial-update mode, a collision is defined as the temporal, spatial and differential union of pixels pertaining to unique update requests which are active. The difference between a full-updated and a partial-updated collision is that in the latter case, the collision state occurs only if the next-pixel states differ between the updates. Thus, two partial-update rectangles may intersect, but if all of the pixels in the intersecting update regions are being updated to the same gray-state, the original update and associated LUT/waveform are not disturbed and the intersection does not constitute a collision.
- It should be noted that collisions may be a one-to-many effect. In the case of partial updates, there may be a spatial and temporal union of rectangles with no differential pixel-level union. A subsequent update however might form such a pixel-level differential union to all active updates. Since each update pertains to an active LUT, these many collided updates is refered as victim LUTs in the subsequent sections. The update/LUT which caused the collision is referred to as the aggressor LUT.

An example of a partial-update mode collision is shown in the following figure. Each update encompasses the outer rectangle which meets the criteria of the spatial union. The red areas show the pixels which are currently transitioning to gray (from update 1) but need to transition to white (in update 2), these pixels meet the criteria of the differential union and lastly in the temporal domain, the period of time where LUT0 and LUT1 overlap is the temporal union. The pixels which did not change (grey and green areas in the LUT assignment box) are not part of the collision area. In the case of the gray overlapping pixels, LUT0 can continue to update. In the case of the white pixels (which were previously in-active) a new LUT (if available can be assigned) and forms the green region. Note that a subsequent update could potentially collide with the green pixels in update 2.

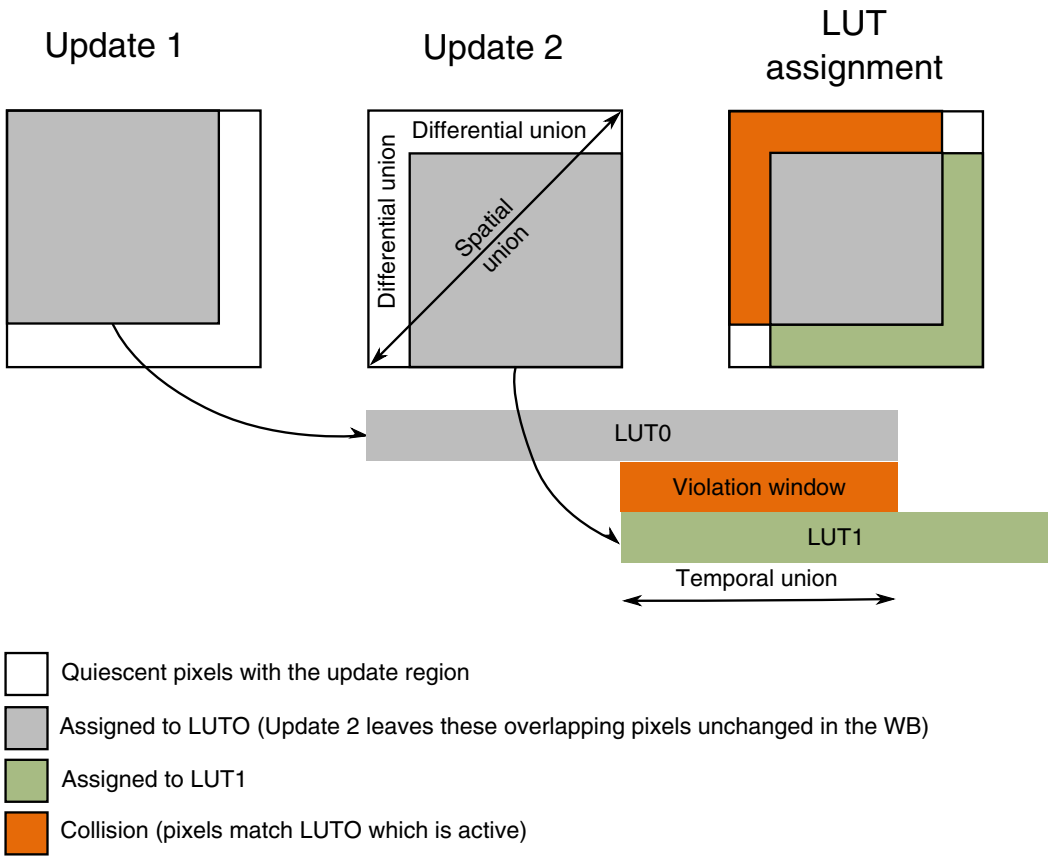


Figure 24-11. Partial Update Collision Handling

NOTE

For FULL update modes, a collision occurs at any intersection of active rectangles/LUTs because in FULL update mode, all pixels within the update rectangle have the waveform applied (even if pixel values are not changing).

It can be seen that when collision is detected (during the WB update process), the EPDC shall "block" the collided pixels being updated, and any pixels which are not collided will be assigned to the requested LUT process. When a collision occurs, the EPDC shall raise the LUT_COL_IRQ interrupt status bit (this will always happen in conjunction with WB_CMPLT_IRQ).

Because the driver is always entered via an interrupt, when the EPDC interrupt fires, it should not only check for WB_CMPLT_IRQ but also for the LUT_COL_IRQ status bit within the EPDC_IRQ register. If LUT_COL_IRQ is set, this means that a collision has occurred. If this is the case the driver should check the value of the EPDC_STATUS_COL register. This register contains one bit for each LUT that has been collided with, that is, it holds the vector of the "victim" LUTs as a result of the last update LUT. Since the driver knows the value of the LUT of the last update, it can store this update LUT value along with this associated "victim" LUT vector in a collision list.

Because the EPDC can drive up to 16 concurrent updates (which can be overlapping when using PARTIAL mode updates), a unique interrupt status bit is provided per LUT so that when an update completes (physically completes the waveform), the SW driver can know which LUT completed. In addition to the LUT completion interrupts (which are mapped into the EPDC_IRQ[LUTn_CMPLT_IRQ] interrupt status bits, the current state of each LUT can also be read from the EPDC_STATUS_LUTS status register.

The i.MX driver can use the LUT status interrupts in conjunction with EPDC_STATUS_LUTS to perform regular checks on the completion of "victim" LUTs against the vectors stored for each colliding LUT in the collision list. The driver may also choose to serialize updates such that it might wait until all victim LUTs have completed before re-issuing the colliding update.

The figure below shows a more extended sequence of a collision. In this case there are a series of updates coming from the application layer. These are A,B,C and D. Updates A, B, and C are set without any collision. When update D is requested a collision is detected. Using the interrupt status collision registers, the SW-driver stores D in a "collision-list". For that list entry it also store B which is the "victim LUT". It uses this information to check when a LUT completes to see if the victim LUTs for any update in the collision list have completed. If they have completed (through a LUT completion interrupt), the SW-driver then sends this update (D in this case) again to the EPDC. It should be noted that from the application perspective it appears that all updates A,B,C and D have been accepted, that is, the collision handling is completely hidden from user-space. Using the collision detection and LUT status interrupts, the application (in conjunction with the driver) could also choose to completely serialize the request (such that between the first and second D update, no other update is accepted).

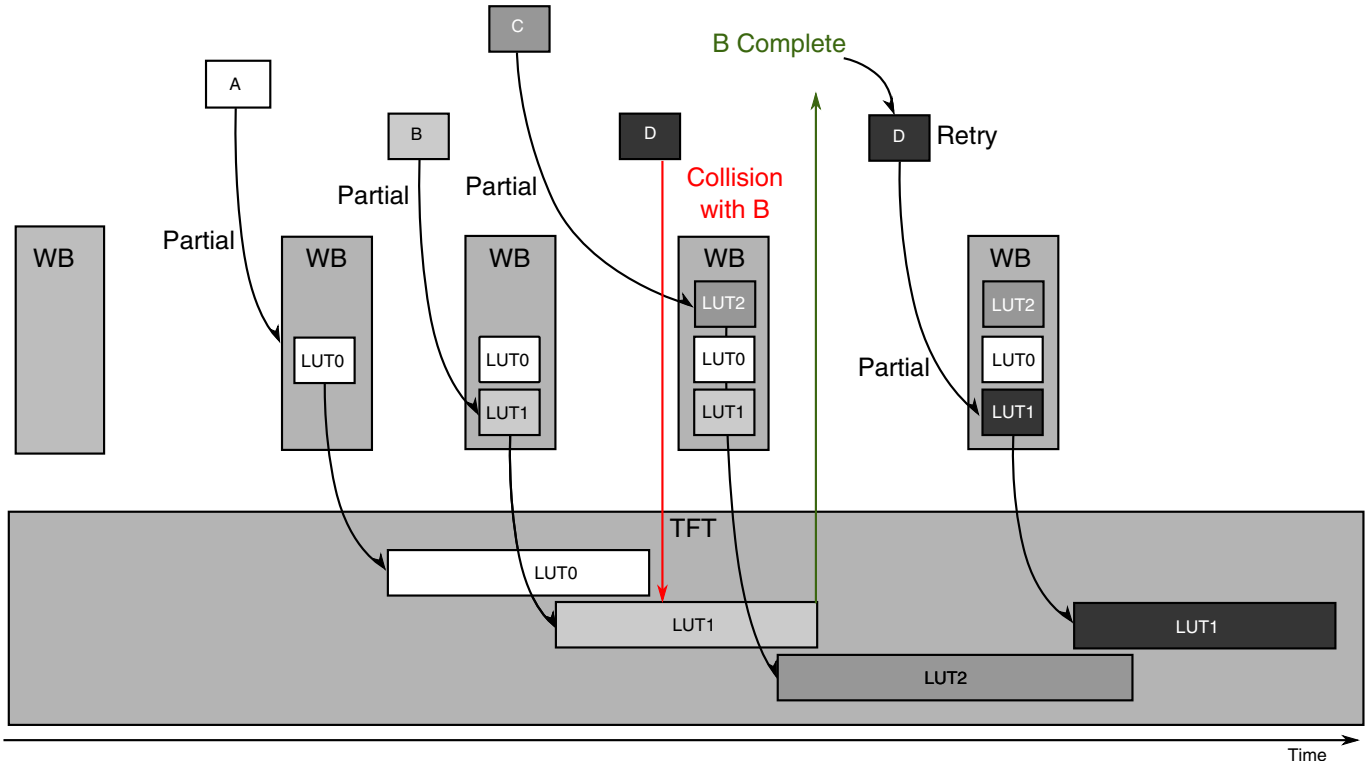


Figure 24-12. Collision Sequence

24.3.6.3 Multiple Update Flow

In most applications, such as e-book readers, besides full-screen page turns, user-interfaces require multiple updates to be performed.

Examples of this include cursors, pen-input, drop-down menus (with highlighting) and other movable graphical objects. As previously mentioned, each update takes a finite amount of time to complete (up to 1 second depending on the waveform mode). As such, it would be too prohibitive to only process one update at a time (serialization)-except in cases where a direct update waveform is used for B&W content.

As described in Initiating a Display Update, there are two conditions that must be met before the SW driver can issue a new update request:

- The Working buffer (WB) update process must be complete
- There must be an available LUT.

Assuming a free LUT is available, the EPDC is designed to have the capability to start new updates at each TFT frame scan (even being able to commence multiple updates within a frame scan blanking period). The general flow for sequential updates (concurrent or serialized) is shown in the following figure.

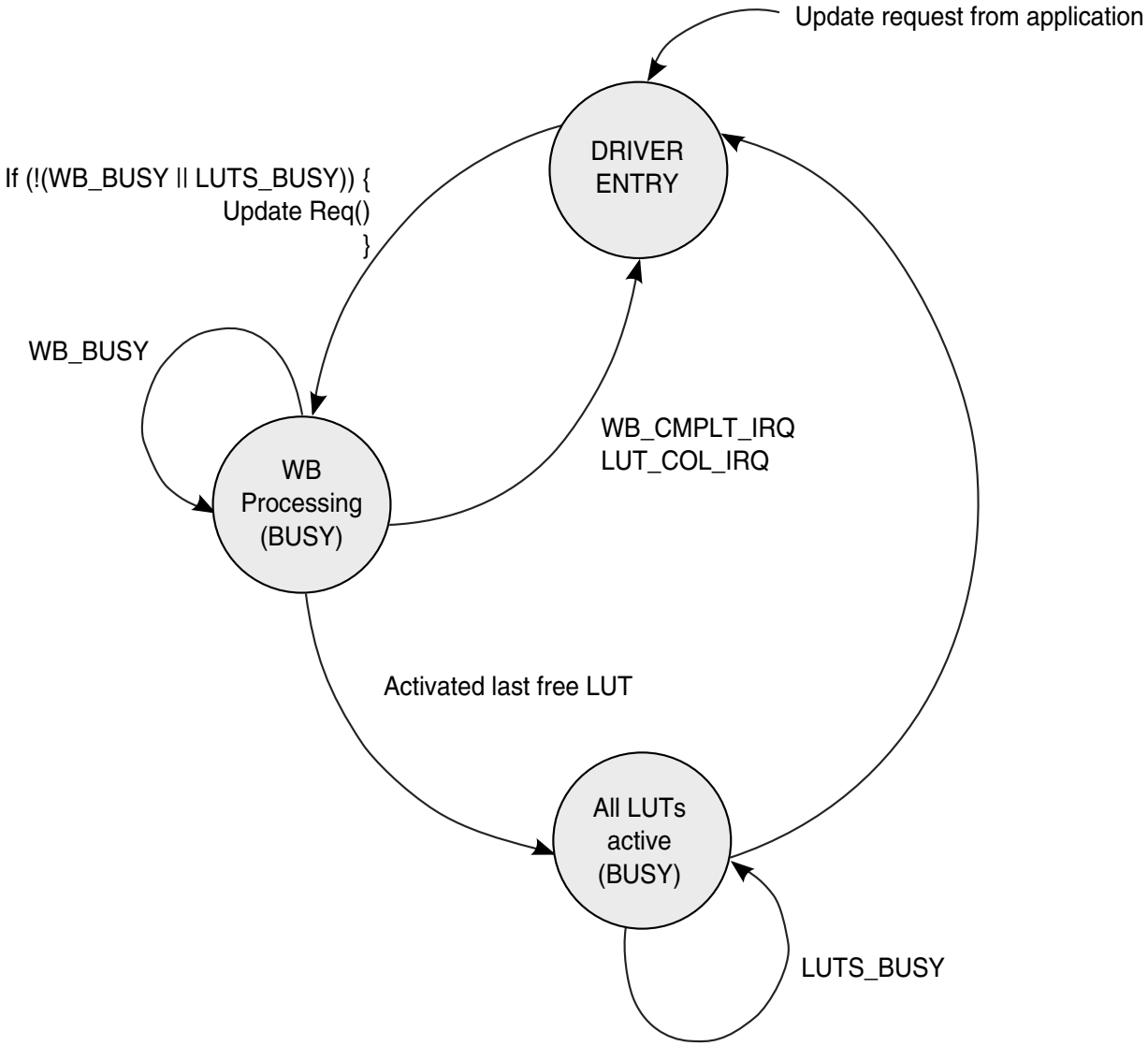


Figure 24-13. Driver Update Flow

This flow assumes that the driver is entered via interrupts. The two busy states are shown to illustrate the conditions under which the driver shall not issue new updates. In general, the driver should be throttled by the WB_CMPLT_IRQ interrupt. Because the EPDC can support up to 16 concurrent updates, its possible to re-enter the update-request (show as Update_Req() in the figure above) any-time that the WB is not busy (even though LUTs are active and the display is being updated). Its possible that if there are 15 LUTs currently active and a new update is sent, there will be a WB_CMPLT_IRQ interrupt issued, but the driver at that time would be in a busy state (because LUTS_BUSY is signaled). The driver shall be re-entered on the next LUTn_CMPLT_IRQ interrupt during which time it can issue a new update (since WB_BUSY and LUTS_BUSY status bits are low).

The update-processing process (which is controlled by the MBM module) is asynchronous to the actual TFT refresh operations (which carry out the physical screen updates), that is, there is no particular relationship between the WB_BUSY status and the state of the screen update. In contrast, the LUTn_CMPLT_IRQ interrupt status and EPDC_STATUS_LUTS provide the status of the physical waveform update. This means that when LUTn_CMPLT_IRQ interrupt fires, it means that the last physical TFT frame scan just completed for that update/LUT.

The EPDC implements proprietary methods to efficiently process the WB such that the WB processing time (the time from the SW driver writing EPDC_UPD_CTRL to the time EPDC_IRQ[WB_CMPLT_IRQ] fires), is dependent on the size of the update rectangle. This allows smaller updates to be processed faster. In most EPD applications, this is ideal because its typically smaller sprites that require the fastest performance. In addition to this the EPDC implements methods to allow store pending updates and activate them on the next available frame-scan. This means for example, that if the TFT scanning is currently in the active region and that during this time a series of small update requests are sent, the EPDC can begin all of those updates on the next available blanking period.

The figure below shows an example of two larger update request, one to LUT4 and one to LUT5. These update request are shown in the context of already active LUTs and thus active frame scan times. The times during which WB_BUSY is high and LUT_BUSY is high are defined as "blocking" because no new updates can be accepted. It should also be noted that new updates are physically commenced by the EPDC on blanking periods. From a SW programming model perspective, the EPDC_STATUS_LUTS[LUTn_STATUS] is activated immediately upon the update request being written to the EPDC. The physical frame-scanning for this particular update does not being commenced until the next available frame scan.

It should be noted that updates and WB processing can actually occur during the blanking period (or can begin in the active scan time and continue into the blanking period). For this reason, the EPDC provides some tuning controls via EPDC_TCE_CTRL[VSCAN_HOLDOFF] such that more time can be given to finish processing update request and less time to "locking down" the set of LUTs that will be activated for the upcoming active scan. The EPDC will start to pre-fill its refresh pixel FIFOs after the VSCAN_HOLDOFF period. Its possible to tune this value to allow the minimum time for the pixel FIFO pre-fill operation (making sure that refresh under-runs do not occur). If a pixel-FIFO under-run occurs, EPDC_IRQ[TCE_UNDERRUN_IRQ] will fire.

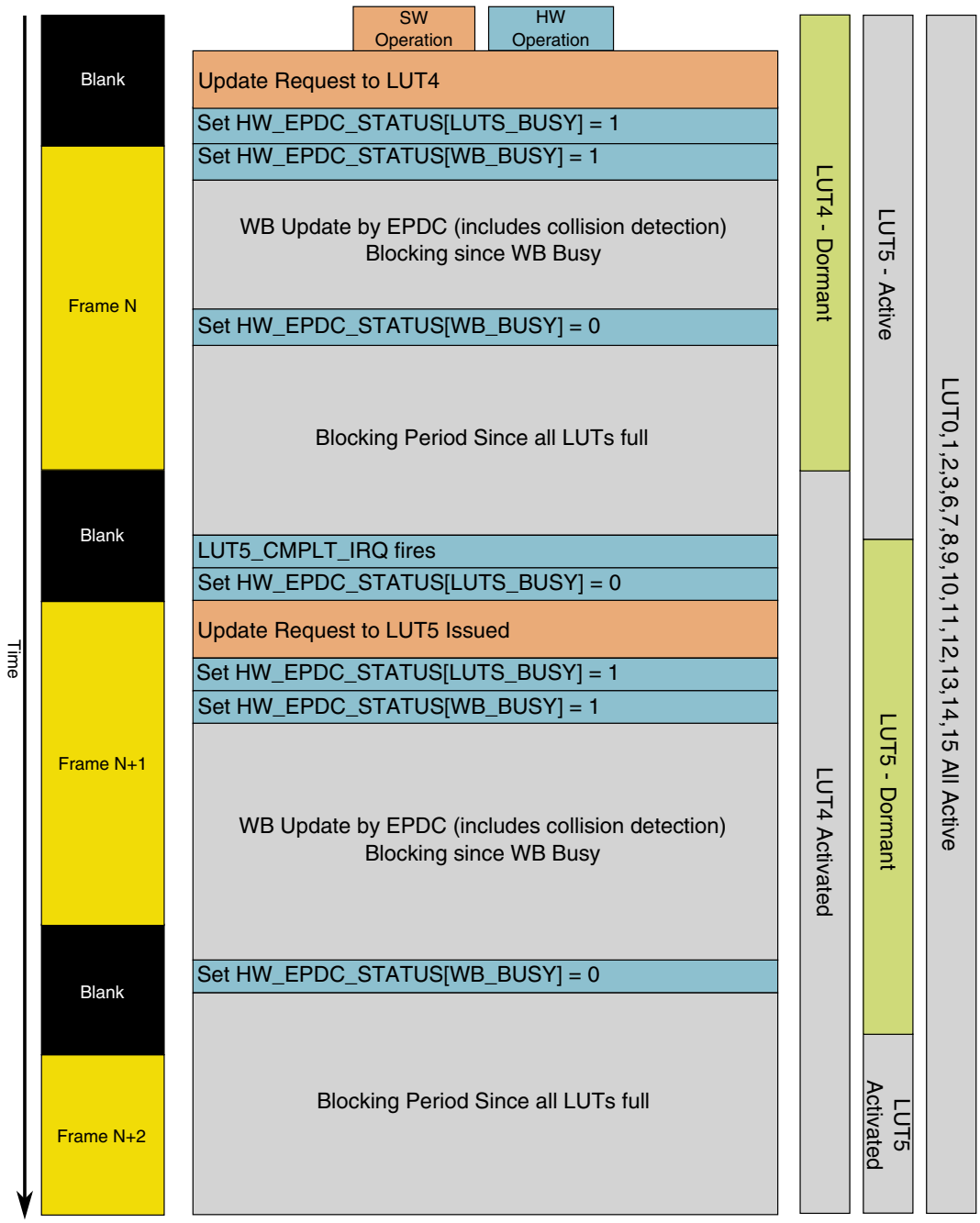


Figure 24-14. Temporal Flow of Update Processing

24.3.7 Architectural Clock Gating (Low Power Mode)

As with most modern IP, the EPDC contains low-level hardware controlled automatic clock gating throughout the design.

These clock gating elements typically gate clocks locally for individual or groups of sequential elements (flip-flops) when these registers are not currently active.

This tier of clock gating is useful for reducing power during active operation of the EPDC.

Most e-paper applications utilizing EPDs involve the processor and display controller to be in a low-power inactive state most of the time. This is because EPD displays only require refresh operations when the display content is being updated. As such, in between update operations, the EPDC provides a module level clock-gating feature that can be controlled by the driver SW as part of a higher-level power management solution.

The EPDC module can have all its clocks completely gated-off by the setting of EPDC_CTRL[CLKGATE]. There a couple of considerations that the SW driver must adhere to before placing the EPDC into this low power state:

- All active LUTs (updates) must be complete
- Any TFT operations must be complete (including any final blanking operations)

In order to maximize display update frequency (that is, the ability intercept new updates during each frame-scan), the LUT completion interrupts fire on the very end of the last frame's active scan period for that update. This means that the TCE still must compete the final "FRAME_END" blanking time to guarantee coherency of its state machines with the panel state (that is, receiving the final LUT completion IRQ is not enough to determine that the EPDC can have all its clocks shut down). Based on this, the EPDC provides the necessary interrupt and status bits allow the driver to correctly perform the clock-gating operation, but using the following method:

- Once the final EPDC_IRQ[LUTn_CMPLT_IRQ] is reached, the driver should enable the TCE_IDLE interrupt mask bit via EPDC_IRQ_MASK[TCE_IDLE_IRQ_EN], un-mask interrupts and return.
- Once the final blanking time of the last LUT (FRAME_END time) is completed and the TCE reaches its idle state, the EPDC shall assert the EPDC_IRQ[TCE_IDLE_IRQ] interrupt which will cause the driver to re-enter. At this point, the SW knows that the EPDC is in a completely idle state and can perform a set on EPDC_CTRL[CLKGATE]

Alternative implementations of the driver could simply involve polling for the EPDC_IRQ[TCE_IDLE_IRQ] interrupt status bit (which operates regardless of the interrupt being masked or not) before setting the clock-gate (this avoids two iterations of interrupt based driver-entry, but does require a wait loop in the interrupt handler which may be undesirable).

It should be noted that the time interval between the last LUTs LUTn_CMPLT_IRQ time and TCE_IDLE_IRQ interrupt is equal to the FRAME_END time.

When the application is ready to send more updates, the driver can immediately re-enable the EPDC by un-gating the module-level clock by clearing the EPDC_CTRL[CLKGATE] field. This un-gating of the clock is instant and the driver is free to immediately send update requests.

24.3.8 Performance Tuning and Considerations

The EPDC is designed to meet the performance requirements of the most demanding e-paper applications. These performance targets must be met in the context of highly integrated SoCs. In such a context, system memory is typically shared across multiple masters including the ARM platform. Because of this, there is no guarantee of latency from system memory and as such the EPDC provides a number of mechanisms to help deal with this as well as providing some ability to tune various parameters allowing the user to trade-off update processing performance with refresh functionality.

24.3.8.1 Memory and Bus Bandwidth Requirements

The EPDC MBM module contains up to 6 internal AXI requesting functions (5 read and 1 write). These are arbitrated internally before being sent to the SoC memory system. The EPDC interfaces to the SoC bus system via a dual-channel 64-bit AXI bus master port. Table 24-2 shows the various internal requesters and their properties. Note that there are two round-robin loops. Because there is only one write source, and the EPDC supports dual-channel AXI, there is no arbitration on the WB write operations.

Table 24-2. EPDC Bus Requestor Profiles

Operation	Priority	Burst Length (x8 Bytes)	Maximum Outstanding Transactions	FIFO size
Wavefor LUT read	0	8	2	16x68
WB pixel read FIFO0 (refresh)	RR 1	8 or 16	8	256x64
WB pixel read FIFO1 (refresh)	RR 1	8 or 16	8	256x64
WB read for update processing	RR 2	8	2	16x64
WB write for update processing	N/A	8	2	16x64
UPD read for update processing	RR 2	8	2	16x64

As described in Memory Requirements, it can be seen that all lines in a buffer are padded to the nearest 8 byte boundary. This is important to note for the bandwidth calculations. The maximum bandwidth scenario involves processing a full-screen update while

performing refresh operations, but refresh operations always have the highest priority. Based on this, the memory bandwidth required by the EPDC is a function of the refresh pixel rate which in turn is a function of resolution, frame-rate and blanking/active time.

The refresh bandwidth can be calculated as follows:

Active Time = Active time of the frame-scan time as a fraction (for example, 0.8)

Frame Rate = TFT frame refresh rate (Hz) (for example, 106 Hz)

Bandwidth (MB/S) = roundup4(EPDC_RES[HORIZONTAL]) x EPDC_RES[VERTICAL] x Frame Rate x (1/Active Time) x 2 x$(1/1024^2)$

As an example, for a panel with resolution 2048 x 1536 with 106 Hz refresh and estimated 80% active time, 20% blanking time, the refresh bandwidth is:

BW (MB/S) = 2048 x1536 x 106 x (1/0.8) x 2 x $(1/1024^2)$ = 795 MB/s

(note that 2048 mod 4 = 0), that is, as per 7.4.1, the WB requires 2 bytes per pixel.

Remaining bandwidth is used for other operations. Because update processing operations are not real-time (that is, do not necessarily have to occur at the refresh frame-rate), the time required to perform the update can be calculated as a function of the available bandwidth (actual bandwidth minus refresh bandwidth) and the update size.

24.3.8.2 Pixel Latency FIFOs

The EPDC contains two working buffer latency FIFOs which are used to load working buffer pixel data which is used by the TCE to perform the panel refresh operations. These FIFOs are sized at 1024 pixels each in order to provide significant system-memory latency tolerance. One pixel FIFO is dedicated for the main screen and the second is in dual-scan cases for the second half of the screen.

Under no circumstance should the EPDC pixel FIFOs reach an under-run condition. The EPDC provides an interrupt status bit to flag such as condition (TCE_UNDERRUN_IRQ). During development, this interrupt must be enabled. The pixel values stored in the FIFOs are using by the TCE to perform look-up operations (from the LUTs) to generate TFT voltage control pixels. If an under-run occurs, the result will be unknown data being used as the source for the look-ups which can damage the panel.

24.3.8.3 Basic Watermarking Control

The EPDC provides a basic pixel pre-fetching control mechanism that is applied to the beginning of a new update (i.e. from an idle state). This control is provided by EPDC_FIFOCTRL[FIFO_INIT_LEVEL]. The control allows the value to be set between 0-255, with a reset value of 128. Because the internal width of the FIFO holds 4-pixels per entry, the actual number of pixels is (FIFO_INIT_LEVEL+1) x 4.

The EPDC uses this value to pre-fill the pixel latency FIFO(s) to this level before the TCE commences the refresh operations which drive the update to the panel. This control has no effect on the FIFO(s) when the EPDC is performing sequential frame-scans. Please refer to 7.8.4 for details on influencing FIFO pre-fill operations during active scan times.

It must be noted that this value must be set less than (EPDC_RES[VERTICAL] x EPDC_RE[HORIZONTAL]) / 4.

24.3.8.4 Update/Refresh Tuning (VSCAN_HOLDOFF)

During active frame-scan time, the EPDC provides the ability to process and display new update requests. Because all pixels in the WB pertaining to a particular update are bound to a single LUT. Until all pixels pertaining to that update have been processed in the WB, its LUT cannot be activated. Referring to Figure 24-14, it can be seen that the blanking period provides time to pre-load the pixel FIFOs, but also allows time for completing WB processing updates and loading the associated frame's worth of waveform data into LUT memories.

The EPDC_TCE_CTRL[VSCAN_HOLDOFF] control field allows control over the vertical blanking period allotment for working buffer processing and pre-loading of the pixel FIFO. The time-window defined by VSCAN_HOLDOFF is allotted for working-buffer processing and LUT loading (of newly processed updates). The remainder of the time is allotted for pre-filling the pixel FIFO

It must be noted that aggressive use of VSCAN_HOLDOFF can result in pixel FIFO under-run (since the FIFO pre-fill time is compromised).

24.3.8.5 System-Level Arbitration Control

As previously mentioned, the TCE refresh operation is time critical. Because the EPDC is designed to be integrated into a multi-master SoC, system-memory is shared between multiple requesters in the system. The EPDC provides a dynamic Quality of Service (QoS) priority elevation mechanism that can be leveraged at the SoC level.

The EPDC_FIFOCTRL registers can be used in to drive the epdc_panic output signal during times when the EPDC requests require priority elevation. The expectation from the EPDC, is that as long as epdc_panic is asserted, transaction requests from the EPDC shall have their priority elevated at the SoC-level bus arbitration control points.

Referring to Figure 24-15, the registers and associated functionality are as follows:

- EPDC_FIFOCTRL[ENABLE_PRIORITY]-This must be set to enable epdc_panic functionality.
- EPDC_FIFOCTRL[FIFO_L_LEVEL]-Determines FIFO threshold which when crossed causes epdc_panic to assert.
- EPDC_FIFOCTRL[FIFO_H_LEVEL]-Once epdc_panic has been asserted (due to the pixel FIFO(s) crossing FIFO_L_LEVEL, this value (which must be set higher than FIFO_L_LEVEL), determines the point at which epdc_panic shall de-assert as the FIFO(s) fills.

It should be noted that the delta between FIFO_H_LEVEL and FIFO_L_LEVEL is intended as a "de-bounce" mechanism. The purpose is to avoid being in a constant "panic" situation which would cause thrashing of the epdc_panic signal.

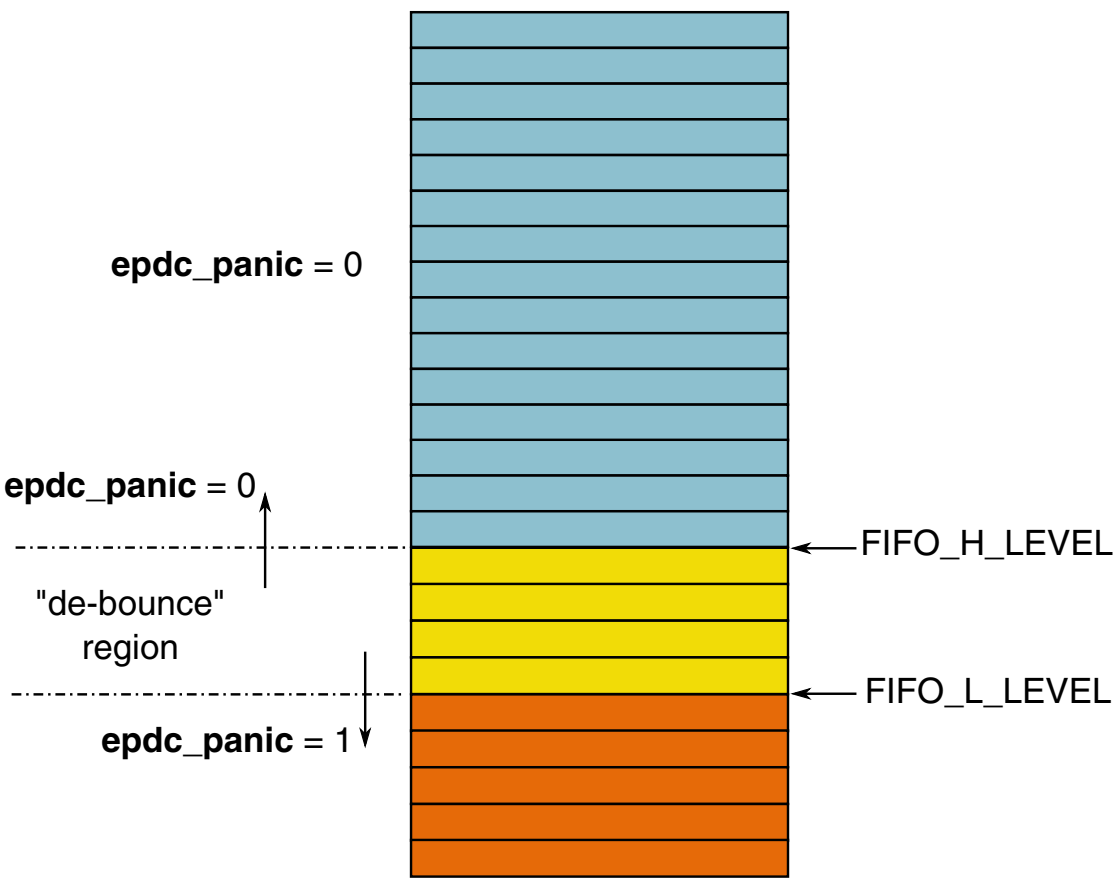


Figure 24-15. FIFO Priority Elevation

24.4 Programmable Registers

EPDC Register Format Summary

EPDC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_0000	EPDC Control Register (EPDC_CTRL)	32	R/W	C000_0000h	24.4.1/ 1193
4101_0004	EPDC Control Register (EPDC_CTRL_SET)	32	R/W	C000_0000h	24.4.1/ 1193
4101_0008	EPDC Control Register (EPDC_CTRL_CLR)	32	R/W	C000_0000h	24.4.1/ 1193
4101_000C	EPDC Control Register (EPDC_CTRL_TOG)	32	R/W	C000_0000h	24.4.1/ 1193
4101_0020	EPDC Waveform Address Pointer (EPDC_WVADDR)	32	R/W	0000_0000h	24.4.2/ 1194
4101_0030	EPDC Working Buffer Address (EPDC_WB_ADDR)	32	R/W	0000_0000h	24.4.3/ 1194
4101_0040	EPDC Screen Resolution (EPDC_RES)	32	R/W	0000_0000h	24.4.4/ 1195
4101_0050	EPDC Format Control Register (EPDC_FORMAT)	32	R/W	0000_0000h	24.4.5/ 1196
4101_0054	EPDC Format Control Register (EPDC_FORMAT_SET)	32	R/W	0000_0000h	24.4.5/ 1196
4101_0058	EPDC Format Control Register (EPDC_FORMAT_CLR)	32	R/W	0000_0000h	24.4.5/ 1196
4101_005C	EPDC Format Control Register (EPDC_FORMAT_TOG)	32	R/W	0000_0000h	24.4.5/ 1196
4101_00A0	EPDC FIFO Control Register (EPDC_FIFOCTRL)	32	R/W	0080_0000h	24.4.6/ 1197
4101_00A4	EPDC FIFO Control Register (EPDC_FIFOCTRL_SET)	32	R/W	0080_0000h	24.4.6/ 1197
4101_00A8	EPDC FIFO Control Register (EPDC_FIFOCTRL_CLR)	32	R/W	0080_0000h	24.4.6/ 1197
4101_00AC	EPDC FIFO Control Register (EPDC_FIFOCTRL_TOG)	32	R/W	0080_0000h	24.4.6/ 1197
4101_0100	EPDC Update Region Address (EPDC_UPD_ADDR)	32	R/W	0000_0000h	24.4.7/ 1198
4101_0120	EPDC Update Command Co-ordinate (EPDC_UPD_CORD)	32	R/W	0000_0000h	24.4.8/ 1198

Table continues on the next page...

EPDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_0140	EPDC Update Command Size (EPDC_UPD_SIZE)	32	R/W	0000_0000h	24.4.9/ 1199
4101_0160	EPDC Update Command Control (EPDC_UPD_CTRL)	32	R/W	0000_0000h	24.4.10/ 1200
4101_0164	EPDC Update Command Control (EPDC_UPD_CTRL_SET)	32	R/W	0000_0000h	24.4.10/ 1200
4101_0168	EPDC Update Command Control (EPDC_UPD_CTRL_CLR)	32	R/W	0000_0000h	24.4.10/ 1200
4101_016C	EPDC Update Command Control (EPDC_UPD_CTRL_TOG)	32	R/W	0000_0000h	24.4.10/ 1200
4101_0180	EPDC Update Fixed Pixel Control (EPDC_UPD_FIXED)	32	R/W	0000_0000h	24.4.11/ 1201
4101_0184	EPDC Update Fixed Pixel Control (EPDC_UPD_FIXED_SET)	32	R/W	0000_0000h	24.4.11/ 1201
4101_0188	EPDC Update Fixed Pixel Control (EPDC_UPD_FIXED_CLR)	32	R/W	0000_0000h	24.4.11/ 1201
4101_018C	EPDC Update Fixed Pixel Control (EPDC_UPD_FIXED_TOG)	32	R/W	0000_0000h	24.4.11/ 1201
4101_01A0	EPDC Temperature Register (EPDC_EPDC_TEMP)	32	R/W	0000_0000h	24.4.12/ 1201
4101_0200	EPDC Timing Control Engine Control Register (EPDC_EPDC_TCE_CTRL)	32	R/W	0000_0010h	24.4.13/ 1202
4101_0204	EPDC Timing Control Engine Control Register (EPDC_EPDC_TCE_CTRL_SET)	32	R/W	0000_0010h	24.4.13/ 1202
4101_0208	EPDC Timing Control Engine Control Register (EPDC_EPDC_TCE_CTRL_CLR)	32	R/W	0000_0010h	24.4.13/ 1202
4101_020C	EPDC Timing Control Engine Control Register (EPDC_EPDC_TCE_CTRL_TOG)	32	R/W	0000_0010h	24.4.13/ 1202
4101_0220	EPDC Timing Control Engine Source-Driver Config Register (EPDC_TCE_SDCFG)	32	R/W	0000_0000h	24.4.14/ 1204
4101_0224	EPDC Timing Control Engine Source-Driver Config Register (EPDC_TCE_SDCFG_SET)	32	R/W	0000_0000h	24.4.14/ 1204
4101_0228	EPDC Timing Control Engine Source-Driver Config Register (EPDC_TCE_SDCFG_CLR)	32	R/W	0000_0000h	24.4.14/ 1204
4101_022C	EPDC Timing Control Engine Source-Driver Config Register (EPDC_TCE_SDCFG_TOG)	32	R/W	0000_0000h	24.4.14/ 1204
4101_0240	EPDC Timing Control Engine Gate-Driver Config Register (EPDC_TCE_GDCFG)	32	R/W	0000_0000h	24.4.15/ 1205
4101_0244	EPDC Timing Control Engine Gate-Driver Config Register (EPDC_TCE_GDCFG_SET)	32	R/W	0000_0000h	24.4.15/ 1205

Table continues on the next page...

EPDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_0248	EPDC Timing Control Engine Gate-Driver Config Register (EPDC_TCE_GDCFG_CLR)	32	R/W	0000_0000h	24.4.15/ 1205
4101_024C	EPDC Timing Control Engine Gate-Driver Config Register (EPDC_TCE_GDCFG_TOG)	32	R/W	0000_0000h	24.4.15/ 1205
4101_0260	EPDC Timing Control Engine Horizontal Timing Register 1 (EPDC_TCE_HSCAN1)	32	R/W	0000_0000h	24.4.16/ 1206
4101_0264	EPDC Timing Control Engine Horizontal Timing Register 1 (EPDC_TCE_HSCAN1_SET)	32	R/W	0000_0000h	24.4.16/ 1206
4101_0268	EPDC Timing Control Engine Horizontal Timing Register 1 (EPDC_TCE_HSCAN1_CLR)	32	R/W	0000_0000h	24.4.16/ 1206
4101_026C	EPDC Timing Control Engine Horizontal Timing Register 1 (EPDC_TCE_HSCAN1_TOG)	32	R/W	0000_0000h	24.4.16/ 1206
4101_0280	EPDC Timing Control Engine Horizontal Timing Register 2 (EPDC_TCE_HSCAN2)	32	R/W	0000_0000h	24.4.17/ 1207
4101_0284	EPDC Timing Control Engine Horizontal Timing Register 2 (EPDC_TCE_HSCAN2_SET)	32	R/W	0000_0000h	24.4.17/ 1207
4101_0288	EPDC Timing Control Engine Horizontal Timing Register 2 (EPDC_TCE_HSCAN2_CLR)	32	R/W	0000_0000h	24.4.17/ 1207
4101_028C	EPDC Timing Control Engine Horizontal Timing Register 2 (EPDC_TCE_HSCAN2_TOG)	32	R/W	0000_0000h	24.4.17/ 1207
4101_02A0	EPDC Timing Control Engine Vertical Timing Register (EPDC_TCE_VSCAN)	32	R/W	0000_0000h	24.4.18/ 1207
4101_02A4	EPDC Timing Control Engine Vertical Timing Register (EPDC_TCE_VSCAN_SET)	32	R/W	0000_0000h	24.4.18/ 1207
4101_02A8	EPDC Timing Control Engine Vertical Timing Register (EPDC_TCE_VSCAN_CLR)	32	R/W	0000_0000h	24.4.18/ 1207
4101_02AC	EPDC Timing Control Engine Vertical Timing Register (EPDC_TCE_VSCAN_TOG)	32	R/W	0000_0000h	24.4.18/ 1207
4101_02C0	EPDC Timing Control Engine OE timing control Register (EPDC_TCE_OE)	32	R/W	0000_0000h	24.4.19/ 1208
4101_02C4	EPDC Timing Control Engine OE timing control Register (EPDC_TCE_OE_SET)	32	R/W	0000_0000h	24.4.19/ 1208
4101_02C8	EPDC Timing Control Engine OE timing control Register (EPDC_TCE_OE_CLR)	32	R/W	0000_0000h	24.4.19/ 1208
4101_02CC	EPDC Timing Control Engine OE timing control Register (EPDC_TCE_OE_TOG)	32	R/W	0000_0000h	24.4.19/ 1208
4101_02E0	EPDC Timing Control Engine Driver Polarity Register (EPDC_TCE_POLARITY)	32	R/W	0000_001Eh	24.4.20/ 1209
4101_02E4	EPDC Timing Control Engine Driver Polarity Register (EPDC_TCE_POLARITY_SET)	32	R/W	0000_001Eh	24.4.20/ 1209

Table continues on the next page...

EPDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_02E8	EPDC Timing Control Engine Driver Polarity Register (EPDC_TCE_POLARITY_CLR)	32	R/W	0000_001Eh	24.4.20/ 1209
4101_02EC	EPDC Timing Control Engine Driver Polarity Register (EPDC_TCE_POLARITY_TOG)	32	R/W	0000_001Eh	24.4.20/ 1209
4101_0300	EPDC Timing Control Engine Timing Register 1 (EPDC_TCE_TIMING1)	32	R/W	0000_0000h	24.4.21/ 1210
4101_0304	EPDC Timing Control Engine Timing Register 1 (EPDC_TCE_TIMING1_SET)	32	R/W	0000_0000h	24.4.21/ 1210
4101_0308	EPDC Timing Control Engine Timing Register 1 (EPDC_TCE_TIMING1_CLR)	32	R/W	0000_0000h	24.4.21/ 1210
4101_030C	EPDC Timing Control Engine Timing Register 1 (EPDC_TCE_TIMING1_TOG)	32	R/W	0000_0000h	24.4.21/ 1210
4101_0310	EPDC Timing Control Engine Timing Register 2 (EPDC_TCE_TIMING2)	32	R/W	0000_0001h	24.4.22/ 1211
4101_0314	EPDC Timing Control Engine Timing Register 2 (EPDC_TCE_TIMING2_SET)	32	R/W	0000_0001h	24.4.22/ 1211
4101_0318	EPDC Timing Control Engine Timing Register 2 (EPDC_TCE_TIMING2_CLR)	32	R/W	0000_0001h	24.4.22/ 1211
4101_031C	EPDC Timing Control Engine Timing Register 2 (EPDC_TCE_TIMING2_TOG)	32	R/W	0000_0001h	24.4.22/ 1211
4101_0320	EPDC Timing Control Engine Timing Register 3 (EPDC_TCE_TIMING3)	32	R/W	0000_0001h	24.4.23/ 1212
4101_0324	EPDC Timing Control Engine Timing Register 3 (EPDC_TCE_TIMING3_SET)	32	R/W	0000_0001h	24.4.23/ 1212
4101_0328	EPDC Timing Control Engine Timing Register 3 (EPDC_TCE_TIMING3_CLR)	32	R/W	0000_0001h	24.4.23/ 1212
4101_032C	EPDC Timing Control Engine Timing Register 3 (EPDC_TCE_TIMING3_TOG)	32	R/W	0000_0001h	24.4.23/ 1212
4101_0400	EPDC IRQ Mask Register (EPDC_IRQ_MASK)	32	R/W	0000_0000h	24.4.24/ 1213
4101_0404	EPDC IRQ Mask Register (EPDC_IRQ_MASK_SET)	32	R/W	0000_0000h	24.4.24/ 1213
4101_0408	EPDC IRQ Mask Register (EPDC_IRQ_MASK_CLR)	32	R/W	0000_0000h	24.4.24/ 1213
4101_040C	EPDC IRQ Mask Register (EPDC_IRQ_MASK_TOG)	32	R/W	0000_0000h	24.4.24/ 1213
4101_0420	EPDC Interrupt Register (EPDC_IRQ)	32	R/W	0000_0000h	24.4.25/ 1215
4101_0424	EPDC Interrupt Register (EPDC_IRQ_SET)	32	R/W	0000_0000h	24.4.25/ 1215

Table continues on the next page...

EPDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_0428	EPDC Interrupt Register (EPDC_IRQ_CLR)	32	R/W	0000_0000h	24.4.25/ 1215
4101_042C	EPDC Interrupt Register (EPDC_IRQ_TOG)	32	R/W	0000_0000h	24.4.25/ 1215
4101_0440	EPDC Status Register - LUTs (EPDC_EPDC_STATUS_LUTS)	32	R	0000_0000h	24.4.26/ 1217
4101_0444	EPDC Status Register - LUTs (EPDC_EPDC_STATUS_LUTS_SET)	32	R	0000_0000h	24.4.26/ 1217
4101_0448	EPDC Status Register - LUTs (EPDC_EPDC_STATUS_LUTS_CLR)	32	R	0000_0000h	24.4.26/ 1217
4101_044C	EPDC Status Register - LUTs (EPDC_EPDC_STATUS_LUTS_TOG)	32	R	0000_0000h	24.4.26/ 1217
4101_0460	EPDC Status Register - Next Available LUT (EPDC_STATUS_NEXTLUT)	32	R	0000_010Fh	24.4.27/ 1219
4101_0480	EPDC LUT Collision Status (EPDC_STATUS_COL)	32	R	0000_0000h	24.4.28/ 1220
4101_0484	EPDC LUT Collision Status (EPDC_STATUS_COL_SET)	32	R	0000_0000h	24.4.28/ 1220
4101_0488	EPDC LUT Collision Status (EPDC_STATUS_COL_CLR)	32	R	0000_0000h	24.4.28/ 1220
4101_048C	EPDC LUT Collision Status (EPDC_STATUS_COL_TOG)	32	R	0000_0000h	24.4.28/ 1220
4101_04A0	EPDC General Status Register (EPDC_STATUS)	32	R	0000_0000h	24.4.29/ 1222
4101_04A4	EPDC General Status Register (EPDC_STATUS_SET)	32	R	0000_0000h	24.4.29/ 1222
4101_04A8	EPDC General Status Register (EPDC_STATUS_CLR)	32	R	0000_0000h	24.4.29/ 1222
4101_04AC	EPDC General Status Register (EPDC_STATUS_TOG)	32	R	0000_0000h	24.4.29/ 1222
4101_0500	EPDC Debug register (EPDC_DEBUG)	32	R/W	0000_0000h	24.4.30/ 1223
4101_0504	EPDC Debug register (EPDC_DEBUG_SET)	32	R/W	0000_0000h	24.4.30/ 1223
4101_0508	EPDC Debug register (EPDC_DEBUG_CLR)	32	R/W	0000_0000h	24.4.30/ 1223
4101_050C	EPDC Debug register (EPDC_DEBUG_TOG)	32	R/W	0000_0000h	24.4.30/ 1223
4101_0540	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT0)	32	R	0000_0000h	24.4.31/ 1224

Table continues on the next page...

EPDC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_0550	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT1)	32	R	0000_0000h	24.4.31/ 1224
4101_0560	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT2)	32	R	0000_0000h	24.4.31/ 1224
4101_0570	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT3)	32	R	0000_0000h	24.4.31/ 1224
4101_0580	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT4)	32	R	0000_0000h	24.4.31/ 1224
4101_0590	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT5)	32	R	0000_0000h	24.4.31/ 1224
4101_05A0	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT6)	32	R	0000_0000h	24.4.31/ 1224
4101_05B0	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT7)	32	R	0000_0000h	24.4.31/ 1224
4101_05C0	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT8)	32	R	0000_0000h	24.4.31/ 1224
4101_05D0	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT9)	32	R	0000_0000h	24.4.31/ 1224
4101_05E0	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT10)	32	R	0000_0000h	24.4.31/ 1224
4101_05F0	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT11)	32	R	0000_0000h	24.4.31/ 1224
4101_0600	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT12)	32	R	0000_0000h	24.4.31/ 1224
4101_0610	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT13)	32	R	0000_0000h	24.4.31/ 1224
4101_0620	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT14)	32	R	0000_0000h	24.4.31/ 1224
4101_0630	EPDC LUTn Debug Information register (EPDC_DEBUG_LUT15)	32	R	0000_0000h	24.4.31/ 1224
4101_0700	EPDC General Purpose I/O Debug register (EPDC_GPIO)	32	R/W	0000_0000h	24.4.32/ 1225
4101_0704	EPDC General Purpose I/O Debug register (EPDC_GPIO_SET)	32	R/W	0000_0000h	24.4.32/ 1225
4101_0708	EPDC General Purpose I/O Debug register (EPDC_GPIO_CLR)	32	R/W	0000_0000h	24.4.32/ 1225
4101_070C	EPDC General Purpose I/O Debug register (EPDC_GPIO_TOG)	32	R/W	0000_0000h	24.4.32/ 1225
4101_07F0	EPDC Version Register (EPDC_VERSION)	32	R	0100_0000h	24.4.33/ 1225

24.4.1 EPDC Control Register (EPDC_CTRL*n*)

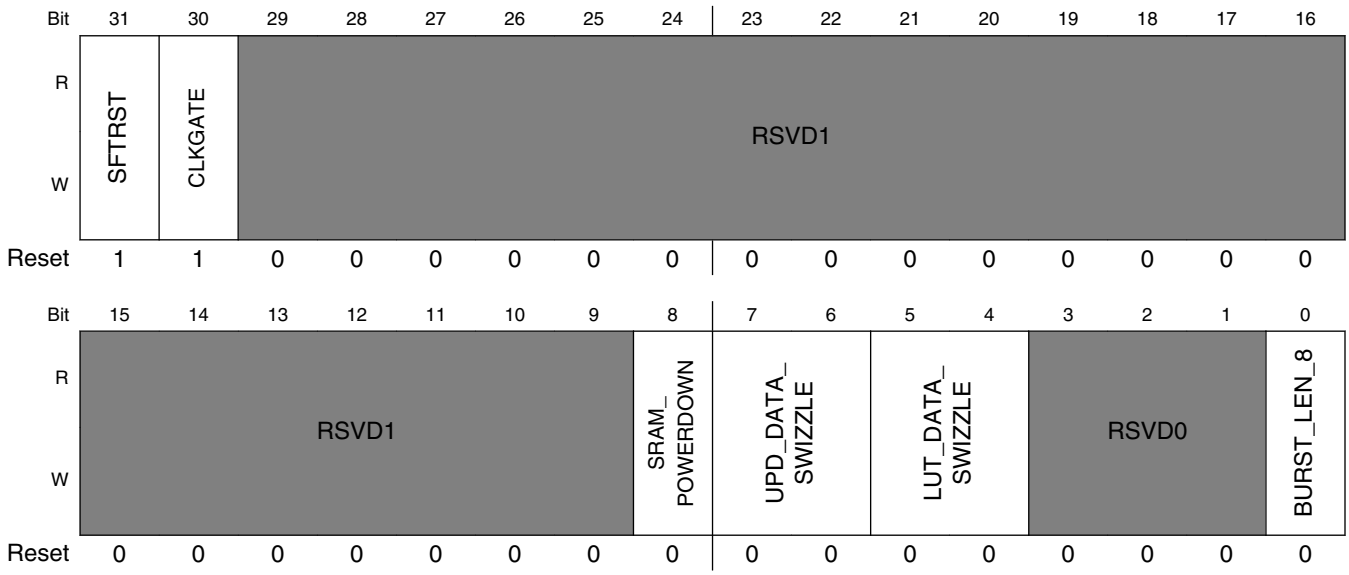
EPDC Main control register. This register controls various high-level functions of the EPDC.

Addresses: EPDC_CTRL is 4101_0000h base + 0h offset = 4101_0000h

EPDC_CTRL_SET is 4101_0000h base + 4h offset = 4101_0004h

EPDC_CTRL_CLR is 4101_0000h base + 8h offset = 4101_0008h

EPDC_CTRL_TOG is 4101_0000h base + Ch offset = 4101_000Ch



EPDC_CTRL*n* field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal EPDC operation. Set this bit to one (default) to disable clocking with the EPDC and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the EPDC block to its default state. Software is initialized by asserting SFTRST, the driver polls until CLKGATE is back to '0'.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29–9 RSVD1	Reserved.
8 SRAM_ POWERDOWN	Enable Power-down of embedded SRAM memories
7–6 UPD_DATA_ SWIZZLE	Specifies how to swap the bytes for the UPD data before the WB construction. Supported configurations: 0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x1 **ALL_BYTES_SWAP** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.

Table continues on the next page...

EPDC_CTRL*n* field descriptions (continued)

Field	Description
5–4 LUT_DATA_ SWIZZLE	Specifies how to swap the bytes for the LUT data before store to LUTRAM. Supported configurations: 0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x1 **ALL_BYTES_SWAP** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
3–1 RSVD0	Reserved.
0 BURST_LEN_8	0- EPDC display fifo logic will issue AXI bursts of length 16. When set to 1, the block will issue bursts of length 8.

24.4.2 EPDC Waveform Address Pointer (EPDC_WVADDR)

Address: EPDC_WVADDR is 4101_0000h base + 20h offset = 4101_0020h

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

EPDC_WVADDR field descriptions

Field	Description
31–0 ADDR	Start address of waveform tables. This address needs to be aligned to a 64-bit double word boundary.

24.4.3 EPDC Working Buffer Address (EPDC_WB_ADDR)

EPDC Working Buffer Address register controls various functions throughout the digital portion of the chip.

Address: EPDC_WB_ADDR is 4101_0000h base + 30h offset = 4101_0030h

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

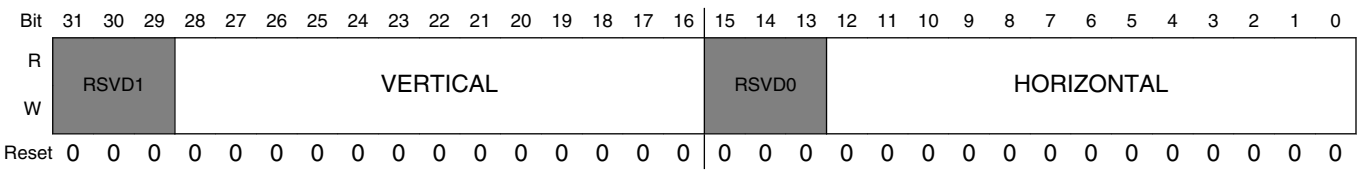
EPDC_WB_ADDR field descriptions

Field	Description
31–0 ADDR	Address for EPDC working buffer. This address must be a aligned to a 64-bit double-word boundary.

24.4.4 EPDC Screen Resolution (EPDC_RES)

EPDC Screen Resolution register defines the horizontal and vertical resolution of the target display panel.

Address: EPDC_RES is 4101_0000h base + 40h offset = 4101_0040h



EPDC_RES field descriptions

Field	Description
31–29 RSVD1	Reserved.
28–16 VERTICAL	Vertical Resoltion (in pixels)
15–13 RSVD0	Reserved.
12–0 HORIZONTAL	Horizontal Resolution (in pixels)

24.4.5 EPDC Format Control Register (EPDC_FORMAT*n*)

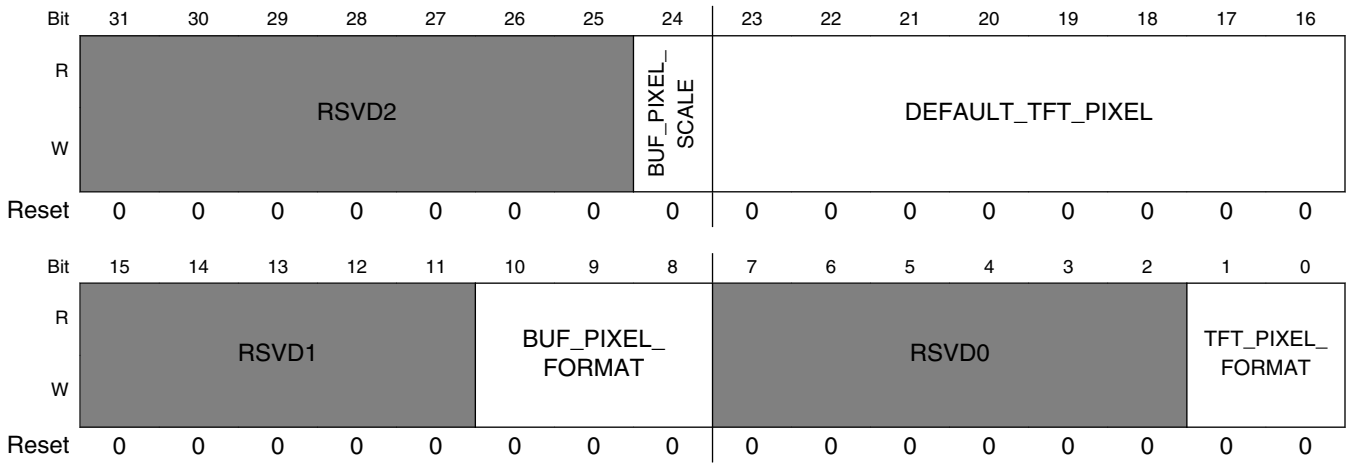
EPDC Pixel format control register defines formats for buffer and TFT pixels and controls various functions throughout the digital portion of the chip.

Addresses: EPDC_FORMAT is 4101_0000h base + 50h offset = 4101_0050h

EPDC_FORMAT_SET is 4101_0000h base + 54h offset = 4101_0054h

EPDC_FORMAT_CLR is 4101_0000h base + 58h offset = 4101_0058h

EPDC_FORMAT_TOG is 4101_0000h base + 5Ch offset = 4101_005Ch



EPDC_FORMAT*n* field descriptions

Field	Description
31–25 RSVD2	Reserved.
24 BUF_PIXEL_ SCALE	Selects method of conversion from 8-bit input. 0x0 **TRUNCATE** — Use Truncate method (LSB) 0x1 **ROUND** — Use rounding method (with saturation)
23–16 DEFAULT_TFT_ PIXEL	Default TFT pixel value. This value is used as the source-driver voltage value (TFT-pixel) for either partial-updates where a pixel has not changed or for any part of the screen which is not being updated during active frame scans.
15–11 RSVD1	Reserved.
10–8 BUF_PIXEL_ FORMAT	EPDC Input Buffer Pixel format. All update buffers are expected to have 8-bit grayscale pixels. This register defines which MSBs of those pixels are used. It must be noted that this format must match the waveform (P4N is not compatible with 3-bit waveforms) 0x2 **P2N** — 2-bit pixel 0x3 **P3N** — 3-bit pixel 0x4 **P4N** — 4-bit pixel 0x5 **P5N** — 5-bit pixel
7–2 RSVD0	Reserved.

Table continues on the next page...

EPDC_FORMAT*n* field descriptions (continued)

Field	Description
1–0 TFT_PIXEL_ FORMAT	EPDC TFT Pixel Format. This defines how many bits of the SDDO bus are required per pixel. This field must be consistent with the waveform and panel architecture. 0x0 **2B** — 2-bit 0x1 **2BV** — 2-bit and VCOM 0x2 **4B** — 4-bit 0x3 **4BV** — 4-bit and VCOM

24.4.6 EPDC FIFO Control Register (EPDC_FIFOCTRL*n*)

The EPDC FIFO Control Register allows for programmability of pixel FIFO watermarks used in conjunction with system arbitration hardware. This register houses FIFO control bits.

Addresses: EPDC_FIFOCTRL is 4101_0000h base + A0h offset = 4101_00A0h

EPDC_FIFOCTRL_SET is 4101_0000h base + A4h offset = 4101_00A4h

EPDC_FIFOCTRL_CLR is 4101_0000h base + A8h offset = 4101_00A8h

EPDC_FIFOCTRL_TOG is 4101_0000h base + ACh offset = 4101_00ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ENABLE_ PRIORITY	RSVD1							FIFO_INIT_LEVEL							
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIFO_H_LEVEL								FIFO_L_LEVEL							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPDC_FIFOCTRL*n* field descriptions

Field	Description
31 ENABLE_ PRIORITY	Enable watermark-based priority elevation mechanism. 1=Enabled, 0=Disabled. (Only applies to FIFO_H_LEVEL and FIFO_L_LEVEL)
30–24 RSVD1	Reserved.
23–16 FIFO_INIT_ LEVEL	This register sets the watermark for the pixel-fifo.

Table continues on the next page...

EPDC_FIFOCTRL*n* field descriptions (continued)

Field	Description
15–8 FIFO_H_LEVEL	Upper level value of FIFO watermark. Must be greater than FIFO_L_LEVEL. When the pixel FIFO reaches this level or above, the priority elevation request is negated
7–0 FIFO_L_LEVEL	Lower level value of FIFO watermark. When the pixel FIFO reaches this level or below, the priority elevation request is asserted.

24.4.7 EPDC Update Region Address (EPDC_UPD_ADDR)

Address: EPDC_UPD_ADDR is 4101_0000h base + 100h offset = 4101_0100h

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

EPDC_UPD_ADDR field descriptions

Field	Description
31–0 ADDR	Address for incoming region update. This address points to update region which will be processed into the working buffer. This address must be a aligned to a 64-bit double-word boundary. In addition the start address of each line must be a aligned to a 64-bit double-word boundary.

24.4.8 EPDC Update Command Co-ordinate (EPDC_UPD_CORD)

Address: EPDC_UPD_CORD is 4101_0000h base + 120h offset = 4101_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD1			YCORD													RSVD0			XCORD												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPDC_UPD_CORD field descriptions

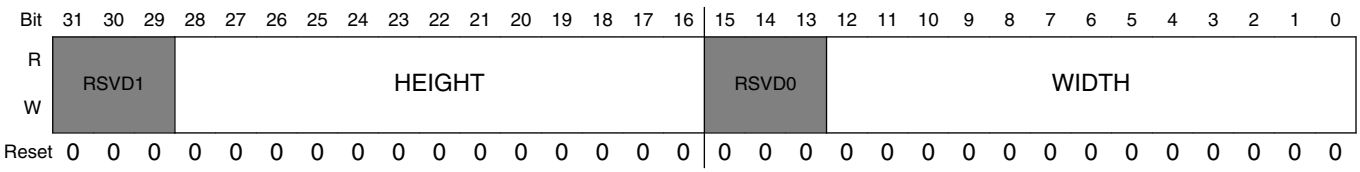
Field	Description
31–29 RSVD1	Reserved.
28–16 YCORD	Y co-ordinate for incoming region update
15–13 RSVD0	Reserved.
12–0 XCORD	X co-ordinate for incoming region update

24.4.9 EPDC Update Command Size (EPDC_UPD_SIZE)

EPDC Update Command Size register controls various functions throughout the digital portion of the chip.

-->

Address: EPDC_UPD_SIZE is 4101_0000h base + 140h offset = 4101_0140h



EPDC_UPD_SIZE field descriptions

Field	Description
31–29 RSVD1	Reserved.
28–16 HEIGHT	Height (in pixels)
15–13 RSVD0	Reserved.
12–0 WIDTH	Width (in pixels)

24.4.10 EPDC Update Command Control (EPDC_UPD_CTRL*n*)

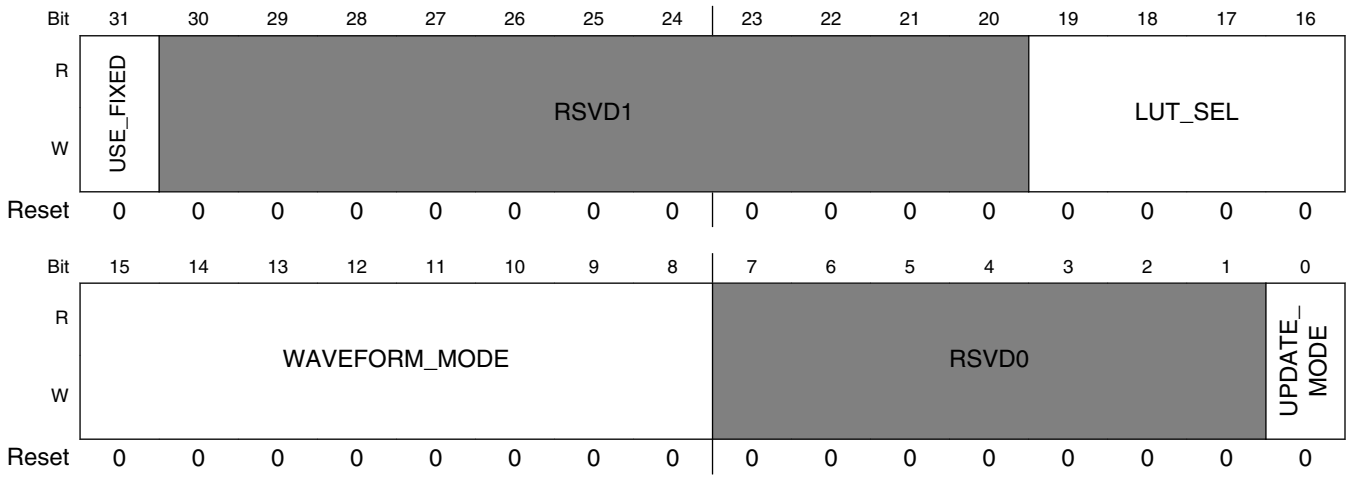
Writing to EPDC Update Command Control register triggers and update request operation.

Addresses: EPDC_UPD_CTRL is 4101_0000h base + 160h offset = 4101_0160h

EPDC_UPD_CTRL_SET is 4101_0000h base + 164h offset = 4101_0164h

EPDC_UPD_CTRL_CLR is 4101_0000h base + 168h offset = 4101_0168h

EPDC_UPD_CTRL_TOG is 4101_0000h base + 16Ch offset = 4101_016Ch



EPDC_UPD_CTRL*n* field descriptions

Field	Description
31 USE_FIXED	Use fixed pixel values (requires programming of EPDC_UPD_FIXED).
30–20 RSVD1	Reserved.
19–16 LUT_SEL	LUT select 0-15
15–8 WAVEFORM_ MODE	Waveform Mode 0-255
7–1 RSVD0	Reserved.
0 UPDATE_MODE	Update Mode 0x0 **PARTIAL** — Partial Update : only process changed pixels in region 0x1 **FULL** — Full Update : process all pixels in region

24.4.11 EPDC Update Fixed Pixel Control (EPDC_UPD_FIXED*n*)

EPDC Update Control register for fixed-pixel updates (enabled via EPDC_UPD_CTRL[USE_FIXED]).

Addresses: EPDC_UPD_FIXED is 4101_0000h base + 180h offset = 4101_0180h
EPDC_UPD_FIXED_SET is 4101_0000h base + 184h offset = 4101_0184h
EPDC_UPD_FIXED_CLR is 4101_0000h base + 188h offset = 4101_0188h
EPDC_UPD_FIXED_TOG is 4101_0000h base + 18Ch offset = 4101_018Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FIXNP_EN	FIXCP_EN	RSVD0													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIXNP								FIXCP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPDC_UPD_FIXED*n* field descriptions

Field	Description
31 FIXNP_EN	If set to 1, current updated region has the NP value defined by FIXNP
30 FIXCP_EN	If set to 1, current updated region has the CP value defined by FIXCP
29–16 RSVD0	Reserved.
15–8 FIXNP	NP value if fixenp_en is set to 1. Data in Y8 format.
7–0 FIXCP	CP value if fixecp_en is set to 1. Data in Y8 format.

24.4.12 EPDC Temperature Register (EPDC_EPDC_TEMP)

Address: EPDC_EPDC_TEMP is 4101_0000h base + 1A0h offset = 4101_01A0h

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

EPDC_EPDC_TEMP field descriptions

Field	Description
31–0 TEMPERATURE	Temperature Value. This value is simply an index (not a temperature value). The index is used by the EPDC to access the correct temperature compensated waveform.

24.4.13 EPDC Timing Control Engine Control Register (EPDC_EPDC_TCE_CTRL*n*)

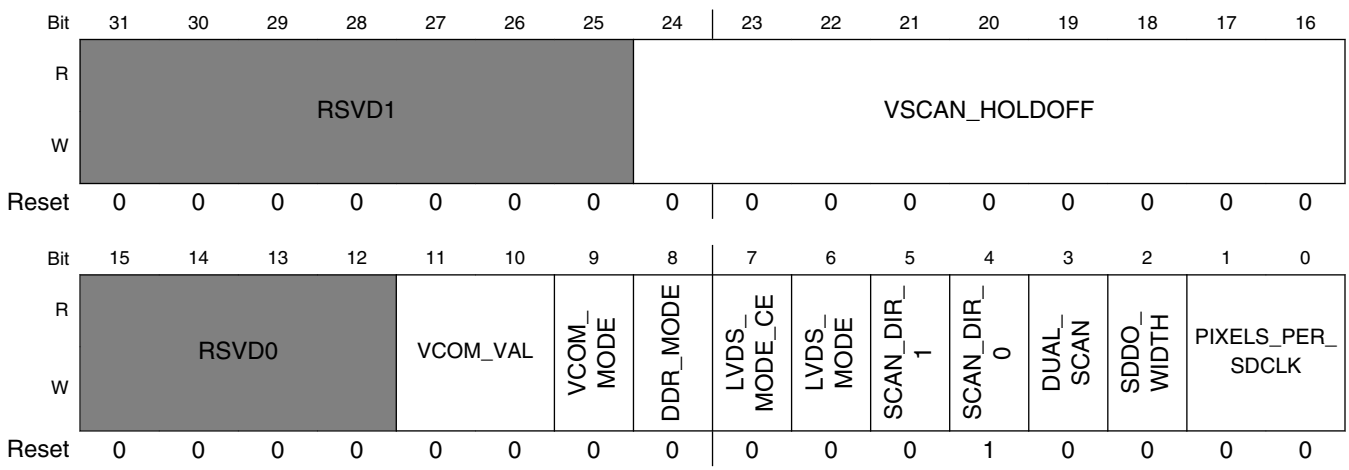
The EPDC Timing Control Engine Control Register houses Horizontal scan timing. Note that line data length is derived from EPDC_RES.

Addresses: EPDC_EPDC_TCE_CTRL is 4101_0000h base + 200h offset = 4101_0200h

EPDC_EPDC_TCE_CTRL_SET is 4101_0000h base + 204h offset = 4101_0204h

EPDC_EPDC_TCE_CTRL_CLR is 4101_0000h base + 208h offset = 4101_0208h

EPDC_EPDC_TCE_CTRL_TOG is 4101_0000h base + 20Ch offset = 4101_020Ch



EPDC_EPDC_TCE_CTRL*n* field descriptions

Field	Description
31–25 RSVD1	Reserved.
24–16 VSCAN_ HOLDOFF	This period (expressed in vertical lines), sets the portion of the vertical blanking available for new LUTs to be activated. The remainder of the blanking period is reserved for pre-filling the TCE pixel FIFOs. Increasing this value allows for multiple smaller updates to be intercepted by the current frame scan. This number should not exceed FRAME_END+FRAME_SYNC+FRAME_BEGIN. Increasing this value can improve the ability for any given update to intercept the next available frame-scan. Excessive values can result in TCE FIFO under-runs.
15–12 RSVD0	Reserved.
11–10 VCOM_VAL	When VCOM_MODE = MANUAL, this value is used to manually set the VCOM value for the VCOM[1:0] pins

Table continues on the next page...

EPDC_EPDC_TCE_CTRL*n* field descriptions (continued)

Field	Description
9 VCOM_MODE	This field determines the method used to drive the VCOM signal. 0x0 **MANUAL** — VCOM Value is set manually using VCOM_VAL field 0x1 **AUTO** — VCOM Value is used from waveform
8 DDR_MODE	If set, SDDO data is driven on both positive and negative edges of SDCLK. Note that this mode is not supported when SDDO_BUS_FORMAT=16BIT and LVDS is not used. This must always be set when LVDS_MODE is set.
7 LVDS_MODE_CE	If set (together with LVDS_MODE=1), SDCE[9:5] shall be driven as the differential inverse of SDCE[4:0]. In this mode the EPDC only supports 5 CE lines.
6 LVDS_MODE	If set, the upper 8-bit of the SDDO bus are used for LVDS differential signalling. Note that this can only be used when SDDO_BUS_FORMAT is set to 16BIT, i.e. LVDS signaling is not supported with an 8-bit SDDO interface. Note that for LVDS_MODE, DDR_MODE must also be set.
5 SCAN_DIR_1	Determines scan direction for each half of the TFT panel 0x1 **UP** — Scan this region from bottom to top 0x0 **DOWN** — Scan this region from top to bottom
4 SCAN_DIR_0	Determines scan direction for each half of the TFT panel 0x1 **UP** — Scan this region from bottom to top 0x0 **DOWN** — Scan this region from top to bottom
3 DUAL_SCAN	Enables dual scan-mode. Note in this mode, SDDO_BUS_FORMAT=16BIT must be selected. and PIXELS_PER_CLK applies to each 8-bit segment of the SDDO bus.
2 SDDO_WIDTH	Selects either 8 or 16 bit SDDO bus format 0x0 **8BIT** — Connect to 8-bit source driver 0x1 **16BIT** — Connct to 16-bit source driver
1–0 PIXELS_PER_SDCLK	Number of TFT pixels per SDCLK period. Note that this value forms the division of the PIXLK to generate the SDCLK such that SDCLK = PIXCLK/PIXELS_PER_SDCLK. For dual-scan mode (DUAL_SCAN=1), this applies to 8-bit half of the 16-bit SDDO. It should be noted that when DDR_MODE is enabled, both edges of the clock have to be accounted for in this value, so for example with an 8-bit SDDO, 2-bit TFT pixel, this field should be set to EIGHT (four pixels on the pos-edge and 4-pixels on the neg-edge) 0x0 **RESERVED** — Reserved 0x1 **TWO** — Two TFT-pixels per SDCLK 0x2 **FOUR** — Four TFT-pixels per SDCLK 0x3 **EIGHT** — Eight TFT-pixels per SDCLK

24.4.14 EPDC Timing Control Engine Source-Driver Config Register (EPDC_TCE_SDCFG*n*)

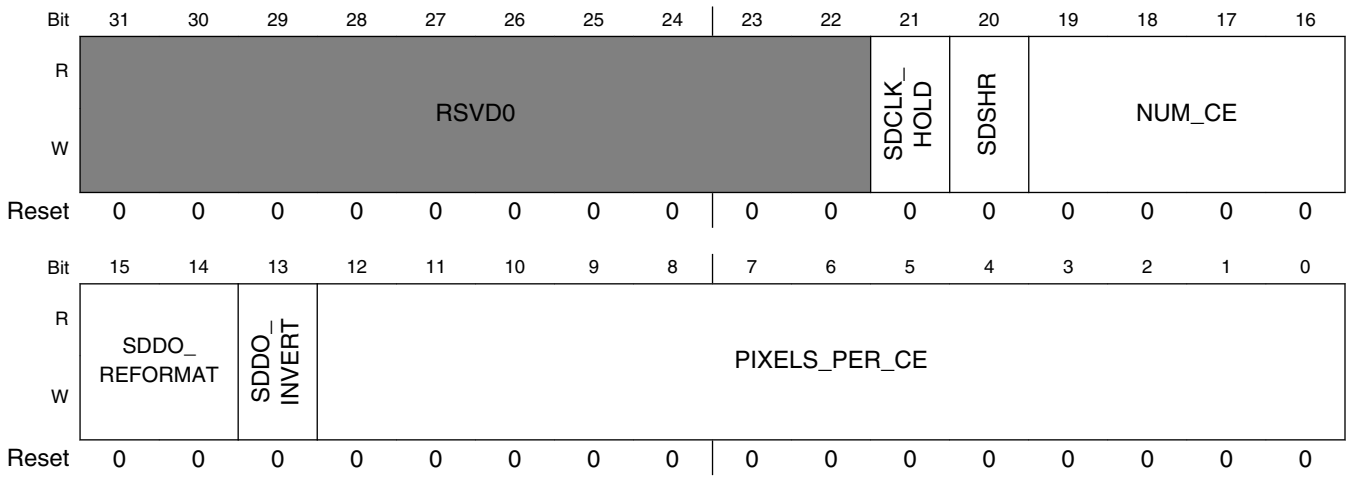
EPDC Timing Control Engine Source-Driver Config Register houses source-driver configuration.

Addresses: EPDC_TCE_SDCFG is 4101_0000h base + 220h offset = 4101_0220h

EPDC_TCE_SDCFG_SET is 4101_0000h base + 224h offset = 4101_0224h

EPDC_TCE_SDCFG_CLR is 4101_0000h base + 228h offset = 4101_0228h

EPDC_TCE_SDCFG_TOG is 4101_0000h base + 22Ch offset = 4101_022Ch



EPDC_TCE_SDCFG*n* field descriptions

Field	Description
31–22 RSVD0	Reserved.
21 SDCLK_HOLD	Setting this bit to 1 holds the SDCLK low during LINE_BEGIN
20 SDSHR	Value for source-driver shift direction output port
19–16 NUM_CE	Number of source driver IC chip-enables. Must be 1-10
15–14 SDDO_ REFORMAT	This register defines the various re-formatting options to enable more flexibility in the source-driver interface: 0x0 **STANDARD** — No change. 0x1 **FLIP_PIXELS** — Reverses the order of the pixels on SDDO. This register setting is sensitive to the TFT pixel width (TFT_PIXEL_FORMAT), e.g. for TFT_PIXEL_FORMAT=2B on an 8-bit bus P3,P2,P1,P0 becomes P0,P1,P2,P3, whereas with TFT_PIXEL_FORMAT=4B, on an 8-bit bus, P1,P0 becomes P0,P1
13 SDDO_INVERT	Setting this bit to 1 reverses the polarity of each SDDO bit so 0xAAAA in 16-bit mode for example becomes 0x5555. This setting can be made in addition to the SDDO_REFORMAT register setting.

Table continues on the next page...

EPDC_TCE_SDCFG*n* field descriptions (continued)

Field	Description
12–0 PIXELS_PER_CE	Number of pixels (outputs) per source-driver IC. Please note that EPDC_RES[HORIZONTAL] must be an integer multiple of PINS_PER_CE.

24.4.15 EPDC Timing Control Engine Gate-Driver Config Register (EPDC_TCE_GDCFG*n*)

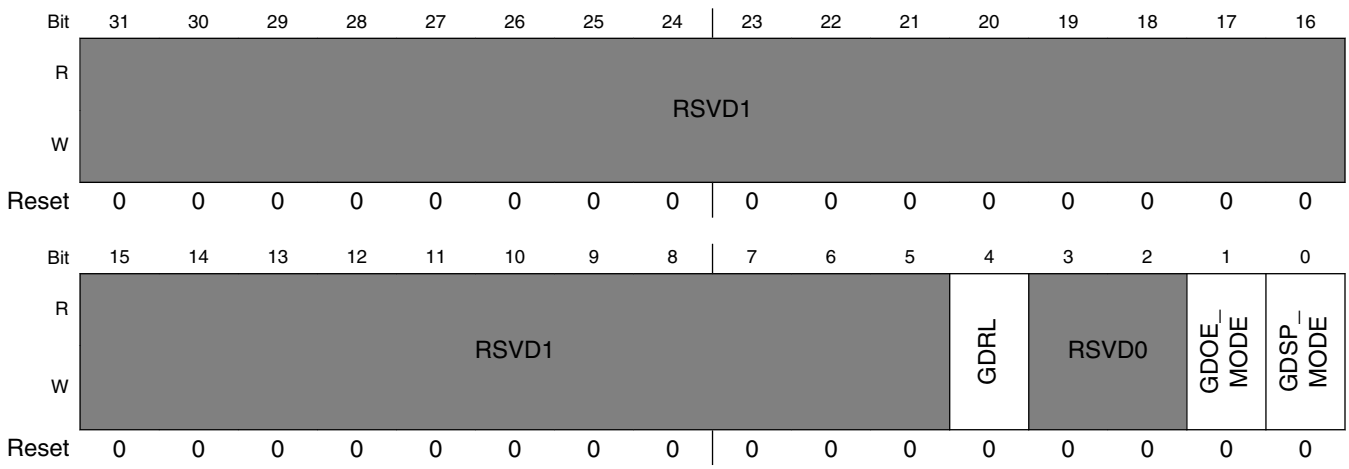
EPDC Timing Control Engine Gate-Driver Config Register houses gate-driver configuration.

Addresses: EPDC_TCE_GDCFG is 4101_0000h base + 240h offset = 4101_0240h

EPDC_TCE_GDCFG_SET is 4101_0000h base + 244h offset = 4101_0244h

EPDC_TCE_GDCFG_CLR is 4101_0000h base + 248h offset = 4101_0248h

EPDC_TCE_GDCFG_TOG is 4101_0000h base + 24Ch offset = 4101_024Ch



EPDC_TCE_GDCFG*n* field descriptions

Field	Description
31–5 RSVD1	Reserved.
4 GDRL	Value for gate-driver right/left shift output port
3–2 RSVD0	Reserved.
1 GDOE_MODE	Selects method for driving GDOE signal. When set to 0, GDOE is driven at all times during the frame-scan except FRAME_SYNC. When set to 1, GDOE is driven as a delayed version of GDCLK delayed by EPDC_TCE_TIMING3[GDOE_OFFSET].
0 GDSP_MODE	Selects method for driving GDSP pulse. When set to 0, GDSP is is always fixed to have a pulse width of one line-time. When set to 1, GDSP has a pulse-width determined by the FRAME_SYNC setting. Note that GDSP_MODE=1 is not compatible with the GDSP_OFFSET function

24.4.16 EPDC Timing Control Engine Horizontal Timing Register 1 (EPDC_TCE_HSCAN1*n*)

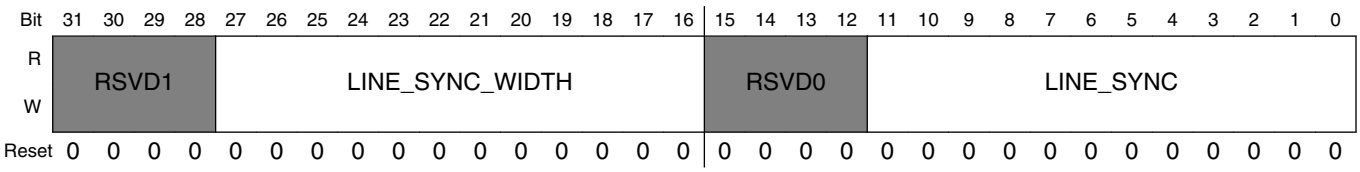
Horizontal scan timing registers. All timing values are expressed in terms of the EPDC's internal PIXCLK, which depending on the PIXELS_PER_SDCLK register setting is either 2:1 or 4:1. Note that line data length is derived from EPDC_RES.

Addresses: EPDC_TCE_HSCAN1 is 4101_0000h base + 260h offset = 4101_0260h

EPDC_TCE_HSCAN1_SET is 4101_0000h base + 264h offset = 4101_0264h

EPDC_TCE_HSCAN1_CLR is 4101_0000h base + 268h offset = 4101_0268h

EPDC_TCE_HSCAN1_TOG is 4101_0000h base + 26Ch offset = 4101_026Ch



EPDC_TCE_HSCAN1*n* field descriptions

Field	Description
31–28 RSVD1	Reserved.
27–16 LINE_SYNC_ WIDTH	Number of PIXCLK cycles for the SDLE active time. Note that this value cannot be larger than LINE_SYNC and must be greater than 0. Typically it is recommended to set this value to be the same as LINE_SYNC
15–12 RSVD0	Reserved.
11–0 LINE_SYNC	Number of PIXCLK cycles for line sync duration. Note that this value encompasses the LINE_SYNC_WIDTH duration. This value must be programmed to a multiple of SDCLK cycles

24.4.17 EPDC Timing Control Engine Horizontal Timing Register 2 (EPDC_TCE_HSCAN2*n*)

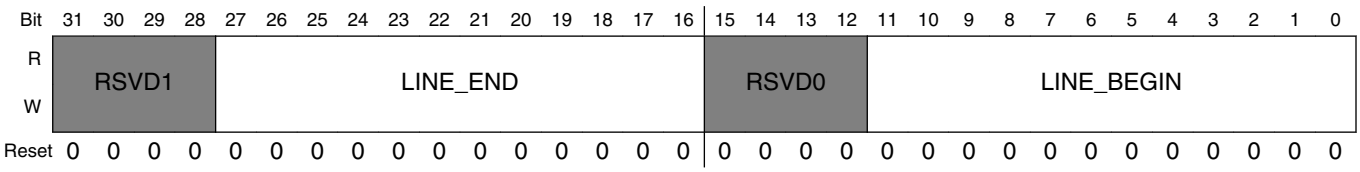
Horizontal scan timing registers. All timing values are expressed in terms of the EPDC's internal PIXCLK, which depending on the PIXELS_PER_SDCLK register setting is either 2:1 or 4:1. Note that line data length is derived from EPDC_RES.

Addresses: EPDC_TCE_HSCAN2 is 4101_0000h base + 280h offset = 4101_0280h

EPDC_TCE_HSCAN2_SET is 4101_0000h base + 284h offset = 4101_0284h

EPDC_TCE_HSCAN2_CLR is 4101_0000h base + 288h offset = 4101_0288h

EPDC_TCE_HSCAN2_TOG is 4101_0000h base + 28Ch offset = 4101_028Ch



EPDC_TCE_HSCAN2*n* field descriptions

Field	Description
31–28 RSVD1	Reserved.
27–16 LINE_END	Number of PIXCLK cycles for line end duration. This defines the duration from the de-assertion of SDCE and assertion of the next SDLE.
15–12 RSVD0	Reserved.
11–0 LINE_BEGIN	Number of PIXCLK cycles for line begin duration. This defines the interval between de-assertion of SDLE and assertion of the SDCE signals. This value must be programmed to a multiple of SDCLK cycles

24.4.18 EPDC Timing Control Engine Vertical Timing Register (EPDC_TCE_VSCAN*n*)

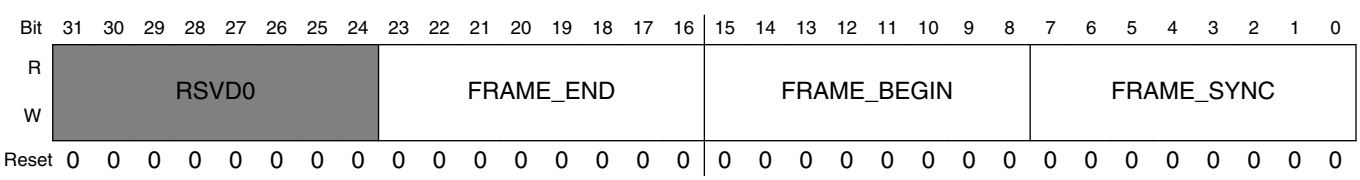
This register houses vertical scan timing. Note that frame data length is derived from EPDC_RES.

Addresses: EPDC_TCE_VSCAN is 4101_0000h base + 2A0h offset = 4101_02A0h

EPDC_TCE_VSCAN_SET is 4101_0000h base + 2A4h offset = 4101_02A4h

EPDC_TCE_VSCAN_CLR is 4101_0000h base + 2A8h offset = 4101_02A8h

EPDC_TCE_VSCAN_TOG is 4101_0000h base + 2ACh offset = 4101_02ACh



EPDC_TCE_VSCAN*n* field descriptions

Field	Description
31–24 RSVD0	Reserved.
23–16 FRAME_END	Number of lines for frame end duration.
15–8 FRAME_BEGIN	Number of lines for frame begin duration.
7–0 FRAME_SYNC	Number of lines for frame sync duration.

24.4.19 EPDC Timing Control Engine OE timing control Register (EPDC_TCE_OE*n*)

This register contains delay programming values for the SDOEZ and SDOED source driver control signals.

Addresses: EPDC_TCE_OE is 4101_0000h base + 2C0h offset = 4101_02C0h

EPDC_TCE_OE_SET is 4101_0000h base + 2C4h offset = 4101_02C4h

EPDC_TCE_OE_CLR is 4101_0000h base + 2C8h offset = 4101_02C8h

EPDC_TCE_OE_TOG is 4101_0000h base + 2CCh offset = 4101_02CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SDOED_WIDTH								SDOED_DLY								SDOEZ_WIDTH								SDOEZ_DLY							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPDC_TCE_OE*n* field descriptions

Field	Description
31–24 SDOED_WIDTH	Number of PIXCLK cycles from SDOED high to SDOED falling (Must be greater than 0)
23–16 SDOED_DLY	Number of PIXCLK cycles from SDOEZ low to SDOED rising (Must be greater than 0)
15–8 SDOEZ_WIDTH	Number of PIXCLK cycles from SDOEZ high to SDOEZ falling (Must be greater than 0)
7–0 SDOEZ_DLY	Number of PIXCLK cycles from SDLE falling edge to SDOEZ rising (Must be greater than 0)

24.4.20 EPDC Timing Control Engine Driver Polarity Register (EPDC_TCE_POLARITY*n*)

This register allows for programming the polarity of source/gate driver control signals. This register houses FIFO control bits.

Addresses: EPDC_TCE_POLARITY is 4101_0000h base + 2E0h offset = 4101_02E0h
EPDC_TCE_POLARITY_SET is 4101_0000h base + 2E4h offset = 4101_02E4h
EPDC_TCE_POLARITY_CLR is 4101_0000h base + 2E8h offset = 4101_02E8h
EPDC_TCE_POLARITY_TOG is 4101_0000h base + 2ECh offset = 4101_02ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSVD0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD0											GDSP_POL	GDOE_POL	SDOE_POL	SDLE_POL	SDCE_POL
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0

EPDC_TCE_POLARITY*n* field descriptions

Field	Description
31–5 RSVD0	Reserved.
4 GDSP_POL	0 = Active Low, 1 = Active High. Applies to the GDSP output
3 GDOE_POL	0 = Active Low, 1 = Active High. Applies to the GDOE output
2 SDOE_POL	0 = Active Low, 1 = Active High. Applies to the SDOE. Does not apply to SDOEZ and SDOED outputs
1 SDLE_POL	0 = Active Low, 1 = Active High. Applies to the SDLE output
0 SDCE_POL	0 = Active Low, 1 = Active High. Applies to all 10 SDCE outputs

24.4.21 EPDC Timing Control Engine Timing Register 1 (EPDC_TCE_TIMING1*n*)

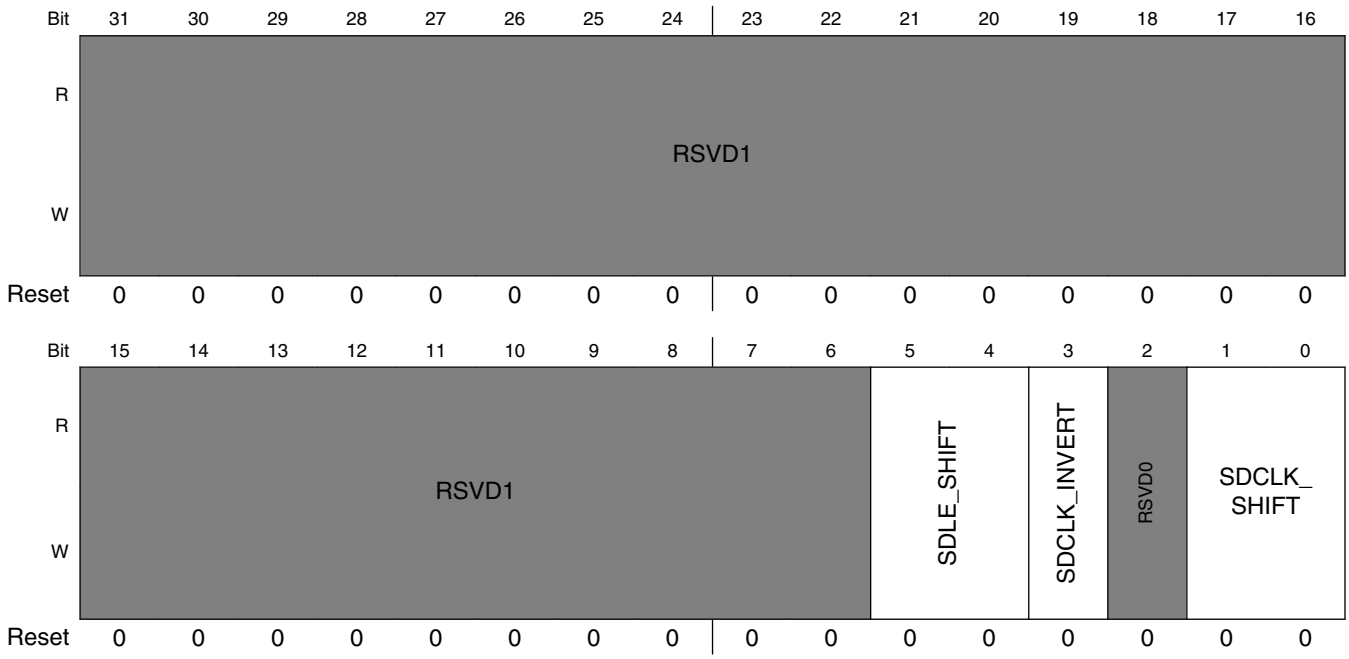
This register contains various timing adjustment controls.

Addresses: EPDC_TCE_TIMING1 is 4101_0000h base + 300h offset = 4101_0300h

EPDC_TCE_TIMING1_SET is 4101_0000h base + 304h offset = 4101_0304h

EPDC_TCE_TIMING1_CLR is 4101_0000h base + 308h offset = 4101_0308h

EPDC_TCE_TIMING1_TOG is 4101_0000h base + 30Ch offset = 4101_030Ch



EPDC_TCE_TIMING1*n* field descriptions

Field	Description
31–6 RSVD1	Reserved.
5–4 SDLE_SHIFT	This register can be used to implement additional timing setup/hold adjustment of source driver signals by adjusting the SDCLK up to 3 PIXCLK cycles 0x0 **NONE** — No shift of SDLE 0x1 **ONE** — Shift SDLE 1 pixclk cycle 0x2 **TWO** — Shift SDLE 2 pixclk cycles 0x3 **THREE** — Shift SDLE 3 pixclk cycles
3 SDCLK_INVERT	Invert phase of SDCLK
2 RSVD0	Reserved.
1–0 SDCLK_SHIFT	This register can be used to implement additional timing setup/hold adjustment of source driver signals by adjusting the SDCLK up to 4 cycles

Table continues on the next page...

EPDC_TCE_TIMING1*n* field descriptions (continued)

Field	Description
	0x0 **NONE** — No shift of SDCLK 0x1 **ONE** — Shift SDCLK 1 pixclk cycle 0x2 **TWO** — Shift SDCLK 2 pixclk cycles 0x3 **THREE** — Shift SDCLK 3 pixclk cycles

24.4.22 EPDC Timing Control Engine Timing Register 2 (EPDC_TCE_TIMING2*n*)

This register contains various timing adjustment controls and houses general purpose timing adjustment registers.

Addresses: EPDC_TCE_TIMING2 is 4101_0000h base + 310h offset = 4101_0310h
EPDC_TCE_TIMING2_SET is 4101_0000h base + 314h offset = 4101_0314h
EPDC_TCE_TIMING2_CLR is 4101_0000h base + 318h offset = 4101_0318h
EPDC_TCE_TIMING2_TOG is 4101_0000h base + 31Ch offset = 4101_031Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GDCLK_HP																GDSP_OFFSET															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

EPDC_TCE_TIMING2*n* field descriptions

Field	Description
31–16 GDCLK_HP	This register controls the GDCLK high-pulse width. It is expressed by N PIXCLKs where N=1 to 65535. Note that GDCLK will always have a period equal to the line-clock timing. A value of 0 is not supported. It is recommended that this value be set to at least a half line-clock time. For panels which use GDCLK to drive GDOE, this high-pulse width should be set to cover tha majority of the line timing
15–0 GDSP_OFFSET	This register allows the user to shift the GDSP pulse by N PIXCLKs where N=1 to 65535. Note that GDSP will always have a pulse width equivalent to the line-clock timing. A value of 0 is not supported.

24.4.23 EPDC Timing Control Engine Timing Register 3 (EPDC_TCE_TIMING3*n*)

This register contains various timing adjustment controls and houses general purpose timing adjustment registers.

Addresses: EPDC_TCE_TIMING3 is 4101_0000h base + 320h offset = 4101_0320h
EPDC_TCE_TIMING3_SET is 4101_0000h base + 324h offset = 4101_0324h
EPDC_TCE_TIMING3_CLR is 4101_0000h base + 328h offset = 4101_0328h
EPDC_TCE_TIMING3_TOG is 4101_0000h base + 32Ch offset = 4101_032Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GDOE_OFFSET																GDCLK_OFFSET															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

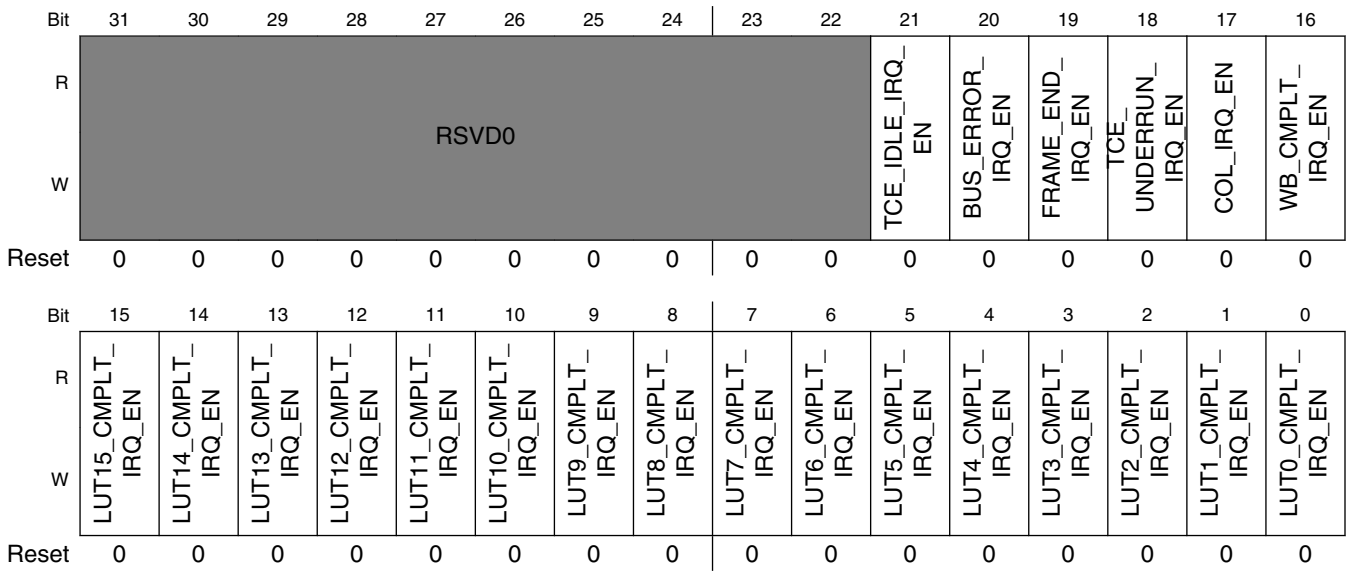
EPDC_TCE_TIMING3*n* field descriptions

Field	Description
31–16 GDOE_OFFSET	When using GDOE_MODE=1, this register sets the delay from GDCLK to the GDOE in terms of N PIXCLK cycles
15–0 GDCLK_ OFFSET	This register allows the user to shift the GDCLK from the line time by N PIXCLK cycles.

24.4.24 EPDC IRQ Mask Register (EPDC_IRQ_MASK*n*)

This register controls IRQ masking for all EPDC interrupts.

Addresses: EPDC_IRQ_MASK is 4101_0000h base + 400h offset = 4101_0400h
EPDC_IRQ_MASK_SET is 4101_0000h base + 404h offset = 4101_0404h
EPDC_IRQ_MASK_CLR is 4101_0000h base + 408h offset = 4101_0408h
EPDC_IRQ_MASK_TOG is 4101_0000h base + 40Ch offset = 4101_040Ch



EPDC_IRQ_MASK*n* field descriptions

Field	Description
31–22 RSVD0	Reserved.
21 TCE_IDLE_IRQ_EN	Enable TCE Idle interrupt detection.
20 BUS_ERROR_IRQ_EN	Enable AXI BUS ERROR interrupt detection.
19 FRAME_END_IRQ_EN	If this bit is set, EPDC will assert the current frame end interrupt. This irq is only available during updating period.
18 TCE_UNDERRUN_IRQ_EN	Enable pixel FIFO under-run condition detection.
17 COL_IRQ_EN	Enable collision detection interrupts for all LUTs

Table continues on the next page...

EPDC_IRQ_MASK*n* field descriptions (continued)

Field	Description
16 WB_CMPLT_ IRQ_EN	Enable WB complete interrupt
15 LUT15_CMPLT_ IRQ_EN	LUT15 Complete Interrupt Enable
14 LUT14_CMPLT_ IRQ_EN	LUT14 Complete Interrupt Enable
13 LUT13_CMPLT_ IRQ_EN	LUT13 Complete Interrupt Enable
12 LUT12_CMPLT_ IRQ_EN	LUT12 Complete Interrupt Enable
11 LUT11_CMPLT_ IRQ_EN	LUT11 Complete Interrupt Enable
10 LUT10_CMPLT_ IRQ_EN	LUT10 Complete Interrupt Enable
9 LUT9_CMPLT_ IRQ_EN	LUT9 Complete Interrupt Enable
8 LUT8_CMPLT_ IRQ_EN	LUT8 Complete Interrupt Enable
7 LUT7_CMPLT_ IRQ_EN	LUT7 Complete Interrupt Enable
6 LUT6_CMPLT_ IRQ_EN	LUT6 Complete Interrupt Enable
5 LUT5_CMPLT_ IRQ_EN	LUT5 Complete Interrupt Enable
4 LUT4_CMPLT_ IRQ_EN	LUT4 Complete Interrupt Enable
3 LUT3_CMPLT_ IRQ_EN	LUT3 Complete Interrupt Enable
2 LUT2_CMPLT_ IRQ_EN	LUT2 Complete Interrupt Enable

Table continues on the next page...

EPDC_IRQ_MASK*n* field descriptions (continued)

Field	Description
1 LUT1_CMPLT_IRQ_EN	LUT1 Complete Interrupt Enable
0 LUT0_CMPLT_IRQ_EN	LUT0 Complete Interrupt Enable

24.4.25 EPDC Interrupt Register (EPDC_IRQ*n*)

This register houses the interrupt bits for the LUT Completions. The IRQ for a specific LUT is triggered when its corrrsponding physical update is competed on the screen. Each interrupt has a corresponding mask register in EPDC_IRQ_MASK.

Addresses: EPDC_IRQ is 4101_0000h base + 420h offset = 4101_0420h

EPDC_IRQ_SET is 4101_0000h base + 424h offset = 4101_0424h

EPDC_IRQ_CLR is 4101_0000h base + 428h offset = 4101_0428h

EPDC_IRQ_TOG is 4101_0000h base + 42Ch offset = 4101_042Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSVD0										TCE_IDLE_IRQ	BUS_ERROR_IRQ	FRAME_END_IRQ	TCE_UNDERRUN_IRQ	LUT_COL_IRQ	WB_CMPLT_IRQ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LUT15_CMPLT_IRQ	LUT14_CMPLT_IRQ	LUT13_CMPLT_IRQ	LUT12_CMPLT_IRQ	LUT11_CMPLT_IRQ	LUT10_CMPLT_IRQ	LUT9_CMPLT_IRQ	LUT8_CMPLT_IRQ	LUT7_CMPLT_IRQ	LUT6_CMPLT_IRQ	LUT5_CMPLT_IRQ	LUT4_CMPLT_IRQ	LUT3_CMPLT_IRQ	LUT2_CMPLT_IRQ	LUT1_CMPLT_IRQ	LUT0_CMPLT_IRQ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPDC_IRQ*n* field descriptions

Field	Description
31–22 RSVD0	Reserved.
21 TCE_IDLE_IRQ	Interrupt to indicate that the TCE has completed TFT frame scans and is in an idle state.

Table continues on the next page...

EPDC_IRQ*n* field descriptions (continued)

Field	Description
20 BUS_ERROR_ IRQ	Interrupt to indicate AXI BUS error occurs.
19 FRAME_END_ IRQ	Interrupt to indicate EPDC has completed the current frame and is in the vertical blanking period.
18 TCE_ UNDERRUN_ IRQ	Interrupt to signify that a pixel FIFO under-run has occured.
17 LUT_COL_IRQ	Collision detection interrupt. Check EPDC_STATUS_COL.
16 WB_CMPLT_ IRQ	Working buffer process complete Interrupt
15 LUT15_CMPLT_ IRQ	LUT15 Complete Interrupt
14 LUT14_CMPLT_ IRQ	LUT14 Complete Interrupt
13 LUT13_CMPLT_ IRQ	LUT13 Complete Interrupt
12 LUT12_CMPLT_ IRQ	LUT12 Complete Interrupt
11 LUT11_CMPLT_ IRQ	LUT11 Complete Interrupt
10 LUT10_CMPLT_ IRQ	LUT10 Complete Interrupt
9 LUT9_CMPLT_ IRQ	LUT9 Complete Interrupt
8 LUT8_CMPLT_ IRQ	LUT8 Complete Interrupt
7 LUT7_CMPLT_ IRQ	LUT7 Complete Interrupt
6 LUT6_CMPLT_ IRQ	LUT6 Complete Interrupt

Table continues on the next page...

EPDC_IRQ*n* field descriptions (continued)

Field	Description
5 LUT5_CMPLT_IRQ	LUT5 Complete Interrupt
4 LUT4_CMPLT_IRQ	LUT4 Complete Interrupt
3 LUT3_CMPLT_IRQ	LUT3 Complete Interrupt
2 LUT2_CMPLT_IRQ	LUT2 Complete Interrupt
1 LUT1_CMPLT_IRQ	LUT1 Complete Interrupt
0 LUT0_CMPLT_IRQ	LUT0 Complete Interrupt

24.4.26 EPDC Status Register - LUTs (EPDC_EPDC_STATUS_LUTS*n*)

The EPDC Status Register provides a read-only view to various input conditions and internal states.

Addresses: EPDC_EPDC_STATUS_LUTS is 4101_0000h base + 440h offset = 4101_0440h
EPDC_EPDC_STATUS_LUTS_SET is 4101_0000h base + 444h offset = 4101_0444h
EPDC_EPDC_STATUS_LUTS_CLR is 4101_0000h base + 448h offset = 4101_0448h
EPDC_EPDC_STATUS_LUTS_TOG is 4101_0000h base + 44Ch offset = 4101_044Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LUT15_STS	LUT14_STS	LUT13_STS	LUT12_STS	LUT11_STS	LUT10_STS	LUT9_STS	LUT8_STS	LUT7_STS	LUT6_STS	LUT5_STS	LUT4_STS	LUT3_STS	LUT2_STS	LUT1_STS	LUT0_STS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

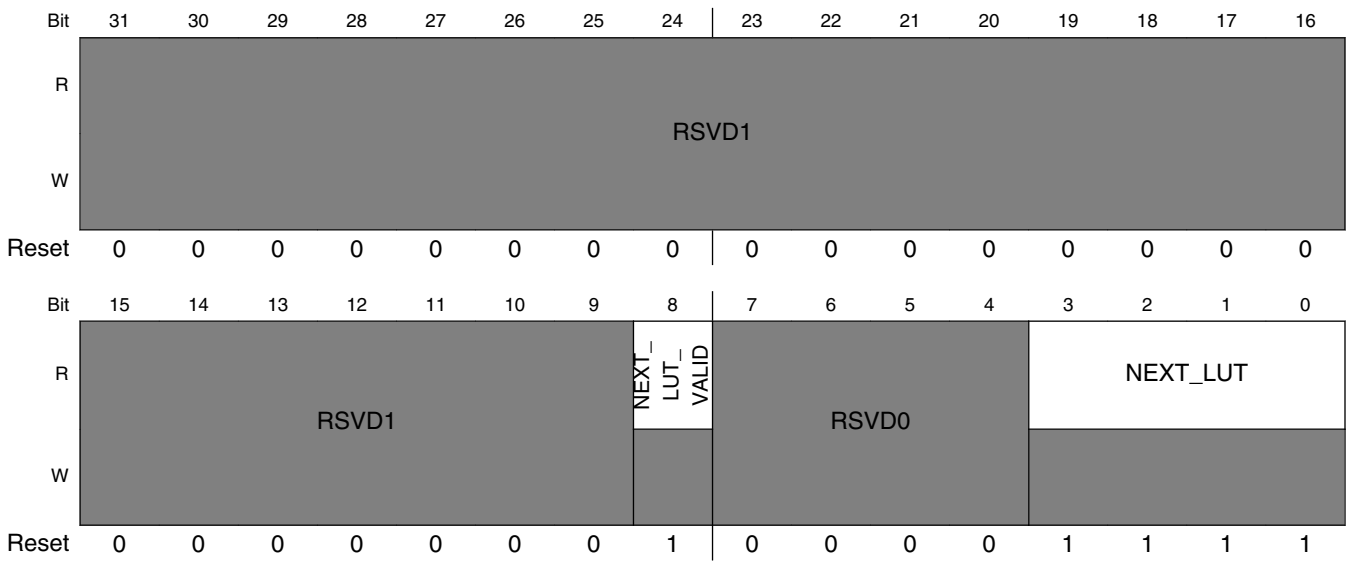
EPDC_EPDC_STATUS_LUTS*n* field descriptions

Field	Description
31–16 RSVD0	Reserved.
15 LUT15_STS	LUT15 Status : 1=ACTIVE, 0=IDLE
14 LUT14_STS	LUT14 Status : 1=ACTIVE, 0=IDLE
13 LUT13_STS	LUT13 Status : 1=ACTIVE, 0=IDLE
12 LUT12_STS	LUT12 Status : 1=ACTIVE, 0=IDLE
11 LUT11_STS	LUT11 Status : 1=ACTIVE, 0=IDLE
10 LUT10_STS	LUT10 Status : 1=ACTIVE, 0=IDLE
9 LUT9_STS	LUT9 Status : 1=ACTIVE, 0=IDLE
8 LUT8_STS	LUT8 Status : 1=ACTIVE, 0=IDLE
7 LUT7_STS	LUT7 Status : 1=ACTIVE, 0=IDLE
6 LUT6_STS	LUT6 Status : 1=ACTIVE, 0=IDLE
5 LUT5_STS	LUT5 Status : 1=ACTIVE, 0=IDLE
4 LUT4_STS	LUT4 Status : 1=ACTIVE, 0=IDLE
3 LUT3_STS	LUT3 Status : 1=ACTIVE, 0=IDLE
2 LUT2_STS	LUT2 Status : 1=ACTIVE, 0=IDLE
1 LUT1_STS	LUT1 Status : 1=ACTIVE, 0=IDLE
0 LUT0_STS	LUT0 Status : 1=ACTIVE, 0=IDLE

24.4.27 EPDC Status Register - Next Available LUT (EPDC_STATUS_NEXTLUT)

The EPDC Status Register provides a read-only view to various input conditions and internal states. It holds the value of next available LUT and can be used for fast LUT assignment. This value can be read and then used in an update command as part of the EPDC_UPD_CTRL register write.

Address: EPDC_STATUS_NEXTLUT is 4101_0000h base + 460h offset = 4101_0460h



EPDC_STATUS_NEXTLUT field descriptions

Field	Description
31–9 RSVD1	Reserved.
8 NEXT_LUT_ VALID	This bitfield can be used to check against a LUTs full condition.
7–4 RSVD0	Reserved.
3–0 NEXT_LUT	Next available LUT value

24.4.28 EPDC LUT Collision Status (EPDC_STATUS_COL*n*)

EPDC LUT Collision Status Register and EPDC_IRQ[WB_CMPLT_IRQ] works in conjuction with EPDC_IRQ[LUT_COL_IRQ]. When a collision occurs, the interrupt is set and all status bits are set for LUTs which were touched by the collision. It does not set the bit for the LUT that caused the collision. There is a single interrupt mask which is used to control all the IRQ bits in this register (in EPDC_IRQ_MASK).

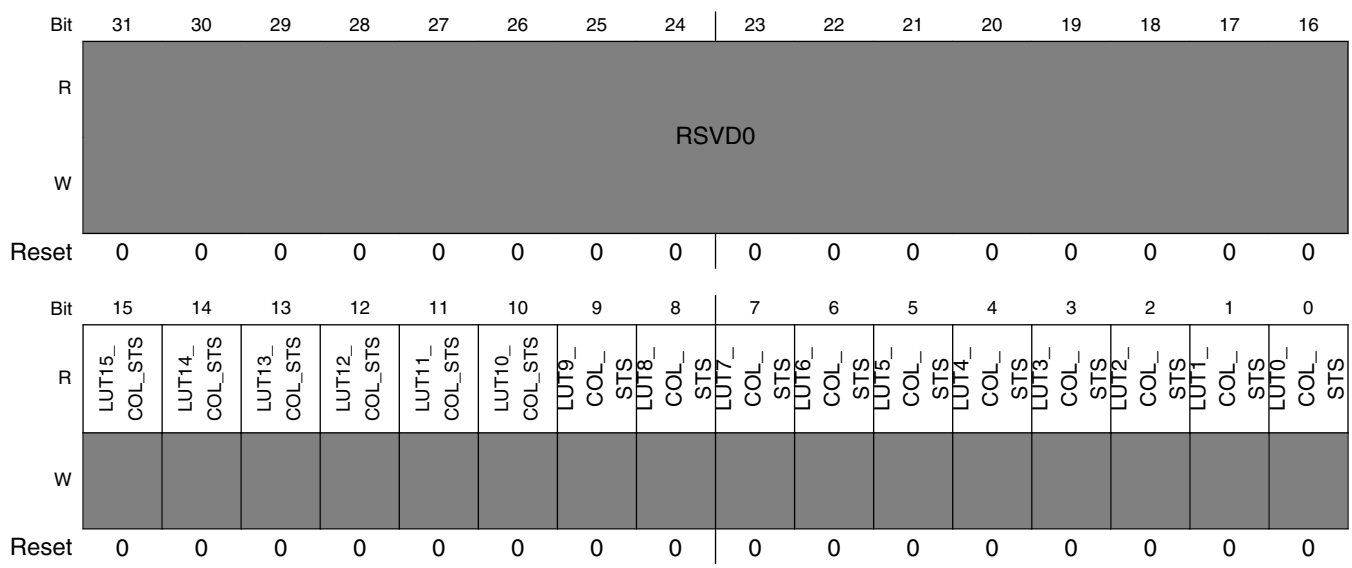
NOTE

A collision caused by a LUT which was set-up for no collision detection (according to EPDC_IRQ_MASKn[COL_IRQ_EN]) will not trigger any collision LUT IRQ or status update.

NOTE

Clearing the interrupt bit EPDC_IRQ[LUT_COL_IRQ] clears this register.

Addresses: EPDC_STATUS_COL is 4101_0000h base + 480h offset = 4101_0480h
EPDC_STATUS_COL_SET is 4101_0000h base + 484h offset = 4101_0484h
EPDC_STATUS_COL_CLR is 4101_0000h base + 488h offset = 4101_0488h
EPDC_STATUS_COL_TOG is 4101_0000h base + 48Ch offset = 4101_048Ch



EPDC_STATUS_COL*n* field descriptions

Field	Description
31–16 RSVD0	Reserved.

Table continues on the next page...

EPDC_STATUS_COL*n* field descriptions (continued)

Field	Description
15 LUT15_COL_STS	LUT15 Collision Status
14 LUT14_COL_STS	LUT14 Collision Status
13 LUT13_COL_STS	LUT13 Collision Status
12 LUT12_COL_STS	LUT12 Collision Status
11 LUT11_COL_STS	LUT11 Collision Status
10 LUT10_COL_STS	LUT10 Collision Status
9 LUT9_COL_STS	LUT9 Collision Status
8 LUT8_COL_STS	LUT8 Collision Status
7 LUT7_COL_STS	LUT7 Collision Status
6 LUT6_COL_STS	LUT6 Collision Status
5 LUT5_COL_STS	LUT5 Collision Status
4 LUT4_COL_STS	LUT4 Collision Status
3 LUT3_COL_STS	LUT3 Collision Status
2 LUT2_COL_STS	LUT2 Collision Status
1 LUT1_COL_STS	LUT1 Collision Status
0 LUT0_COL_STS	LUT0 Collision Status

24.4.29 EPDC General Status Register (EPDC_STATUS*n*)

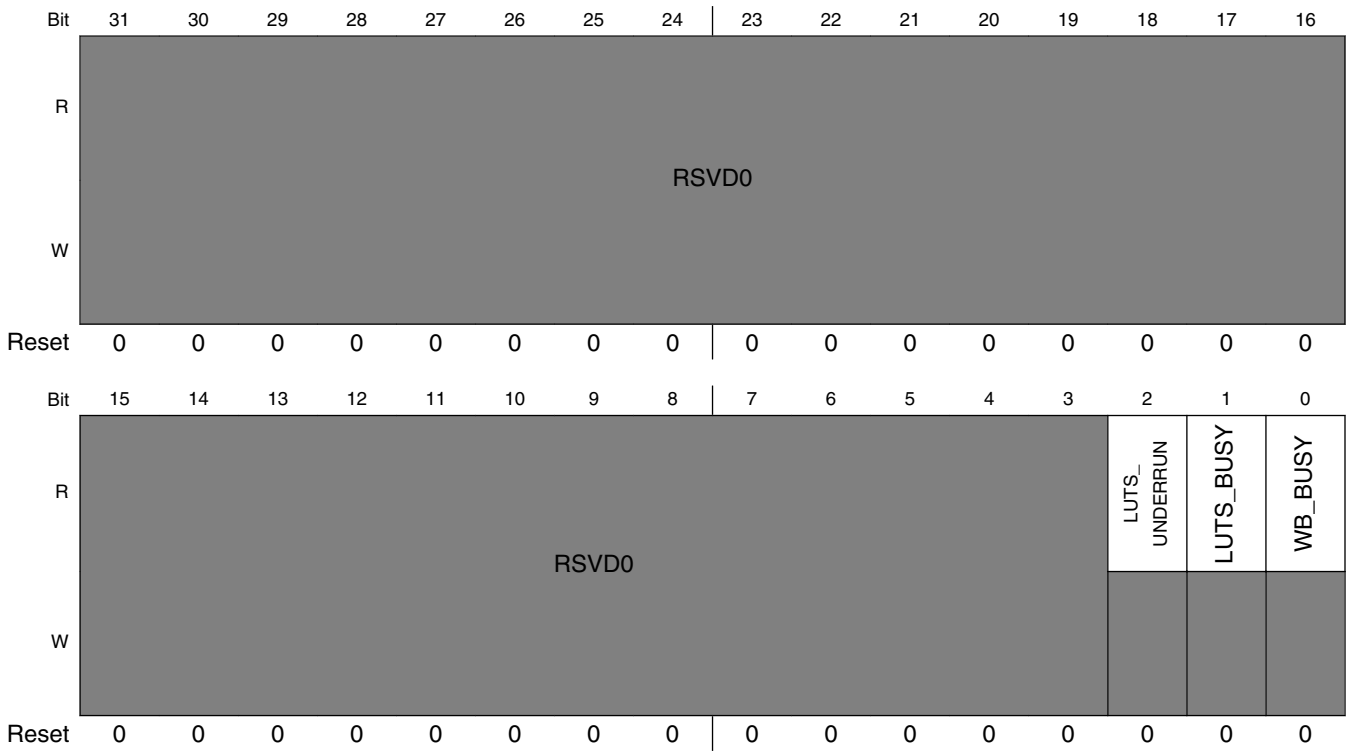
This register houses non-LUT-specific status bits.

Addresses: EPDC_STATUS is 4101_0000h base + 4A0h offset = 4101_04A0h

EPDC_STATUS_SET is 4101_0000h base + 4A4h offset = 4101_04A4h

EPDC_STATUS_CLR is 4101_0000h base + 4A8h offset = 4101_04A8h

EPDC_STATUS_TOG is 4101_0000h base + 4ACh offset = 4101_04ACh



EPDC_STATUS*n* field descriptions

Field	Description
31–3 RSVD0	Reserved.
2 LUTS_ UNDERRUN	Provides a summary status of LUT fill. 1= not enough time for active luts read during blanking period before vscan_holdoff. 0=complete all active luts fill during blanking period before VSCAN_HOLDOFF.
1 LUTS_BUSY	Provides a summary status of LUTs. 1= All LUTs are busy, 0= LUTs are available
0 WB_BUSY	Working buffer process is busy cannot accept new update requests. When WB_BUSY is 1, software should wait for the WB_CMPLT_IRQ interrupt. When this interrupt occurs WB_BUSY is cleared immediately. This is a real-time status of the process.

24.4.30 EPDC Debug register (EPDC_DEBUG*n*)

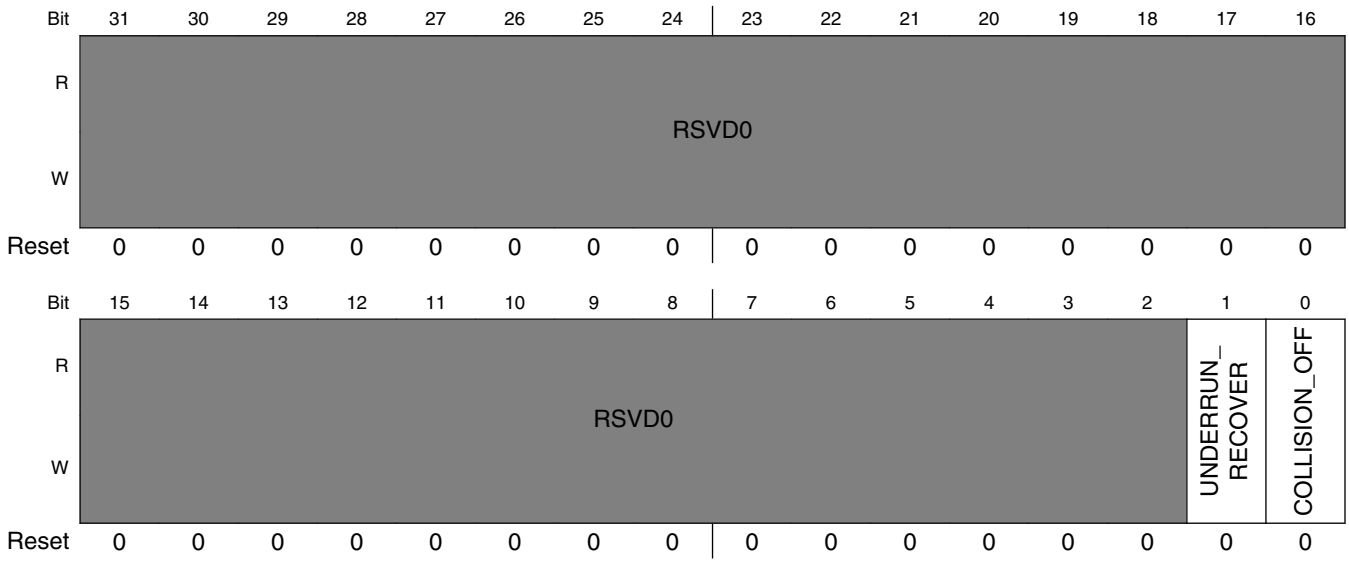
This register is for debug/testing purposes only. Writing one to this register may damage the display panel.

Addresses: EPDC_DEBUG is 4101_0000h base + 500h offset = 4101_0500h

EPDC_DEBUG_SET is 4101_0000h base + 504h offset = 4101_0504h

EPDC_DEBUG_CLR is 4101_0000h base + 508h offset = 4101_0508h

EPDC_DEBUG_TOG is 4101_0000h base + 50Ch offset = 4101_050Ch



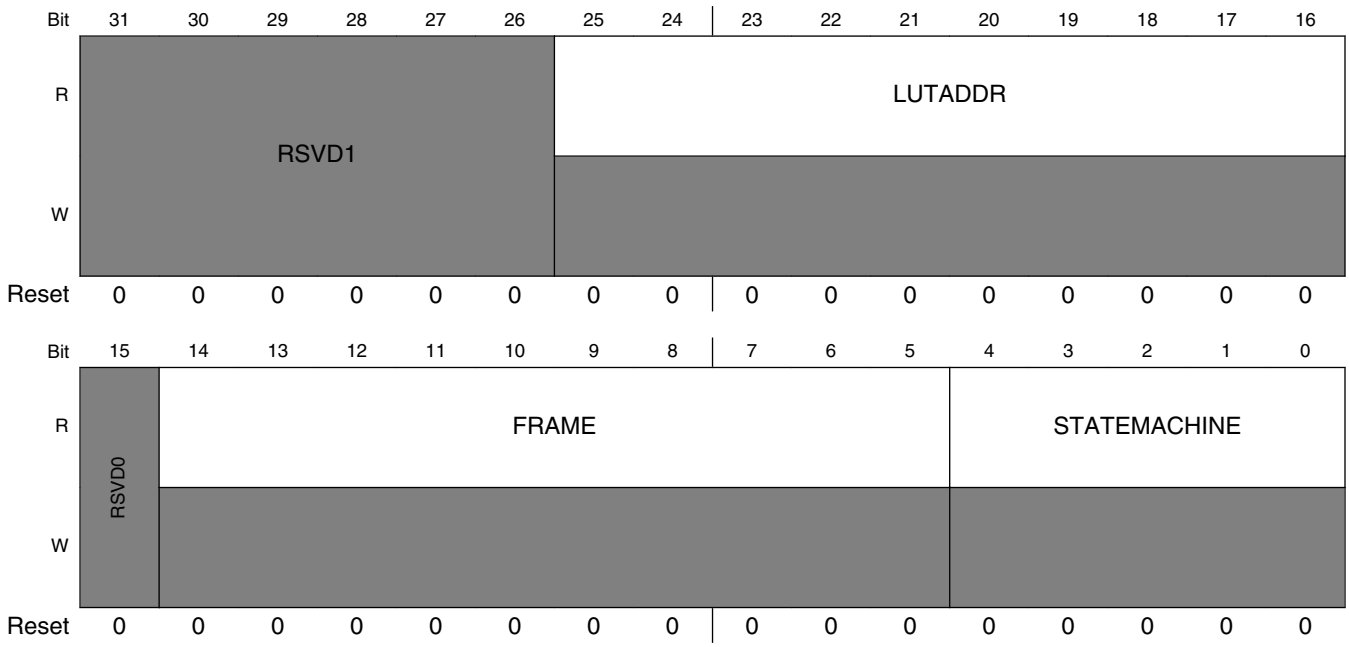
EPDC_DEBUG*n* field descriptions

Field	Description
31–2 RSVD0	Reserved.
1 UNDERRUN_ RECOVER	Set 1 to enable EPDC to recover the display at the next vsync when display fifo underrun happens.
0 COLLISION_ OFF	Set 1 to allow update to take effect even when collision is detected.

24.4.31 EPDC LUTn Debug Information register (EPDC_DEBUG_LUT*n*)

This register gives debug visibility for LUTn and houses LUTn debug bits.

Addresses: 4101_0000h base + 540h offset + (16d × *n*), where *n* = 0d to 15d



EPDC_DEBUG_LUT*n* field descriptions

Field	Description
31–26 RSVD1	Reserved.
25–16 LUTADDR	LUT address to be filled.
15 RSVD0	Reserved..
14–5 FRAME	The remaining number of frames for this update.
4–0 STATEMACHINE	LUTn state machine

24.4.32 EPDC General Purpose I/O Debug register (EPDC_GPIO*n*)

GPIO register to control ipp_epdc_bdr[1:0], ipp_epdc_pwr[3:0] and ipp_epdc_pwrcom output signals. This register houses software control signal registers.

Addresses: EPDC_GPIO is 4101_0000h base + 700h offset = 4101_0700h
EPDC_GPIO_SET is 4101_0000h base + 704h offset = 4101_0704h
EPDC_GPIO_CLR is 4101_0000h base + 708h offset = 4101_0708h
EPDC_GPIO_TOG is 4101_0000h base + 70Ch offset = 4101_070Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSVD0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD0									PWRCOM	PWRCTRL				BDR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPDC_GPIO*n* field descriptions

Field	Description
31–7 RSVD0	Reserved.
6 PWRCOM	Controls ipp_epdc_pwrcom output
5–2 PWRCTRL	Controls ipp_epdc_pwrctrl[3:0] output
1–0 BDR	Controls ipp_epdc_bdr[1:0] output

24.4.33 EPDC Version Register (EPDC_VERSION)

This register reflects the version number for the EPDC.

Address: EPDC_VERSION is 4101_0000h base + 7F0h offset = 4101_07F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPDC_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 25
Enhanced Periodic Interrupt Timer (EPIT)

25.1 Overview

The EPIT is a 32-bit set-and-forget timer that begins counting after the EPIT is enabled by software. It is capable of providing precise interrupts at regular intervals with minimal processor intervention.

The following figure illustrates the block diagram of the EPIT.

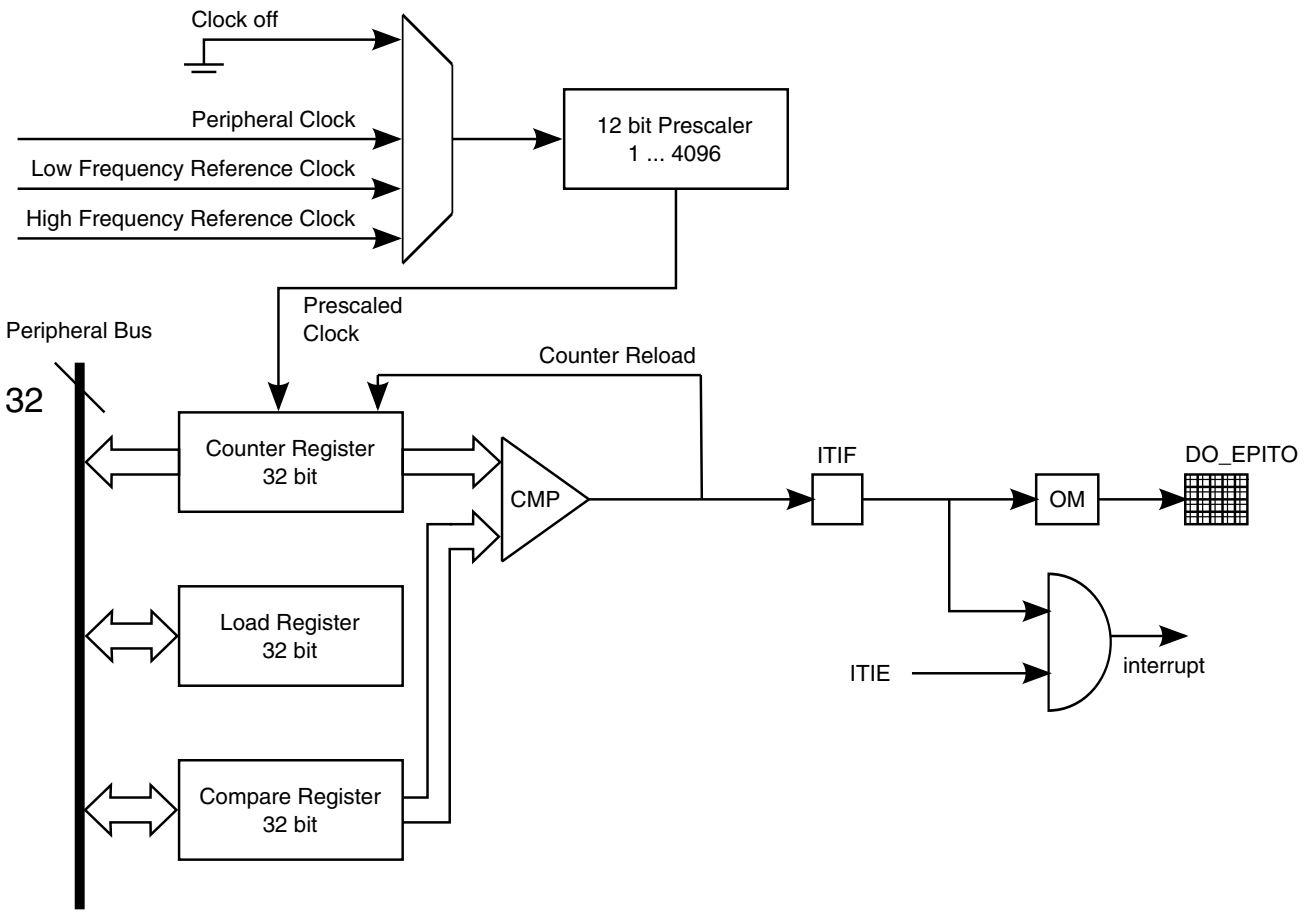


Figure 25-1. EPIT Block Diagram

25.1.1 Features

Key features of the EPIT include:

- 32-bit down counter with clock source selection
- 12-bit prescaler for division of input clock frequency
- Counter value can be programmed on the fly

- Can be programmed to be active during low-power and debug modes
- Interrupt generation when counter reaches the Compare value

25.1.2 Modes and Operations

EPIT supports the modes described in the indicated sections:

- Operating Modes
  - Set-and-Forget Mode
  - Free-Running Mode

EPIT supports the operations as described in Operations.

25.2 External Signals

The following table describes all EPIT's I/O Signals.

Table 25-1. Block I/O Signals

Name	I/O	Description	Reset State	Pull Up
DO_EPITO	O	Output pin at chip boundary for indication of occurrence of output compare event through a specified transition.	0	-

25.3 Functional Description

This section provides a complete functional description of the block.

25.3.1 Operating Modes

This section describes all functional operation modes of the block.

The EPIT can be programmed to function in set-and-forget or free-running modes.

25.3.1.1 Set-and-Forget Mode

This mode of operation is selected when the RLD bit in control register (EPIT_EPITCR) is set to a value of one.

The counter cannot be directly written from the block data bus. Instead, it gets its data from the load register (EPIT_EPITLR). Whenever the counter reaches a count of zero, the value in EPIT_EPITLR is loaded into the counter to be decremented toward zero. The counter may by directly initialized, without having to wait for the count to reach zero, when the EPIT_EPITLR is written with the IOVW bit in EPIT_EPITCR set.

25.3.1.2 Free-Running Mode

This mode of operation is selected when the RLD bit is cleared.

In this mode, the counter rolls over from 0x0000_0000 to 0xFFFF_FFFF without reloading from the modulus register and continues to count down. In this mode when writing to EPIT_EPITLR with the required initialization value after having set the IOVW bit, the counter can also be directly initialized.

25.3.2 Operations

The EPIT has a single 32-bit down counter which starts counting when the block is enabled by software.

The start value of the counter is loaded from the EPIT load register which can be written to at any time by the processor. The value in the compare register determines the time of occurrence of the interrupt.

When EPIT is disabled (EN=0), both the main counter and the prescaler counter freeze their count at current count values. ENMOD bit is a r/w bit which decides the counter value when the EPIT is enabled again by setting EN bit. If ENMOD bit is set, then the main counter is loaded with the load value (If RLD=1)/ 0xFFFF_FFFF (If RLD=0) and the prescaler counter is reset (0x000), when EPIT is enabled (EN=1). If ENMOD is programmed to 0 then both main counter and prescaler counter restart counting from their frozen values, when EPIT is enabled (EN=1). If EPIT is programmed to be disabled in a low-power mode (STOP/WAIT), then both the main counter and the prescaler counter freeze at their current count values when EPIT enters low-power mode. When EPIT exits the low-power mode, both the main counter and the prescaler counter start counting from their frozen values irrespective of the ENMOD bit.

A hardware reset resets all EPIT registers to their respective reset values. There is a software reset which has the same effect on all registers except for the EN, ENMOD, STOPEN and WAITEN bits in the control register. The state of these bits are not affected by software reset. A software reset can be asserted even when the EPIT is disabled.

25.3.3 Clocks

The clock that feeds the prescaler can be selected from among the following sources:

- **High-frequency reference clock**

  This clock is provided by the Clock Control Module (CCM). This clock remains on during low-power mode when the peripheral clock is turned off, allowing EPIT to use this clock in low-power mode. In normal mode, the CCM synchronizes this clock to ahb_clk; in low-power mode, CCM switches to an unsynchronized version.

- **Low-frequency reference clock**

  This 32 kHz reference clock is provided by the CCM. This clock remains on in low-power mode when the peripheral clock is turned off, so EPIT can use this clock during low-power mode. In normal mode, the CCM synchronizes this clock to ahb_clk; in low-power mode, CCM switches to an unsynchronized version. This clock is derived from the external 32kHz crystal.

- **Peripheral clock**

  This is the peripheral clock (PER Clock) which is provided (and optionally gated) by the CCM. This clock is typically used in normal operations. In low-power modes, if the EPIT is programmed to be disabled (via STOPEN or WAITEN), then the peripheral clock can be switched off.

The clock input source is determined by the CLKSRC field in the control register. The clock input to the prescaler can also be disabled by setting CLKSRC to 0b00. **This field value should only be changed after first disabling the EPIT by clearing the EN bit in the EPIT_EPITCR**. For other programming requirements that apply while changing clock source, refer section Change of Clock Source.

The PRESCALER field in the control register is used to select the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value between 1 and 4096. A change in the value of the PRESCALER field is immediately reflected on its output clock frequency. The following figure shows the timing for a change in the prescaler value.

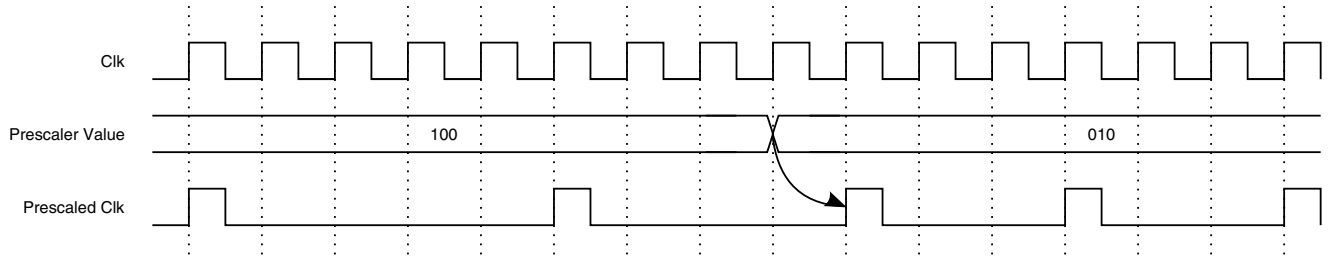


Figure 25-2. Prescaler Value Change Diagram

25.3.4 Compare Event

When the programmed value of EPIT_EPITCMPR matches the value in EPIT_EPITCNR a compare status flag is set, and an interrupt is generated if the OCIEN bit is set in the control register.

The compare output pin is set, cleared, toggled, or not affected at all depending on the setting of the output mode (OM) bits in the control register. If an interrupt is required at rollover (when the counter value reaches 0x0000_0000 and the new value is loaded) then the compare register value should be set equal to the load register value in set-and-forget mode, or equal to 0xFFFF_FFFF in free-running mode.

The following figure shows the timing for a compare event and interrupt.

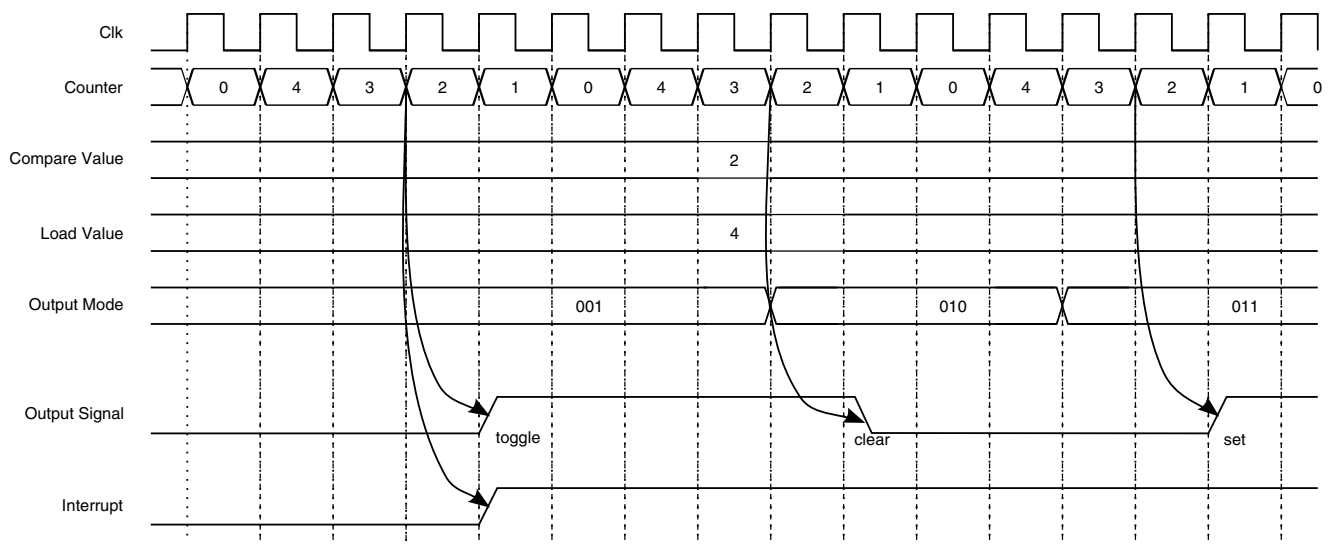


Figure 25-3. Compare Event and Interrupt Timing Diagram

25.3.4.1 Counter Value Overwrite

The EPIT counter value can be overwritten to acquire a desired value at any point of time. The procedure for this is to set the IOVW bit in the control register and then write the desired value into the load register.

This results in the load register acquiring that value and also the counter being overwritten with it. If the EPIT is running the counter resumes counting from the overwritten value.

25.3.4.2 Low-Power Mode Behavior

The EPIT timer's behavior in low-power modes depends on which clock source is being used.

If the selected clock source is available and the corresponding low-power enable bit is set, then the EPIT continues to function in the low-power mode. If the EPIT is programmed to be disabled in a low-power mode (STOP/WAIT), then main counter and the prescaler counter freeze at the current count values when the EPIT enters low-power mode. When the EPIT exits the low-power mode, both main counter and prescaler counter start counting from their frozen values irrespective of the ENMOD bit.

25.3.4.3 Debug Mode Behavior

In debug mode, the user has the option to run or halt the EPIT timers. If the DBGEN bit is reset in the EPIT Control Register, the timer is halted.

When debug mode is exited, the timer operation reverts to what it was prior to entering debug mode.

25.4 Initialization/ Application Information

25.4.1 Change of Clock Source

The CLKSRC field in EPIT_EPITCR determines the clock source. This field value should be changed only after disabling the EPIT (EN = 0).

Below is the software sequence which must be followed while changing clock source.

1. Disable the EPIT by setting EN=0 in EPIT_EPITCR.
2. Program OM=00 in the EPIT_EPITCR.

3. Disable the EPIT interrupts.
4. Program CLKSRC to desired clock source in EPIT_EPITCR.
5. Clear the EPIT status register (EPIT_EPITSR), that is, write "1" to clear (w1c).
6. Enable the EPIT interrupts.
7. Set ENMOD= 1 in the EPIT_EPITCR, to bring the EPIT Counter to defined state (EPIT_EPITLR value or 0xFFFF_FFFF).
8. Enable EPIT (EN=1) in the EPIT_EPITCR

25.5 Programmable Registers

The EPIT includes five user-accessible 32-bit registers. The following table summarizes these registers and their addresses.

Peripheral bus write access to the EPIT control register (EPITCR) and the EPIT load register (EPITLR) results in one cycle of wait state, while other valid peripheral bus accesses are with 0 wait state.

EPIT memory map

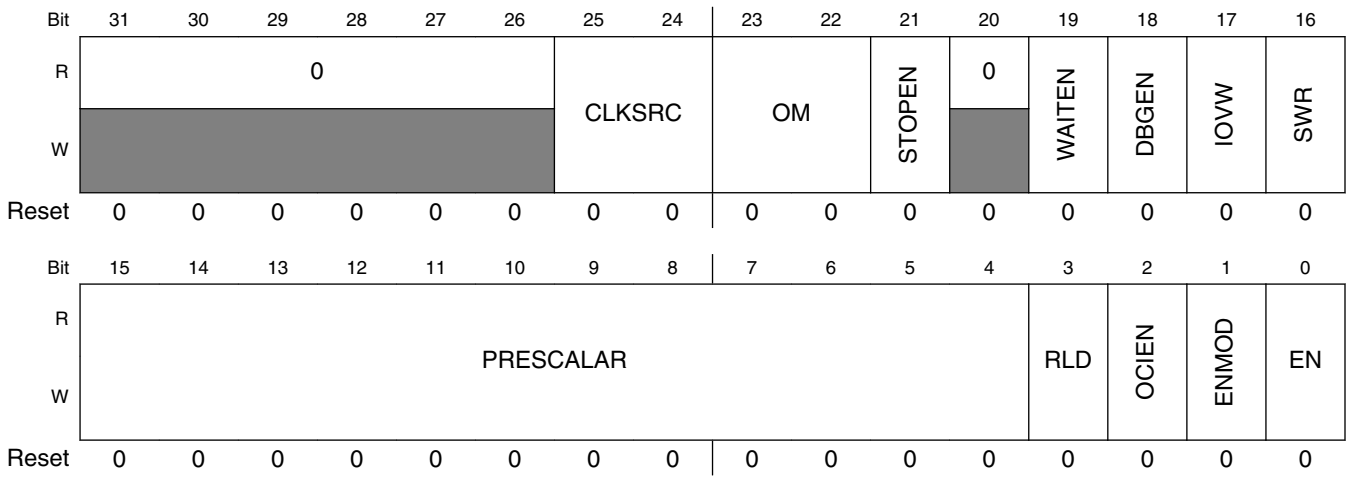
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_C000	Control register (EPIT_EPITCR)	32	R/W	0000_0000h	25.5.1/ 1234
53FA_C004	Status register (EPIT_EPITSR)	32	R/W	0000_0000h	25.5.2/ 1237
53FA_C008	Load register (EPIT_EPITLR)	32	R/W	FFFF_ FFFFh	25.5.3/ 1238
53FA_C00C	Compare register (EPIT_EPITCMPR)	32	R/W	0000_0000h	25.5.4/ 1238
53FA_C010	Counter register (EPIT_EPITCNR)	32	R	FFFF_ FFFFh	25.5.5/ 1239

25.5.1 Control register (EPIT_EPITCR)

The EPIT control register (EPIT_EPITCR) is used to configure the operating settings of the EPIT. It contains the clock division prescaler value and also the interrupt enable bit. Additionally it contains other control bit which are outlined below.

Peripheral Bus Write access to EPIT Control Register (EPIT_EPITCR) results in one cycle of the wait state, while other valid peripheral bus accesses are with 0 wait state.

Address: EPIT_EPITCR is 53FA_C000h base + 0h offset = 53FA_C000h



EPIT_EPITCR field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value zero. Reserved. Writing to these bits does not affect the functionality of EPIT. These bits are always read as zero.
25–24 CLKSRC	Select clock source These bits determine which clock input is to be selected for running the counter. This field value should only be changed when the EPIT is disabled by clearing the EN bit in this register. For other programming requirements while changing clock source, refer to Change of Clock Source. 00 Clock is off 01 Peripheral clock 10 High-frequency reference clock 11 Low-frequency reference clock
23–22 OM	EPIT output mode.This bit field determines the mode of EPIT output on the output pin. 00 EPIT output is disconnected from pad 01 Toggle output pin 10 Clear output pin 11 Set output pin
21 STOPEN	EPIT stop mode enable. This read/write control bit enables the operation of the EPIT during stop mode. This bit is reset by a hardware reset and unaffected by software reset. 0 EPIT is disabled in stop mode 1 EPIT is enabled in stop mode
20 Reserved	This read-only field is reserved and always has the value zero. Reserved. Writing to these bits does not affect the functionality of EPIT. These bits are always read as zero.
19 WAITEN	This read/write control bit enables the operation of the EPIT during wait mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled in wait mode 1 EPIT is enabled in wait mode

Table continues on the next page...

EPIT_EPITCR field descriptions (continued)

Field	Description
18 DBGEN	This bit is used to keep the EPIT functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode.This bit is reset by hardware reset. A software reset does not affect this bit. 0 Inactive in debug mode 1 Active in debug mode
17 IOVW	EPIT counter overwrite enable. This bit controls the counter data when the modulus register is written. When this bit is set, all writes to the load register overwrites the counter contents and the counter starts subsequently counting down from the programmed value. 0 Write to load register does not result in counter value being overwritten. 1 Write to load register results in immediate overwriting of counter value.
16 SWR	Software reset. The EPIT is reset when this bit is set to 1. It is a self clearing bit. This bit is set when the block is in reset state and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values, except for the EN, ENMOD, STOPEN, WAITEN and DBGEN bits in this control register 0 EPIT is out of reset 1 EPIT is undergoing reset
15–4 PRESCALAR	Counter clock prescaler value. This bit field determines the prescaler value by which the clock is divided before it goes to the counter 0x000 Divide by 1 0x001 Divide by 2 0xFFFF Divide by 4096
3 RLD	Counter reload control. This bit is cleared by hardware reset. It decides the counter functionality, whether to run in free-running mode or set-and-forget mode. 0 When the counter reaches zero it rolls over to 0xFFFF_FFFF (free-running mode) 1 When the counter reaches zero it reloads from the modulus register (set-and-forget mode)
2 OCIEN	Output compare interrupt enable. This bit enables the generation of interrupt on occurrence of compare event. 0 Compare interrupt disabled 1 Compare interrupt enabled
1 ENMOD	EPIT enable mode. When EPIT is disabled (EN=0), both main counter and prescaler counter freeze their count at current count values. ENMOD bit is a r/w bit that determines the counter value when the EPIT is enabled again by setting EN bit. If ENMOD bit is set, then main counter is loaded with the load value (If RLD=1)/ 0xFFFF_FFFF (If RLD=0) and prescaler counter is reset, when EPIT is enabled (EN=1). If ENMOD is programmed to 0 then both main counter and prescaler counter restart counting from their frozen values when EPIT is enabled (EN=1). If EPIT is programmed to be disabled in a low-power mode (STOP/WAIT/ DEBUG), then both the main counter and the prescaler counter freeze at their current count values when EPIT enters low-power mode. When EPIT exits the low-power mode, both main counter and prescaler counter start counting from their frozen values irrespective of the ENMOD bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 Counter starts counting from the value it had when it was disabled. 1 Counter starts count from load value (RLD=1) or 0xFFFF_FFFF (If RLD=0)

Table continues on the next page...

EPIT_EPITCR field descriptions (continued)

Field	Description
0 EN	This bit enables the EPIT. EPIT counter and prescaler value when EPIT is enabled (EN =1), is dependent upon ENMOD and RLD bit as described for ENMOD bit. It is recommended that all registers be properly programmed before setting this bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled 1 EPIT is enabled

25.5.2 Status register (EPIT_EPITSR)

The EPIT status register (EPIT_EPITSR) has a single status bit for the output compare event. The bit is a write 1 to clear bit.

Address: EPIT_EPITSR is 53FA_C000h base + 4h offset = 53FA_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															OCIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPIT_EPITSR field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value zero. Reserved. Writing to these bits does not affect the functionality of EPIT. These bits are always read as zero.
0 OCIF	Output compare interrupt flag. This bit is the interrupt flag that is set when the content of counter equals the content of the compare register (EPIT_EPITCMPR). The bit is a write 1 to clear bit. 0 Compare event hasn't occurred 1 Compare event occurred

25.5.3 Load register (EPIT_EPITLR)

The EPIT load register (EPIT_EPITLR) contains the value that is to be loaded into the counter when EPIT counter reaches zero if the RLD bit in EPIT_EPITCR is set. If the IOVW bit in the EPIT_EPITCR is set then a write to this register overwrites the value of the EPIT counter register in addition to updating this registers value. This overwrite feature is active even if the RLD bit is not set.

Address: EPIT_EPITLR is 53FA_C000h base + 8h offset = 53FA_C008h

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

EPIT_EPITLR field descriptions

Field	Description
31–0 LOAD	Load value. Value that is loaded into the counter at the start of each count cycle.

25.5.4 Compare register (EPIT_EPITCMPR)

The EPIT compare register (EPIT_EPITCMPR) holds the value that determines when a compare event is generated.

Address: EPIT_EPITCMPR is 53FA_C000h base + Ch offset = 53FA_C00Ch

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

EPIT_EPITCMPR field descriptions

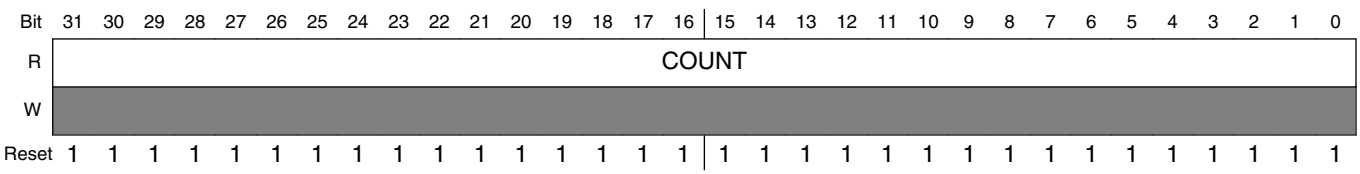
Field	Description
31–0 COMPARE	Compare Value. When the counter value equals this bit field value a compare event is generated.

25.5.5 Counter register (EPIT_EPITCNR)

The EPIT counter register (EPIT_EPITCNR) contains the current count value and can be read at any time without disturbing the counter. This is a read-only register and any attempt to write into it generates a transfer error. But if the IOVW bit in EPIT_EPITCR is set, the value of this register can be overwritten with a write to EPIT_EPITLR. This change is reflected when this register is subsequently read.

.

Address: EPIT_EPITCNR is 53FA_C000h base + 10h offset = 53FA_C010h



EPIT_EPITCNR field descriptions

Field	Description
31–0 COUNT	Counter value. This contains the current value of the counter.

Chapter 26
Enhanced Secured Digital Host Controller (eSDHCv2)

26.1 Overview

The Enhanced Secured Digital Host Controller Version 2 (ESDHCv2, referred to as ESDHC hereafter) provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in Figure 26-1. The ESDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards. It handles the SD/SDIO/MMC protocols at the transmission level.

The types of cards supported by the ESDHC are described briefly as follows:

The Multi Media Card (MMC) is a universal low cost data storage and communication media that is designed to cover a wide area of applications including mobile video and gaming. Old MMC cards are based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment and data transfer protocol are forward compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into Memory card, I/O card and Combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, Figure 26-1 does not show cards with reduced size or mini cards.

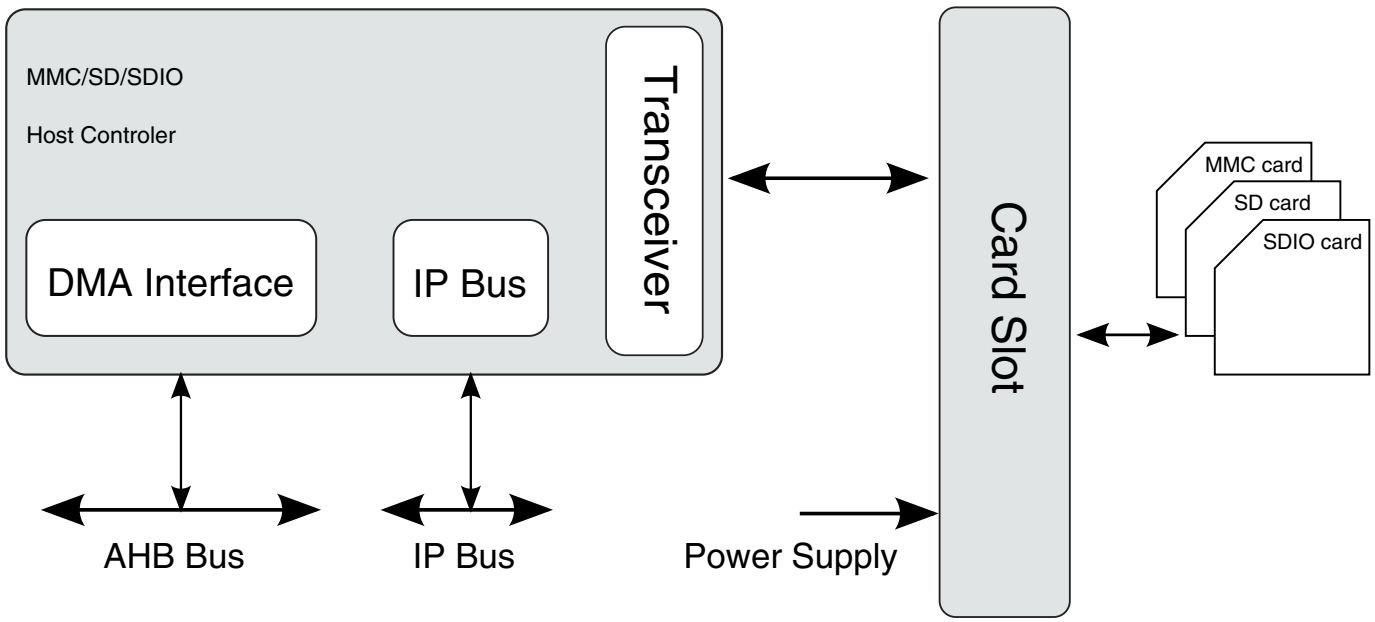


Figure 26-1. System Connection of the ESDHC

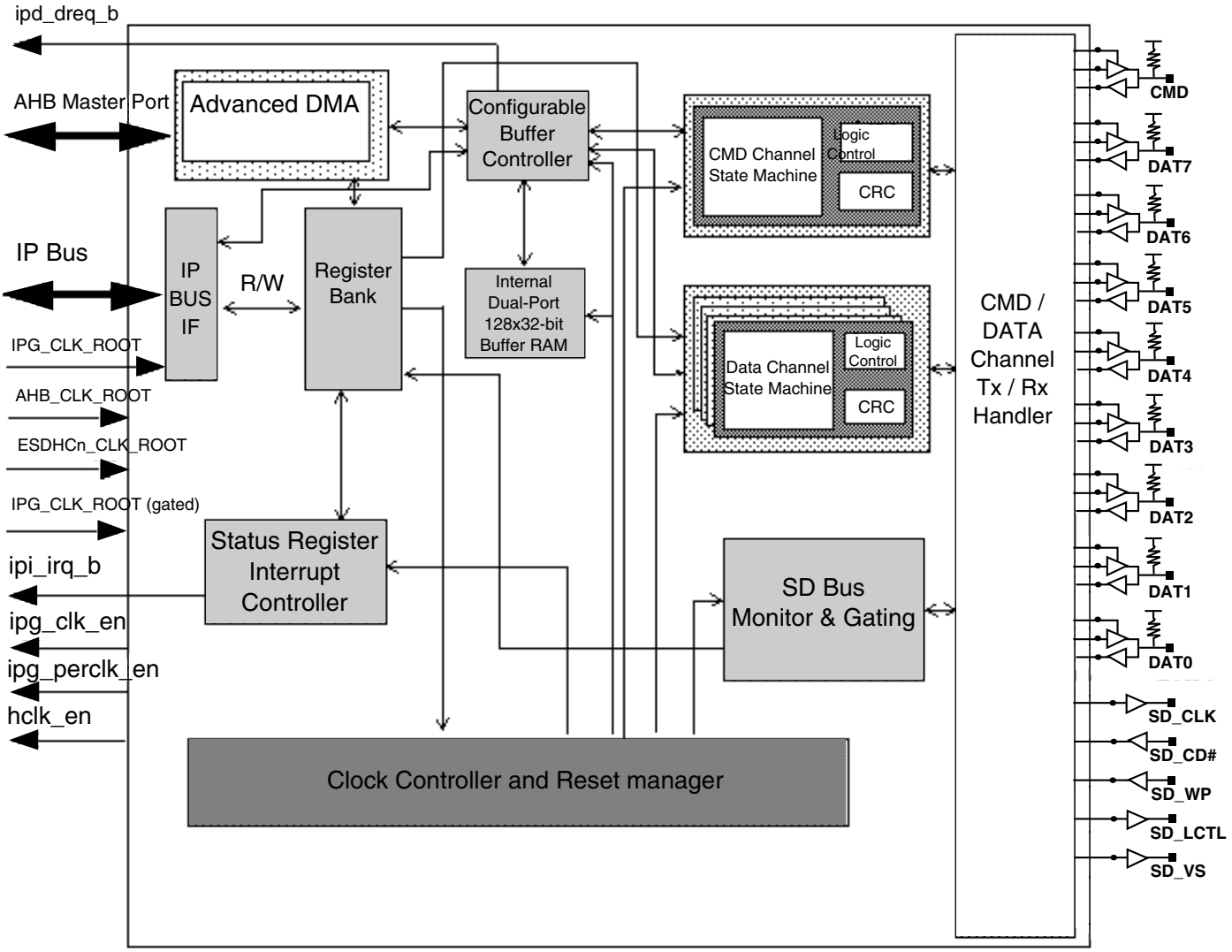


Figure 26-2. enhanced Secure Digital Host Controller Block Diagram

26.1.1 Features

The features of the ESDHC include the following:

- Conforms to the SD Host Controller Standard Specification version 2.0 including Test Event register support
- Compatible with the MMC System Specification version 4.2/4.3
- Compatible with the SD Memory Card Specification version 2.0 and supports the High Capacity SD Memory Card
- Compatible with the SDIO Card Specification version 2.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards

- Card bus clock frequency up to 52 MHz
- Supports 1-bit / 4-bit SD and SDIO modes, 1-bit / 4-bit / 8-bit MMC modes devices
  - Up to 200 Mbps of data transfer for SD/SDIO cards using 4 parallel data lines
  - Up to 416 Mbps of data transfer for MMC cards using 8 parallel data lines in SDR (Single Data Rate) mode
- Supports Single Block, Multi Block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports the write protection switch for write operations
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal and external DMA capabilities
- Supports Advanced DMA to perform linked memory access

26.1.2 Modes and Operations

26.1.2.1 Data transfer Modes

The ESDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- Identification Mode (up to 400 kHz)
- MMC full speed mode (up to 20 MHz)
- MMC high speed mode (up to 52 MHz)
- SD/SDIO full speed mode (up to 25 MHz)
- SD/SDIO high speed mode (up to 50 MHz)

26.2 External Signals

26.2.1 Signals Overview

The ESDHC has 15 associate I/O signals.

- The SD_CLK is an internally generated clock used to drive the MMC, SD, SDIO cards.
- The CMD I/O is used to send commands and receive responses to/from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the ESDHC and the card.
- The SD_CD# and SD_WP are card detection and write protection signals directly routed from the socket. A low on SD_CD# means that a card is inserted and a high on SD_WP means that the write protect switch is active.
- SD_OD is an output signal generated in SoC level outside ESDHC and is used to select the external open drain resistor.
- SD_LCTL is an output signal used to drive an external LED to indicate that the SD interface is busy.

SD_CD#, SD_WP, SD_OD, SD_LCTL are all optional for system implementation. If the ESDHC is desired to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high.

26.2.2 Ports Table

See Table 26-1 for the signal properties of the I/Os.

Table 26-1. Properties of I/O Signals

Name	Port	Function	Reset State	Pull up
SD_CLK	O	Clock for MMC/SD/SDIO card	0	N/A
SD_CMD	I/O	CMD line connect to card	1	Pull up
SD_DAT7	I/O	DAT7 line in 8-bit mode Not used in other modes	1	Pull up
SD_DAT6	I/O	DAT6 line in 8-bit mode Not used in other modes	1	Pull up
SD_DAT5	I/O	DAT5 line in 8-bit mode Not used in other modes	1	Pull up
SD_DAT4	I/O	DAT4 line in 8-bit mode Not used in other modes	1	Pull up

Table continues on the next page...

Table 26-1. Properties of I/O Signals (continued)

Name	Port	Function	Reset State	Pull up
SD_DAT3	I/O	DAT3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	0	Should be pull-up/pull-down configurable as this port may be used for card detection
SD_DAT2	I/O	DAT2 line or Read Wait in 4-bit mode Read Wait in 1-bit mode	1	Pull up
SD_DAT1	I/O	DAT1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	1	Pull up
SD_DAT0	I/O	DAT0 line in all modes Also used to detect busy state	1	Pull up
SD_CD#	I	Card detection pin If not used tie high	N/A	N/A
SD_WP	I	Card write protect detect If not used tie low	N/A	N/A
SD_OD	O	Open drain select (not generated within the ESDHC). Optional output	N/A	N/A
SD_LCTL	O	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	0	N/A

26.3 Functional Description

The following sections provide a brief functional description of the major system blocks, including the Data Buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock & reset manager and clock generator.

26.3.1 Data Buffer

The ESDHC uses one configurable data buffer, so that data can be transferred between the system bus (IP Bus or AHB Bus) and the SD card, with an optimized manner to maximize throughput between the two clock domains (that is, the IP peripheral clock, and the master clock). See the table below for illustration of the buffer scheme. The buffer is used as temporary storage for data being transferred between the host system and the

card. The watermark levels for read and write are both configurable, and can be any number from 1 to 128 words. The burst lengths for read and write are also configurable, and can be any number from 1 to 31 words.

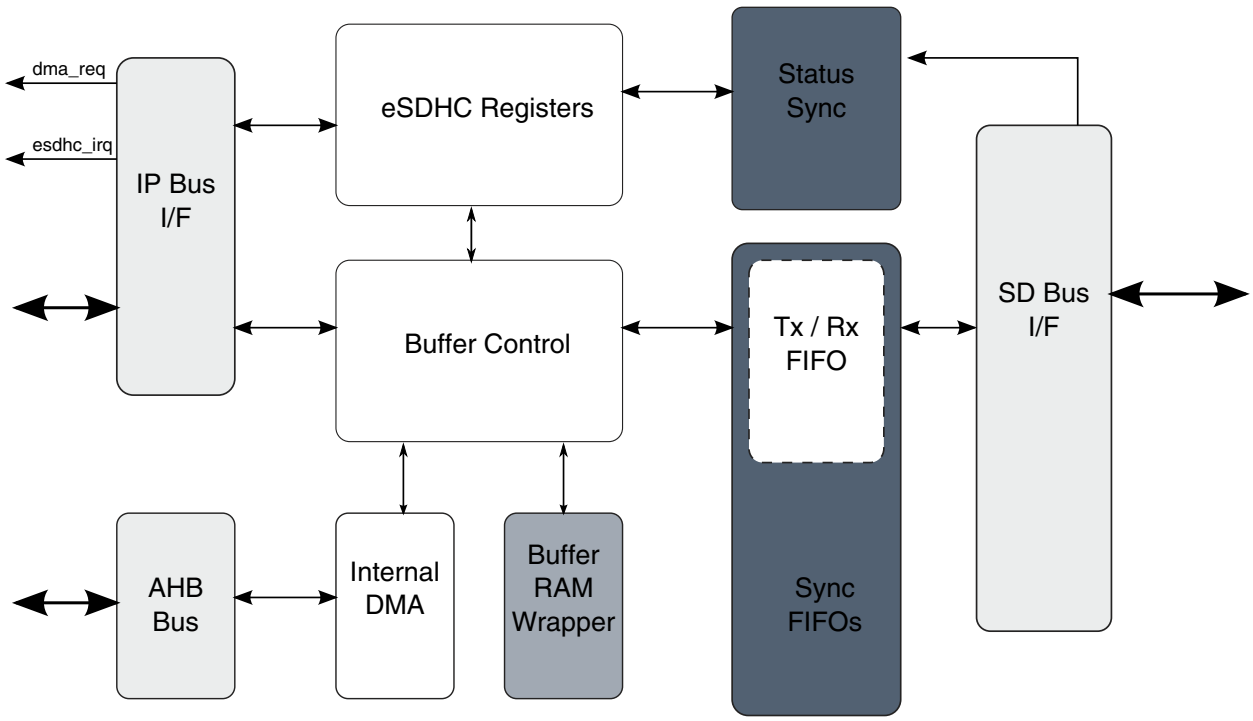


Figure 26-3. ESDHC Buffer Scheme

There are 3 transfer modes to access the data buffer:

- ARM platform polling mode:
  - For a host read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, then by polling the BRR bit the Host Driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar.
- External DMA mode:
  - For a read operation, when there are more words received in the buffer than the amount set in the RD_WML register, a DMA request is sent out to inform the external DMA to fetch the data. The request will be immediately de-asserted when there is an access on the Buffer Data Port register. If the number of words in the buffer after the current burst meets or exceeds RD_WML value, then the DMA request is asserted again. For instance, if there are twice as many words in the buffer than the RD_WML value, there are two successive DMA requests with only one cycle of de-assertion between. The write operation is similar.

NOTE

Data buffer accesses by the ARM platform polling mode and external DMA mode both use the IP Bus. In these mode, if the external DMA is enabled, an external DMA request is sent out every time the buffer is ready.

- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus. For internal DMA access mode, the external DMA request will never be sent out.

    For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except INCR4 and INCR8, the burst type is always INCR mode and the burst length depends on the shortest of following factors:

1. Burst length configured in the burst length field of the Watermark Level register
2. Watermark Level boundary
3. Block size boundary
4. Data boundary configured in the current descriptor (if the ADMA is active)
5. 1 Kbyte address boundary defined in the AHB protocol

Write operation is similar.

Sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actually byte order is swapped inside the buffer, according to the endian mode configured by software. See Figure 26-4 and Figure 26-5. For a host write operation, byte order is swapped after data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, byte order is swapped before the data is stored into the buffer.

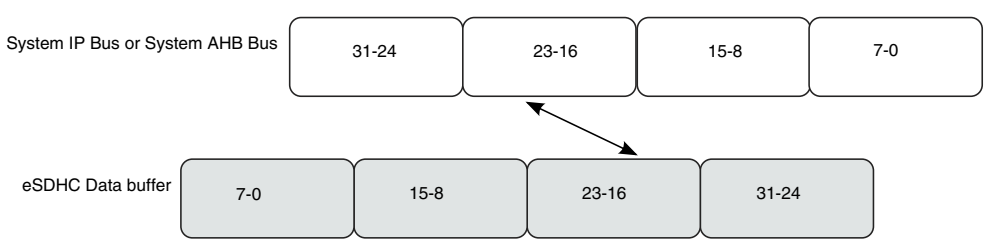


Figure 26-4. Data Swap between System Bus and ESDHC Data Buffer in Byte Little Endian Mode

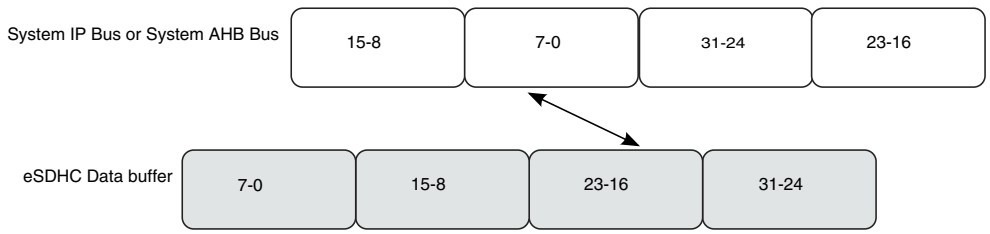


Figure 26-5. Data Swap between System Bus and ESDHC Data Buffer in Half Word Big Endian Mode

26.3.1.1 Write Operation Sequence

There are three ways to write data into the buffer when the user transfers data to the card:

1. By using external DMA through the ESDHC DMA request signal.
2. By processor core polling through the BWR bit in Interrupt Status register (interrupt or polling).
3. By using the internal DMA.

When the internal DMA is not used, (i.e. the DMAEN bit in the Transfer Type register is not set when the command is sent), the ESDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register, and is ready for receiving new data. At the same time, the ESDHC would set the BWR bit. The buffer write ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the ESDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the ESDHC will abort the data transfer and abandon the current block. The Host Driver should read the contents of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi block mode, the ESDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver shall send CMD12 in this scenario and re-start the write operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started after error recovery.

The ESDHC will not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the Host System does not write data in time, the ESDHC will stop the SD_CLK to avoid the data buffer under-run situation.

26.3.1.2 Read Operation Sequence

There are three ways to read data from the buffer when the user transfers data from the card:

1. By using the external DMA through the ESDHC DMA request signal
2. By processor core polling through the BRR bit in Interrupt Status register (interrupt or polling)
3. By using the internal DMA

When internal DMA is not used (i.e. DMAEN bit in Transfer Type register is not set when the command is sent), the ESDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register (that is, available and ready for system fetching data). At the same time, the ESDHC would set the BRR bit. The buffer read ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the ESDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the ESDHC will abort the data transfer and abandon the current block. The Host Driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi block mode, the ESDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver shall send CMD12 in this scenario and re-start the read operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started after error recovery.

For any read transfer mode, the ESDHC will not start data transmission until the number of words set in the RD_WML register are in the buffer. If the buffer is full and the Host System does not read data in time, the ESDHC will stop the SD_CLK to avoid the data buffer over-run situation.

26.3.1.3 Data Buffer and Block Size

The user needs to know the buffer size, for the buffer operation during a data transfer, to utilize it in the most optimized way. In the ESDHC, the only data buffer can hold up to 128 words (32-bit), and the watermark levels for write and read can be configured respectively. For both read and write, the watermark level can be from 1 word to the maximum of 128 words. For both read and write, the burst length, can be from 1 word to the maximum of 31 words. The Host Driver may configure the value according to the system situation and requirement.

During a multi-block data transfer, the block length may be set to any value between 1 and 4096 bytes inclusive which satisfies the requirements of the external card. The only restriction is from the external card. It might not support that large of a block or it does not support a partial block access (which is not the integer times of 512 bytes).

For block size not times of 4, that is, not word aligned, ESDHC requires stuff bytes at the end of each block, because ESDHC treats each block individually. For example, if the block size is 7 bytes and there are 12 blocks to write, the system side must write two times for each block. For each block, the ending byte will be abandoned by ESDHC because it only sends 7 bytes to the card and picks data from the following system write, making a total of 24 beats of write access.

26.3.1.4 Dividing Large Data Transfer

This SDIO command CMD53 definition, limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length can't be divided evenly into a multiple of the block size, then there are two ways to transfer the data which depend on the function and the card design. Option 1 is for the Host Driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See figure below for an example showing the dividing of large data transfers. Assuming a kind of WLAN SDIO card only supports block size up to 64 bytes. Although the ESDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size less than64 bytes, so the data must be divided (see example below).

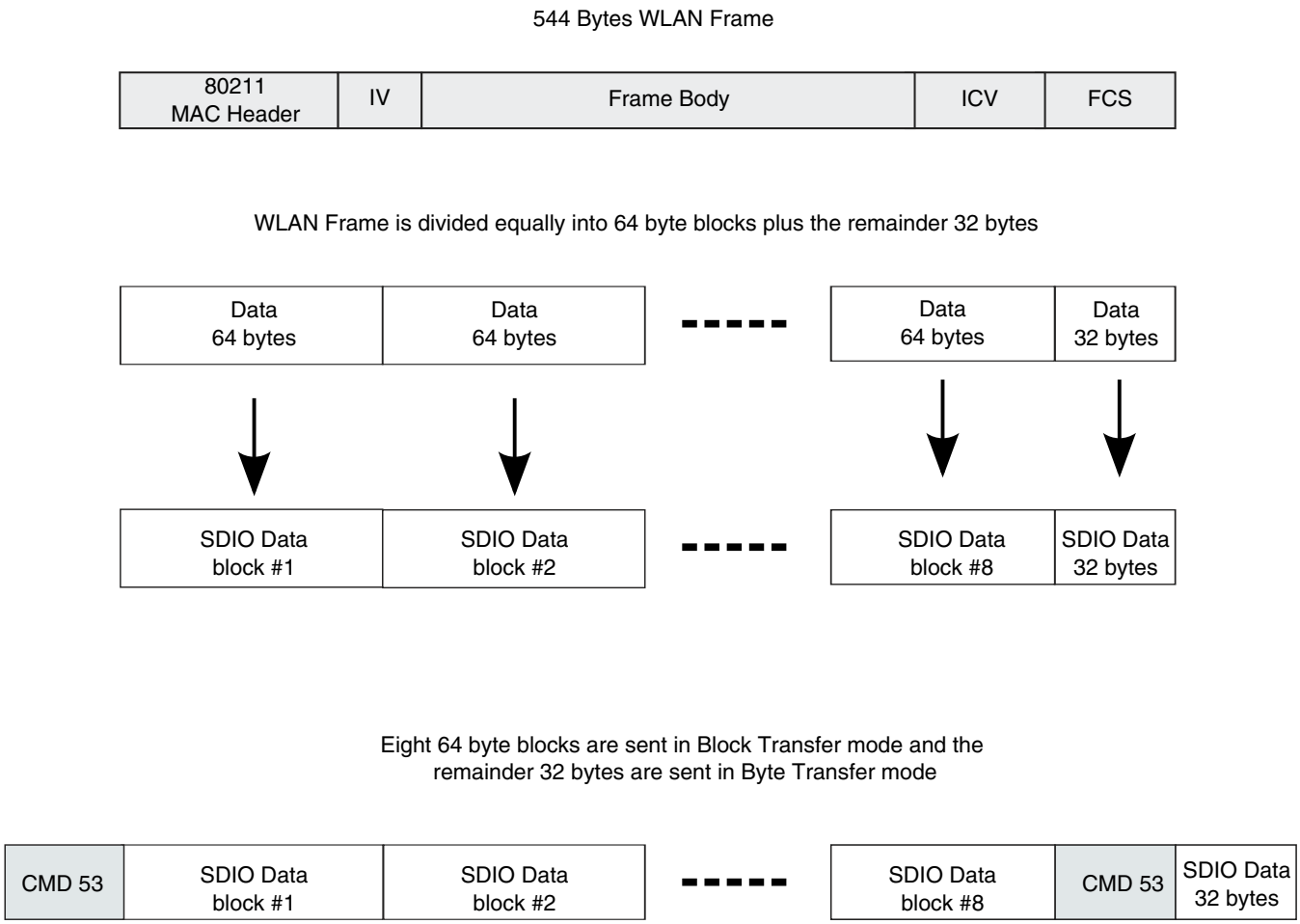


Figure 26-6. Example for Dividing Large Data Transfers

26.3.1.5 External DMA Request

When the internal DMA is not in use, and external DMA request is enabled, the Data Buffer will generate a DMA request to the system. During a write operation, when the number of WR_WML words can be held in the buffer free space, the signal esdhc_dreq_b is asserted to 0, informing the Host System of a DMA write. The BWR bit in the Interrupt Status register is also set, as long as the BWRSEN bit in the Interrupt Status Enable register is set. The DMA request is immediately de-asserted when an access to the Data Port register is made. If the buffer's free space still meets the watermark condition, the DMA request is asserted again after a cycle.

On read operation, when the number of RD_WML words are already in the buffer, the signal esdhc_dreq_b is asserted to 0, informing the Host System for a DMA read. The BRR bit in the Interrupt Status register is also set, as long as the BRRSEN bit in the

Interrupt Status Enable register is set. The DMA request is immediately de-asserted when an access to the Data Port register is made. If the buffer's data still meets the watermark condition, the DMA request is asserted again after a cycle.

Because the DMA burst length can not change during a data transfer for an external DMA transfer, the watermark level (read or write) must be a divisor of the block size. If it is not, transferring of the block may cause buffer under-run (read operation) or over-run (write operation). For example, if the block size is 512 bytes, the watermark level of read (or write) must be a power of two between 1 and 128. For processor core polling access, as the last access in the block transfer can be controlled by software, there is no such issue. The watermark level can be any value, even larger than the block size (but no greater than 128 words). This is because the actual number of bytes transferred by the software can be controlled and does not exceed the block size in each transfer.

The ESDHC also supports non-word aligned block size, as long as the card supports that block size. In this case, the watermark level should be set as the number of words. For example, if the block size is 31 bytes, the watermark level can be set to any number of word. For this case, the BLKSIZE bits of the Block Attribute register shall be set as 1fh. For the ARM platform polling access, the burst length can be 1 to 128 words, without restriction. This is because the software will transfer 8 words, and the ESDHC will also set the BWR or BRR bits when the remaining data does not violate data buffer. Refer to DMA Burst Length for more details about the dynamic watermark level of the data buffer. For the above example, even though 8 words are transferred through the Data Port register, the ESDHC will transfer only 31 bytes over the SD Bus, as required by the BLKSIZE bits. In this data transfer, with non-word aligned block size, the endian mode should be set cautiously, or invalid data will be transferred to/from the card.

26.3.2 DMA AHB Interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, the esdhc_dreq_b will not be asserted during the transfer, but the BWR and BRR bits will be set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register. See the figure below for an illustration of the DMA AHB interface block.

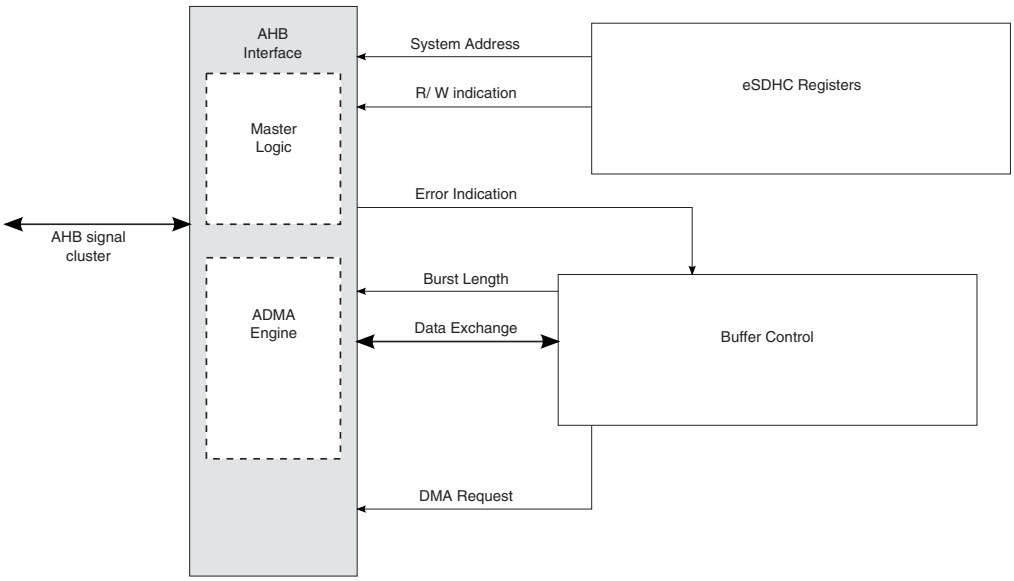


Figure 26-7. DMA AHB Interface Block

26.3.2.1 Internal DMA Request

If the watermark level requirement is met in data transfer, and the Internal DMA is enabled, the Data Buffer block will send a DMA request to AHB interface. Meanwhile, the external DMA request signal (esdhc_dreq_b) is disabled. The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to the ESDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The Data Buffer de-asserts the request once an access to the buffer is made. Upon access to the buffer by internal DMA, the Data Buffer updates its internal buffer pointer, and when the watermark level is satisfied, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to 6 words. After the first burst transfer, if there are more than 2 words in the buffer (which might contain some data of the next block), another DMA request read is sent. This is because the remaining number of words to send for the current block is (31 - 6 * 4) / 4 = 2. The ESDHC will read 2 words out of the buffer, with 7 valid bytes and 1 stuff byte.

26.3.2.2 DMA Burst Length

Just like a ARM platform polling access, the DMA burst length for the internal DMA engine can be from 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block. Take the example in Internal DMA Request again. The following burst length after 6 words are read will be 2 words, and the next burst length will be 6 words again. This is because the next block starts, which is 31 bytes, more than 6 words. The Host Driver writer may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length will be the same.

26.3.2.3 AHB Master Interface

It is possible that the internal AHB DMA engine could fail during the data transfer. When this error occurs, the DMA engine stops the transfer and goes to the idle state as well as the internal data buffer stops accepting incoming data. The DMAE bit in the Interrupt Status register is set to inform the driver.

Once the DMAE interrupt is received, the software shall send a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and re-start the transfer from this address to recover the corrupted block.

26.3.2.4 ADMA Engine

In the SD Host Controller Standard, the new DMA transfer algorithm called the ADMA (Advanced DMA) is defined. For Simple DMA, once the page boundary is reached, a DMA interrupt will be generated and the new system address shall be programmed by the Host Driver. The ADMA defines the programmable descriptor table in the system memory. The Host Driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized since the Host MCU intervention would not be needed during long DMA based data transfers.

There are two types of ADMA: ADMA1 and ADMA2 in Host Controller. ADMA1 can support data transfer of 4KB aligned data in system memory. ADMA2 removes the restriction so that data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA engine can recognize all kinds of descriptors define in SD Host Controller Standard, and if 'End' flag is detected in the descriptor, ADMA engine will stop after this descriptor is processed.

26.3.2.4.1 ADMA Concept and Descriptor Format

For ADMA1, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Set data length descriptor.
- Set data address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

For ADMA2, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Rsv descriptor.
- Set data length & address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it will ignore the high 32-bit. Address field shall be set on word aligned (lower 2-bit is always set to 0). Data length is in byte unit.

ADMA will start read/write operation after it reaches the Tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last Set type descriptor before Tran type descriptor. Every Tran type will trigger a transfer, and the transfer data length is extracted from the most recent Set type descriptor. If there is no Set type descriptor after the previous Trans descriptor, the data length will be the value for previous transfer, or 0 if no Set descriptor is ever met.

For ADMA2, Tran type descriptor contains both data length and transfer data address, so only a Tran type descriptor can start a data transfer.

Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 26-8. Format of the ADMA1 Descriptor Table

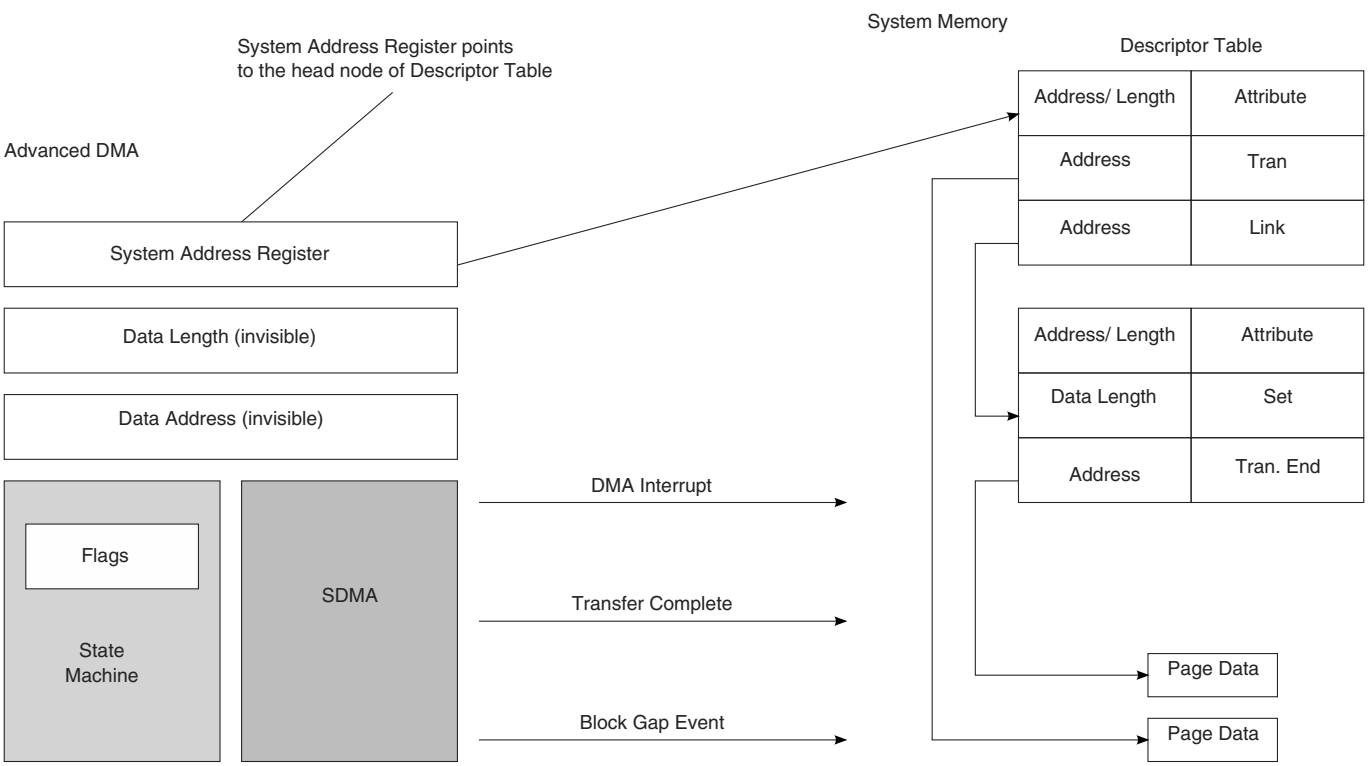


Figure 26-9. Concept and Access Method of ADMA1 Descriptor Table

Address/ Page Field		Address/ Page Field		Address/ Page Field		Attribute Field					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit Length		0000000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Don't Care
0	1	Rsv	Reserved	Same as Nop. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Valid	Valid = 1indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is done.

Figure 26-10. Format of the ADMA2 Descriptor Table

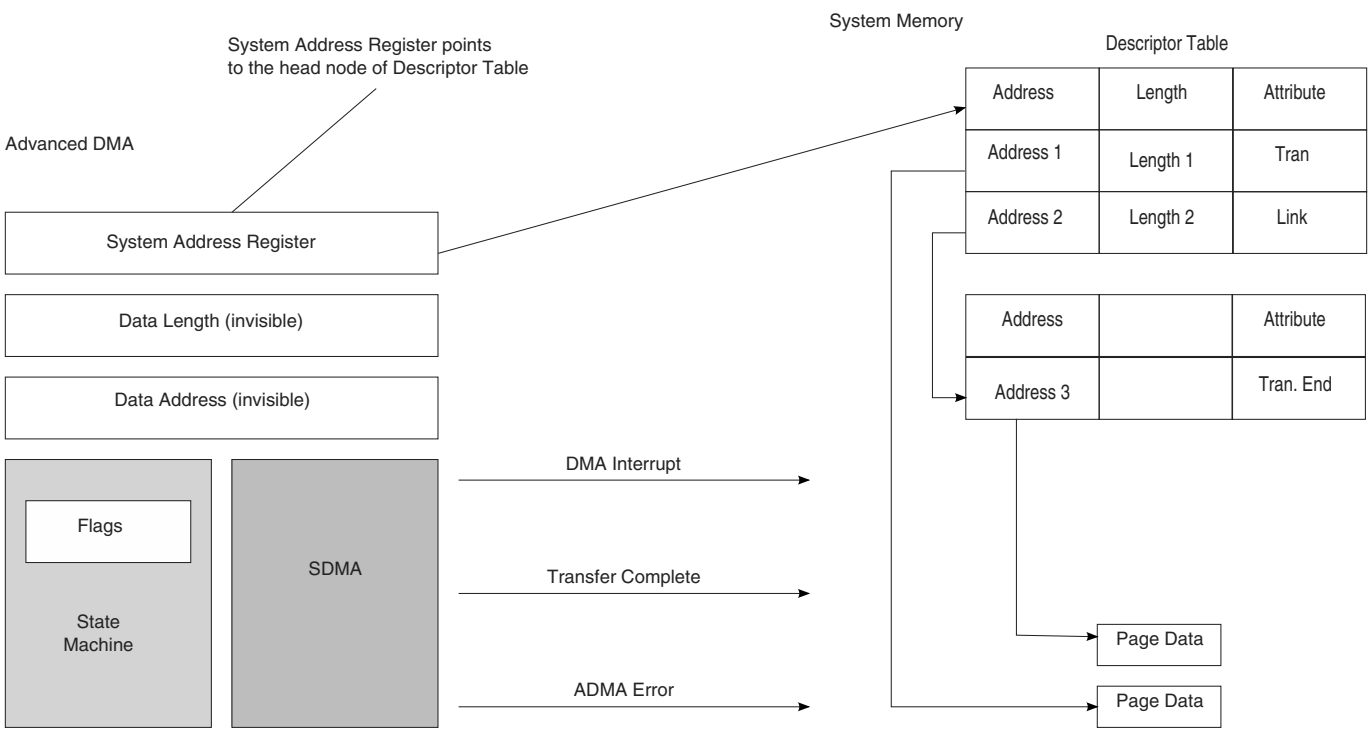


Figure 26-11. Concept and Access Method of ADMA2 Descriptor Table

26.3.2.4.2 ADMA Interrupt

If the 'Interrupt' flag of descriptor is set, ADMA will generate an interrupt according to different type descriptor:

For ADMA1:

- Set type descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type descriptor: interrupt is generated when new descriptor address is set.
- Nop type descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type descriptor: interrupt is generated just after fetch this descriptor.

26.3.2.4.3 ADMA Error-DMA

The ADMA will stop whenever any error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

ADMA descriptor error will be generated when it fails to detect 'Valid' flag in the descriptor. If adma descriptor error occurs, the interrupt is not generated even if the 'Interrupt' flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in Block Attributes register must equal to the whole data length set in descriptor nodes, otherwise data length mismatch error will be generated.

If BLKCNTEN bit is not set, the whole data length set in descriptor should be times of block length, otherwise, when all data set in the descriptor nodes are done not at block boundary, the data mismatch error will occur.

26.3.3 Register Bank with IP Bus Interface

Register accesses via the IP Bus interface are actually on the Register Bank. See the figure below for the block diagram.

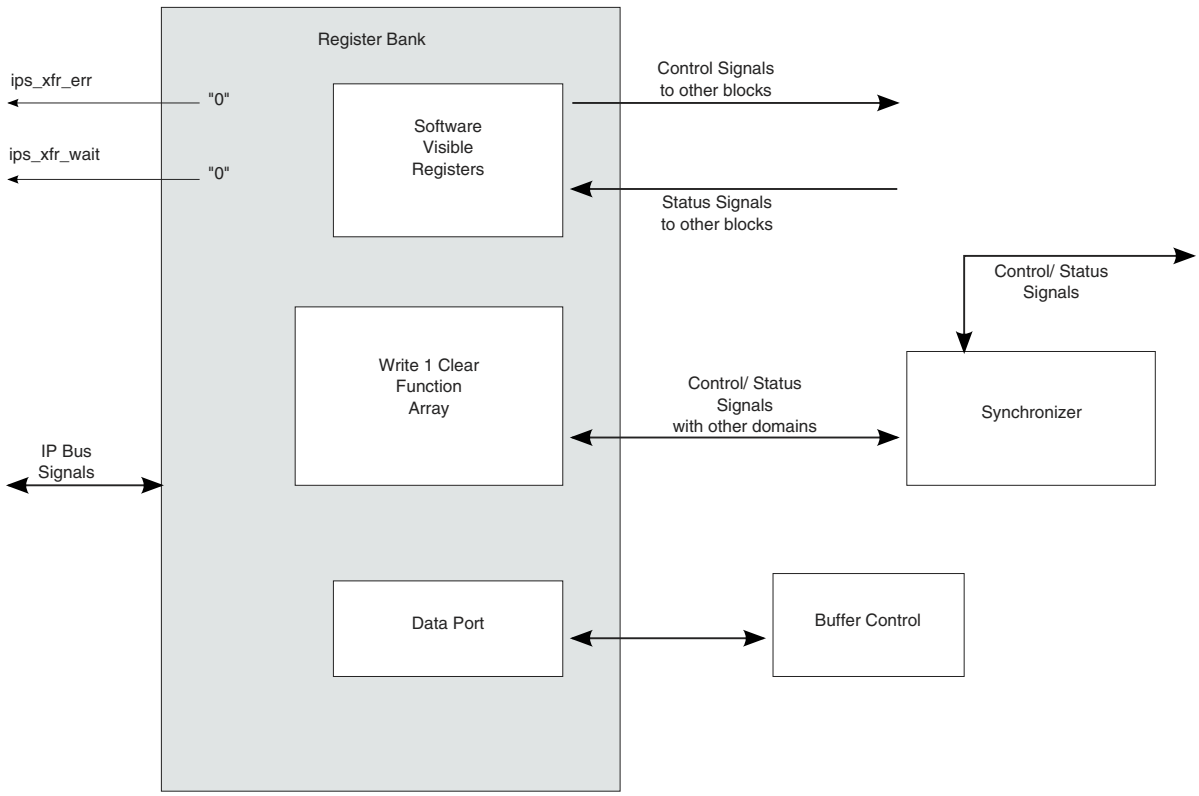


Figure 26-12. Register Bank Diagram

Only 32-bit access is allowed, and no partial read / write is supported, thus all accesses are word aligned.

26.3.4 SD Protocol Unit

The SD protocol unit deals with all SD protocol affairs.

The SD Protocol Unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the DAT[0] line
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD Protocol Unit consists of four sub-blocks:

1. SD transceiver.

2. SD clock and monitor.
3. Command agent.
4. Data agent.

26.3.4.1 SD Transceiver

In the SD protocol unit, the transceiver is the main control sub-block. It consists of an FSM and control sub-block, from which the control signals for all other three sub-blocks are generated.

26.3.4.2 SD Clock & Monitor

This sub-block monitors the signal level on all 8 data lines, the command lines, and directly routes the level values into the Register Bank. The driver can use this for debug purposes.

The sub-block also detects the Card Detection (CD) line as well as the DAT[3] line. The transceiver reports the card insertion state according to the CD state, or the signal level on the DAT[3] line, when the D3CD bit in the Protocol Control register is set.

The sub-block detects the Write Protect (WP) line. With the information of the WP state, the Register Bank will ignore the command, accompanied by a write operation, when the WP switch is on.

If the internal data buffer is in danger, and the SD clock must be gated off to avoid buffer over/under-run, this sub-block will assert the gate of the output SD clock to shut the clock off. After the buffer danger has recovered, and when the system access of the buffer catches up, the clock gate of this sub-block will open and the SD clock will be active again.

This sub-block also drive SD_LCTL output signal when the LCTL bit is set by the driver.

26.3.4.3 Command Agent

The Command Agent deals with the transactions on the CMD line. See figure below for an illustration of the structure for the Command CRC Shift Register.

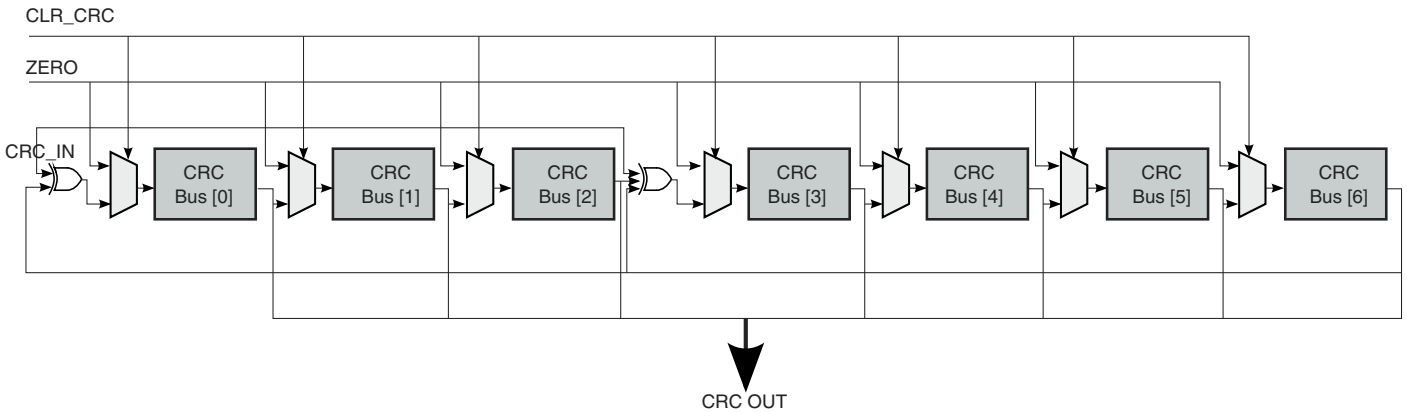


Figure 26-13. Command CRC Shift Register

The CRC polynomials for the CMD are as follows:

Generator polynomial: $G(x) = x^7 + x^3 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$

26.3.4.4 Data Agent

The Data Agent deals with the transactions on the eight data lines. Moreover, this sub-block also detects the busy state on the DAT[0] line, and generate the Read Wait state by the request from the Transceiver. The CRC polynomials for the DAT are as follows:

Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[15:0] = \text{Remainder } [(M(x) * x^{16}) / G(x)]$

26.3.5 Clock & Reset Manager

This sub-block controls all the reset signals within the ESDHC.

There are four kinds of reset signals within ESDHC:

1. Hardware reset.
2. Software reset for all.
3. Software reset for the data part.
4. Software reset for the command part.

All these signals are fed into this sub-block and stable signals are generated inside the module to reset all other modules. The sub-block also gates off all the inside signals.

There are three clocks inside the ESDHC:

1. ipg_clk.

2. ipg_perclk.
3. hclk.

The sub-block monitors the activities of all other sub-blocks, supplies the clocks for them, and when enabled, automatically gates off the corresponding clocks.

26.3.6 Clock Generator

The Clock Generator generates the SD_CLK by peripheral source clock in two stages. Refer to Figure 26-14 for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.

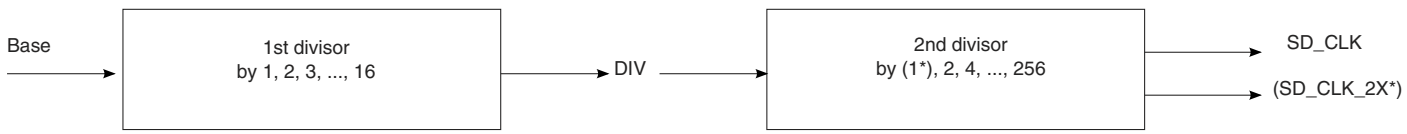


Figure 26-14. Two Stages of the Clock Divider

The first stage outputs an intermediate clock (DIV), which can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler, and outputs the actual clock (SD_CLK).This clock is the driving clock for all sub-blocks of the SD Protocol Unit, and the sync FIFOs (see Figure 26-3) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage, can beDIV, DIV/2, DIV/4,..., or DIV/256. Thus the highest frequency of the SD_CLK is Base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the Base clock is of equal duty ratio (usually true), the duty cycle of SD_CLK is also 50%, even when the compound divisor is an odd value.

26.3.7 SDIO Card Interrupt

26.3.7.1 Interrupts in 1-bit Mode

In this case the DAT[1] pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the DAT[1] low from the SDIO card, until the interrupt service is finished to clear the interrupt.

26.3.7.2 Interrupt in 4-bit Mode

Since the interrupt and data line 1 share Pin 8 in 4-bit mode, an interrupt will only be sent by the card and recognized by the host during a specific time. This is known as the Interrupt Period. The ESDHC will only sample the level on Pin 8 during the Interrupt Period. At all other times, the host will ignore the level on Pin 8, and treat it as the data signal. The definition of the Interrupt Period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the Interrupt Period becomes active two clock cycles after the completion of a data packet. This Interrupt Period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is only a limited period of time that the Interrupt Period can be active due to the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the Interrupt Period. For this case, the Interrupt Period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DAT[1] line will be held low for one clock cycle with the last clock cycle pulling DAT[1] high. On completion of the Interrupt Period, the card releases the DAT[1] line into the high Z state. The ESDHC samples the DAT[1] during the Interrupt Period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10f for further information about the SDIO card interrupt.

26.3.7.3 Card Interrupt Handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, the ESDHC clears the interrupt request to the Host System. The Host Driver should clear this bit before servicing the SDIO Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Status bit is cleared by resetting the SDIO interrupt. Writing to this bit would have no effects. In 1-bit mode, the ESDHC will detect the SDIO Interrupt with or without the SD clock (to support wakeup). In 4-bit mode, the interrupt signal is sampled during the Interrupt Period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status has been set, and the Host Driver needs to service this interrupt, so the SDIO bit in the Interrupt Control Register of SDIO card will be cleared. This is required to clear the SDIO interrupt status latched in the ESDHC and to stop driving the interrupt signal to the

System Interrupt Controller. The Host Driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Enable bit is set to 1, and the ESDHC starts sampling the interrupt signal again.

See figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

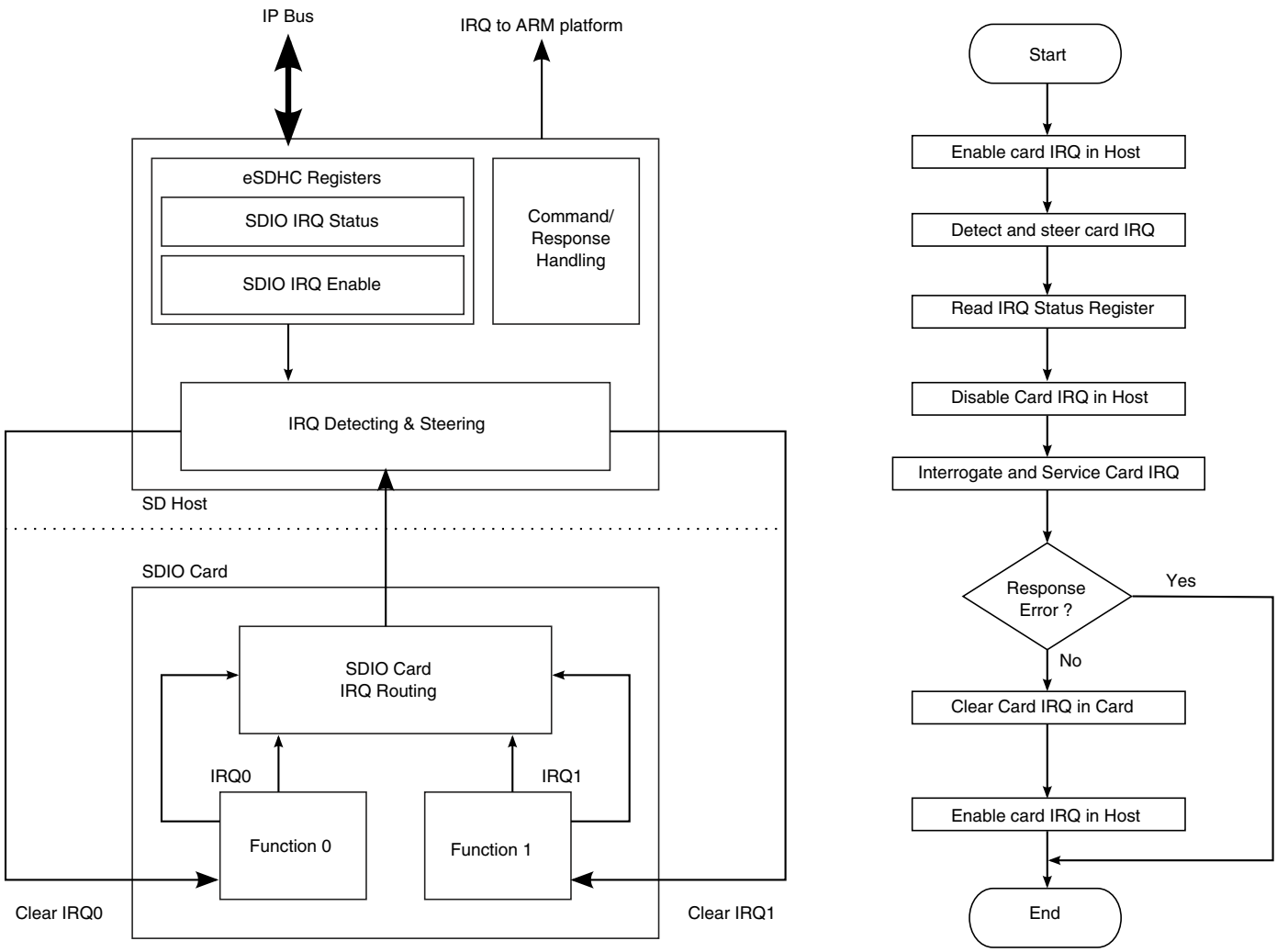


Figure 26-15. Card Interrupt Scheme and Card Interrupt Detection and Handling Procedure

26.3.8 Card Insertion and Removal Detection

The ESDHC uses either the DAT[3] pin or the CD pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DAT[3] will be pulled to a low voltage level by default. When any card is inserted to or removed from the socket, the ESDHC detects the logic value changes on the DAT[3] pin and generates an interrupt. When the DAT[3] pin is not used for card detection (for example, it is implemented in GPIO), the CD pin must be connected for card detection. Whether DAT[3] is configured for card detection or not, the CD pin is always a reference for card detection. Whether the DAT[3] pin or the CD pin is used to detect card insertion, the ESDHC will send an interrupt (if enabled) to inform the Host system that a card is inserted.

26.3.9 Power Management and Wake Up Events

When there is no operation between the ESDHC and the card through the SD bus, the user can completely disable the ipg_clk and ipg_perclk in the chip level clock control module to save power. When the user needs to use the ESDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to the ESDHC are disabled, for instance, when the system is in low power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The ESDHC can generate these interrupt even when there are no clocks enabled. The three interrupts which can be used as wake up events are:

1. Card Removal Interrupt
2. Card Insertion Interrupt
3. Interrupt from SDIO card

The ESDHC offers a power management feature. By clearing the clock enabled bits in the System Control Register (ESDHC_SYSCTL), the clocks are gated in the low position to the ESDHC. For maximum power saving, the user can disable all the clocks to the ESDHC when there is no operation in progress.

These three wake up events (or wakeup interrupts) can also be used to wake up the system from low-power modes.

NOTE

To make the interrupt a wakeup event, when all the clocks to the ESDHC are disabled or when the whole system is in low power mode, the corresponding wakeup enabled bit needs to be set. Refer to Protocol Control (ESDHCV3_PROCTL) for more information on the ESDHC Protocol Control register.

26.3.9.1 Setting Wake Up Events

For the ESDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the ARM platform enters sleep mode. Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No Read or Write Transfer is active
- Data and Command lines are not active
- No interrupts are pending
- Internal data buffer is empty

26.3.10 MMC Fast Boot

The Embedded Multimedia Card (eMMC 4.3) has a fast boot feature. In boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

There are two types of fast boot mode in the eMMC4.3 spec: 'Boot operation' and 'Alternative boot operation.' Each type also has with acknowledge and without acknowledge modes. eSDHCv2 supports both fast boot modes.

NOTE

For the eMMC4.3 card setting, please refer to the eMMC4.3 spec.

26.3.10.1 Boot Operation

NOTE

In this block guide, this fast boot is called normal fast boot mode.

If the CMD line is held LOW for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DAT line(s). The master must keep the CMD line LOW to read all of the boot data.

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes LOW. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode with the CMD line HIGH.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

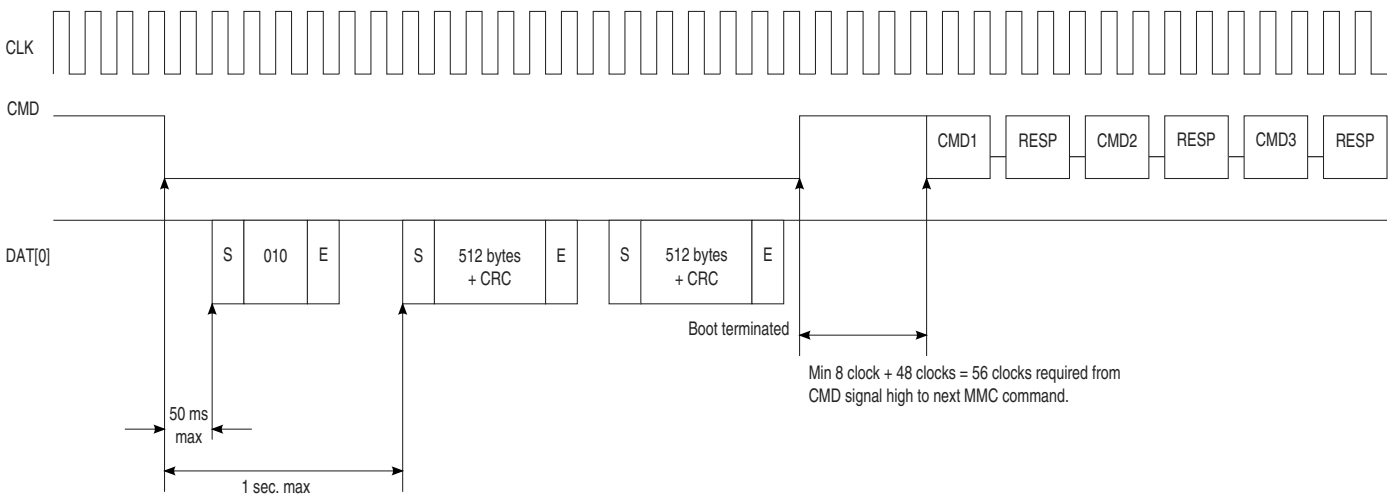


Figure 26-16. MultiMediaCard state diagram (normal boot mode)

26.3.10.2 Alternative Boot Operation

If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

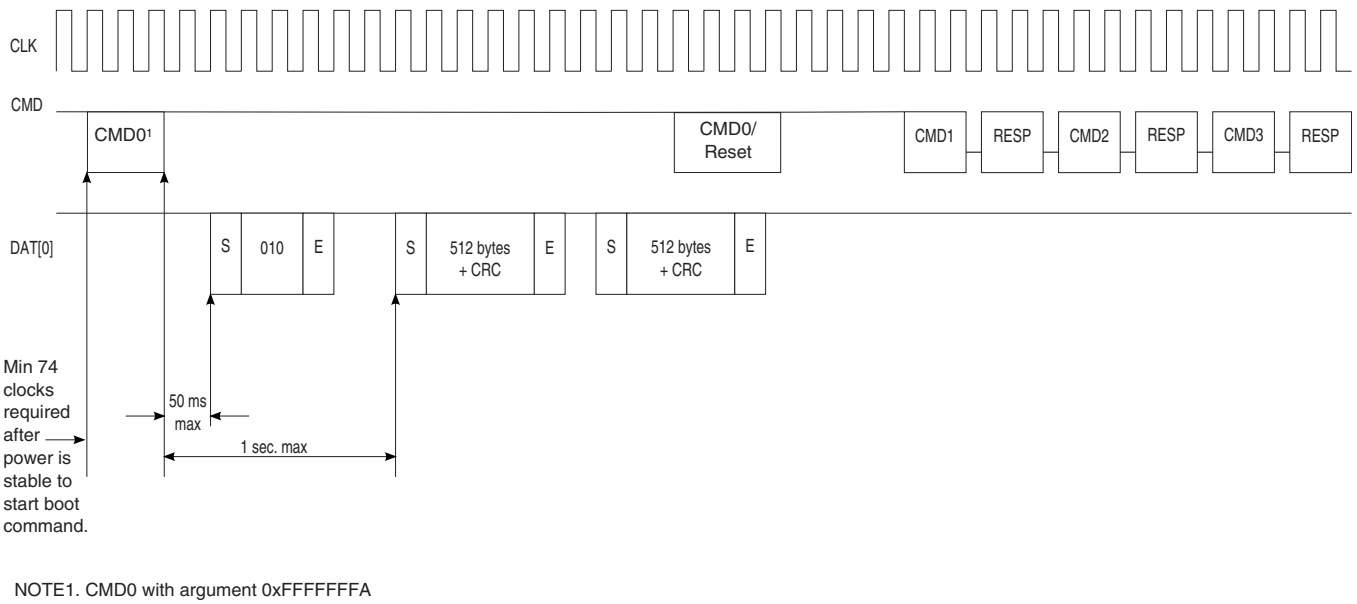
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DAT line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode by issuing CMD0 (Reset).

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



NOTE1. CMD0 with argument 0xFFFFFFFA

Figure 26-17. MultiMediaCard state diagram (alternative boot mode)

26.4 Initialization/Application of ESDHC

All communication between system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as "GO_IDLE_STATE", "SEND_OP_COND", "ALL_SEND_CID" and etc. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for MMC/SD/SDIO for the commands of bc and bcr categories.

After the Broadcast command CMD3 is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DAT I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO for the commands of ac and adtc categories.

26.4.1 Command Send & Response Receive Basic Operation

Assuming the data type WORD is an unsigned 32-bit integer, the below flow is a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
          set MSBSEL bit;
          if (finite block number) {
                    set BCEN bit;
                    if (auto12 command is to use) set AC12EN bit;
          }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. By doing this, make sure the corresponding interrupt status bits are enabled.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and go to the Standby State, no response to the Host when CMD2 is sent. The Host Driver shall deal with 'fake' errors like this with caution.

26.4.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for the MMC cards.

26.4.2.1 Card Detect

See figure below for a flow diagram showing the detection of MMC, SD and SDIO cards using the ESDHC.

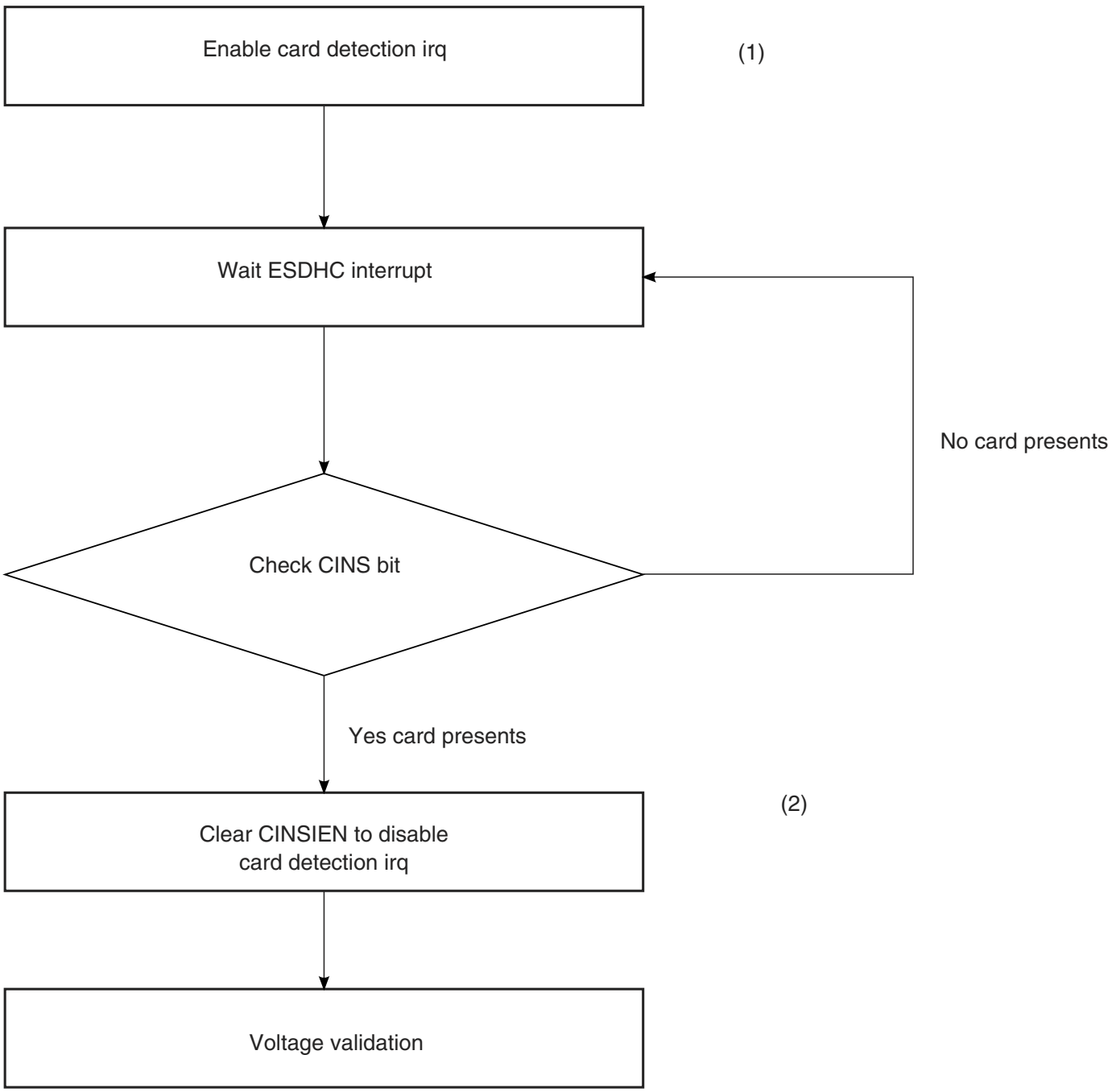


Figure 26-18. Flow Diagram for Card Detection

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt
- When an interrupt from the ESDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards

26.4.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) which is driven by POR (Power On Reset)
- Software reset (Host Only) is proceed by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the Host Controller, respectively
- Card reset (Card Only). The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards, SD Memory cards. This command sets each card into the Idle State regardless of the current card state. For an SD I/O Card, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See figure below for the software flow to reset both the ESDHC and the card.

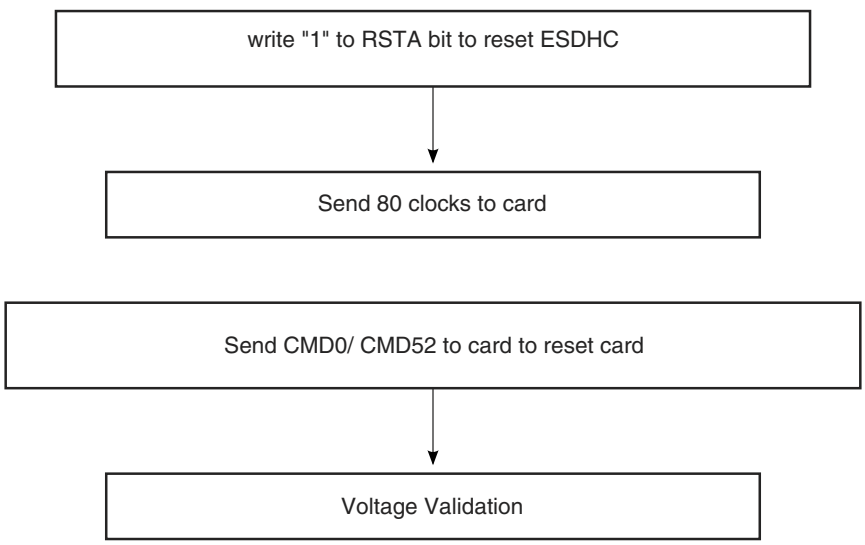


Figure 26-19. Flow Chart for Reset of the ESDHC and SD I/O Card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DVS and SDCLKFS bit fields to get the SD_CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

26.4.2.3 Voltage Validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for Vdd are defined in the Operation Conditions Register (OCR) and may not cover the whole range. Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer Vdd conditions. This means if the host and card have non-common Vdd ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the Vdd range(s) desired by the host. This is accomplished by the host sending the desired Vdd voltage window as the operand of this command. Cards that can't perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification shall be sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
operation voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
          if (0 < number of IO functions) {
                    label the card as SDIO;
                    IORDY = 0;
                    while (!(IORDY in IO OCR response)) { // set voltage range for each IO
function
                              send_command(IO_SEND_OP_COND, <voltage range>, <other
parameter>);
                              wait_for_response(IO_SEND_OP_COND);
                    } // end of while ...
          } // end of if (0 < ...
          if (memory part is present inside SDIO card) Label the card as SDCombo; // this is
an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range
is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific
CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
          send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage
```

```
range
for memory part or SD card
          wait_for_response(SD_APP_OP_COND); // voltage range is set
          if (card type is UNKNOWN) label the card as SD;
          return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
          deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card           if (card is already labelled as
SDCombo) { // change label
                    re-label the card as SDIO;
                    ignore the error or report it;
                    return; // card is identified as SDIO card
          } // of if (card is ...
          send_command(SEND_OP_COND, <voltage range>, <...>);
          if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted,
either
                    label the card as UNKNOWN;
                    return;
          } // of if (RESP_TIMEOUT ...
          if (check for CE-ATA signature succeeded) { // the card is CE-ATA
                    store CE-ATA specific info from the signature;
                    label the card as CE-ATA;
          } // of if (check for CE-ATA ...
          else label the card as MMC;
} // of else
}
```

26.4.2.4 Card Registry

Card registry for the MMC and SD/SDIO/SD Combo cards are different.

For the SD Card, the Identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the Card spec). At this time, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host will request the card to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are put into the Inactive State. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready State), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification State.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA will be used to address the card for future data transfer operations. Once the RCA is received, the card changes its state to the Standby State. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in system.

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (the cards in Ready State) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into the Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

For operation as MMC cards:

```
card_registry()
{
do { // decide RCA for each card until response time-out
          if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
                    send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to
publish its
RCA
                    retrieve RCA from response;
          } // end if (card is labelled as SDCombo ...
          else if (card is labelled as SD) { // for SD card
                    send_command(ALL_SEND_CID, <...>);
                    if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                    send_command(SET_RELATIVE_ADDR, <...>);
                    retrieve RCA from response;
          } // else if (card is labelled as SD ...
          else if (card is labelled as MMC or CE-ATA) { // treat CE-ATA as MMC
                    send_command(ALL_SEND_CID, <...>);
                    rca = 0x1; // arbitrarily set RCA, 1 here for example, this RCA is also
the
relative address to access the CE-ATA card
                    send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper
16
bits
          } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

26.4.3 Card Access

26.4.3.1 Block Write

26.4.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card shall indicate the failure on the DAT line. The transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card will respond with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby State and release the DAT line without interrupting the write operation. When re-selecting the card, it will reactivate the busy indication by pulling DAT to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other two methods (by means of external DMA or ARM platform polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   a. For SD/MMC cards, use SET_BLOCKLEN (CMD16)
   b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the ESDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the ESDHC block attribute register, block number is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the ESDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

26.4.3.1.2 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the SD_CLK at any time to pause all the operations, which is also inaccessible to the Host Driver, the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DAT0 line is not required to de-assert to release the busy state, no suspend command is needed.

Like in the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a. For SD/MMC, use SET_BLOCKLEN (CMD16)
   b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the ESDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the ESDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the ESDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.

7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The Driver shall read the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the Driver should treat this as a non-pause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the Host System is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the Suspend Command for the SDIO card; when such a command is sent, the ESDHC thinks the System will switch to another function on the SDIO card, and flush the data buffer. The ESDHC takes the Resume Command as a normal command with data transfer, and it is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the ESDHC will automatically send a CMD12 to mark the end of the multi-block transfer.

26.4.3.2 Block Read

26.4.3.2.1 Normal Read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer State. For multi blocks read, data blocks will be continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other two methods (by means of external DMA or ARM platform polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a. For SD/MMC, use SET_BLOCKLEN (CMD16)
   b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the ESDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the ESDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the ESDHC DMA when sending the command with data transfer. The AC12EN bit should also be set:
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

26.4.3.2.2 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks. Before setting the SABGREQ bit, make sure the RWCTL bit in the Protocol Control register is set, otherwise the ESDHC will not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended that the RWCTL bit be set once the Read Wait capability of the SDIO card is recognized.

Like in the flow described in Normal Read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports Read Wait.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:

a. For SD/MMC, use SET_BLOCKLEN (CMD16)
b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)

5. Set the ESDHC block length register to be the same as the block length set for the card in Step 2.
6. Set the ESDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the ESDHC DMA when sending the command with data transfer. The AC12EN bit should also be set
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the ESDHC will ignore the Stop At Block Gap Request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause will be transferred to the Host System. No matter if the Suspend Command is sent or not, the internal data buffer is not flushed.

If the Suspend Command is sent and the transfer is later resumed by means of a Resume Command, the ESDHC takes the command as a normal one accompanied with data transfer. It is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the ESDHC will automatically send the CMD12 to mark the end of multi-block transfer.

26.4.3.3 Suspend Resume

The ESDHC supports the Suspend Resume operations of SDIO cards, although slightly different than the suggested implementation of Suspend in the SDIO card specification.

Suspend

After setting the SABGREQ bit, the Host Driver may send a Suspend command to switch to another function of the SDIO card. The ESDHC does not monitor the content of the response, so it doe not know whether or not the Suspend command has succeeded. Accordingly, it does not de-assert Read Wait for read pause. To solve this problem, the Driver does not set the ESDHC_XFERTYP[CMDTYP] register to 01, that is, Suspend option. Instead, the Driver sends this command as if it were a normal command (that is, sets CMDTYP to b00). Only when the command succeeds, and the BS bit is set in the response, does the Driver send another command marked as "Suspend" to inform the ESDHC that the current transfer is suspended. This is shown in the following sequence for Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the ESDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the ESDHC stops driving DAT2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

26.4.3.3.1 Resume

To resume the data transfer, a Resume command shall be issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation above.
2. Send the Resume command. In the Transfer Type register, all bit fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer will be resumed.

26.4.3.4 ADMA1 Usage

To use the ADMA1 in a data transfer, the Host Driver must prepare the correct descriptor chain prior to sending the read/write command. The steps to accomplish this are:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4kB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.
4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

26.4.3.5 Transfer Error

26.4.3.5.1 CRC Error

It is possible at the end of a block transfer, that a write CRC status error or read CRC error occurs. For this type of error the latest block received shall be discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one. For a multi-block transfer, the Host Driver shall issue a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, the ESDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 will be sent by the ESDHC. In this case, the Driver shall re-send or re-obtain the last block with a single block transfer.

26.4.3.5.2 Internal DMA Error

During the data transfer with internal Simple DMA, if the DMA engine encounters some error on the AHB bus, the DMA operation is aborted and DMA Error interrupt is sent to the Host System. When acknowledged by such an interrupt, the Driver shall calculate the start address of data block in which the error occurs. The start address can be calculated by either:

1. Read the DMA System Address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the next burst transfer into account, it is straight forward to obtain the start address of the corrupted block.
2. Read the BLKCNT field of the Block Attribute register. By the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the Block Attribute register does not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and re-start the transfer from the corrupted block to recover from the error.

26.4.3.5.3 ADMA Error-Card Access

There are three kinds of possible ADMA errors. The AHB transfer, invalid descriptor, and data-length mismatch errors. When these errors occur, the DMA transfer stops and the corresponding error status bit is set. For acknowledging the status, the Host Driver should recover the error as shown below and re-transfer from the place of interruption.

1. AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or the next block if no block is corrupted.
2. Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and re-create the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
3. Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain, and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

26.4.3.5.4 Auto CMD12 Error

After the last block of the multi block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, the ESDHC automatically sends a CMD12 to the card to stop the transfer. When errors with this command occur, it is recommended to the Driver to deal with the situations in the following manner:

1. Auto CMD12 response time-out. It is not certain whether the command is accepted by the card or not. The Driver should clear the Auto CMD12 error status bits and re-send the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error. Since card responds to the CMD12, the card will abort the transfer. The Driver may ignore the error and clear the error status bit.
3. Auto CMD12 conflict error or not sent. The command is not sent, so the Driver shall send a CMD12 manually.

26.4.3.6 Card Interrupt

The external cards can inform the Host Controller by means of some special signals. For the SDIO card, it can be the low level on the DAT[1] line during some special period. The ESDHC only monitors the DAT[1] line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the ESDHC, and the Host System is informed by the ESDHC asserting the ESDHC interrupt line, the interrupt service from the Host Driver is called.

As the interrupt factor is controlled by the external card, the interrupt from the SDIO card must be served before the CINT bit is cleared by written 1. Refer to Card Interrupt Handling for the card interrupt handling flow.

26.4.4 Switch Function

MMC cards transferring data at bus widths other than 1-bit is a feature in MMC spec. The high speed timing mode for all card devices, was also defined in various card specifications. To enable these features, a "switch" command shall be issued by the Host Driver.

For SDIO cards, the high speed mode is enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode is queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards, the high speed mode is queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The 4-bit and 8-bit bus width of the MMC is also enabled by the SWITCH command, but with a different argument.

DDR mode is also selected by setting ddr mode bus width via SWITCH.

These new functions can also be disabled by a software reset. For SDIO cards it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following flowcharts do not show current capability check, which is recommended in the function switch process.

26.4.4.1 Query, Enable and Disable SDIO High Speed Mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
cleared;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

26.4.4.2 Query, Enable and Disable SD High Speed Mode

```
enable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFF1 and read 64 bytes of data accompanying the R1 response;
wait data transfer done bit is set;
check if the bit 401 of received 512 bit is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
send CMD6, with argument 0x80FFFFF1 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

26.4.4.3 Query, Enable and Disable MMC High Speed Mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

26.4.4.4 Set MMC Bus Width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

26.4.5 ADMA Operation

26.4.5.1 ADMA1 Operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
```

```
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB align);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

26.4.5.2 ADMA2 Operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

26.4.6 Fast Boot Operation

26.4.6.1 Normal fast boot flow

1. Software must configure INITA bit ESDHC_SYSCTL[27] to make sure 74 card clocks are finished.

2. Software must configure MMC Boot Register (ESDHC_MMCBOOT) (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode or not. If it is necessary to send through DMA mode, bit 7 must be configured to enable automatic stop at block gap feature. And need to configure bit 3-bit0 must be configured to select the ack timeout value according to the sd clk frequence.
3. Software must then configure Block Attributes Register to set block size/no.
4. Software must configure Protocol control register to set DTW (data transfer width).
5. Software must configure Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.

NOTE

DMAEN should be configured as 0 in polling mode. If BCEN is configured as 1, it is best to configure blk no in Bock Attributes Register to the max value.

7. When step 6 is configured, the boot process will begin. Software must poll the data buffer ready status to read the data from buffer in time. If boot time-out happens (ack time out or the first data read time out), interrupt will be triggered, and software must configure MMC Boot Register, bit 6 to 0 to disable boot. This will make CMD high, and after at least 56 clocks, it is ready to begin normal initialization process.
8. If no time out occurs, the software must decide if the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This will make CMD line high and command completed is asserted. After at least 56 clock cycles, it is ready to begin normal initialization process.
9. Reset the host and then can begin the normal process.

26.4.6.2 Alternative fast boot flow

1. Software must configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software must configure MMC Boot Register (ESDHC_MMCBOOT) (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If it is necessary to send through DMA mode, bit 7 must be configured to enable automatic stop at block gap feature. Bit 3 through bit0 must be configured to select the ack timeout value according to the sd clk frequence.
3. Software must then configure Block Attributes Register to set block size/no.
4. Software mustconfigure Protocol control register to set DTW (data transfer width).

5. Software must configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software must configure Transfer Type Register to start the boot process by CMD0 with 0xFFFFFFFA argument . In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.

NOTE

DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1 in polling mode, it is best to configure blk no in Bock Attributes Register to the max value.

7. When the step 6 is configured, boot process begins. Software must poll the data buffer ready status to read the data from buffer in time. If boot time out (acknowledge data time out in 50ms or data time out in 1s), host will send out the interrupt and software need to send CMD0 with reset and then configure boot enable bit to 0 to stop this process. After command completed, configure MMC Boot Register bit 6 to 0 to disable boot. After at least 8 clocks from command completed, card is ready for identification step.
8. If no time out occurs, software must decide when to stop the boot process, and send out the CMD0 with reset. After this command is completed, configure MMC Boot Register bit 6 to stop the process. After 8 clocks from command are completed, the slave (card) is ready for the identification step.
9. Reset the host and then can begin the normal process.

26.4.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the

beginning of the image, DMA parameters must be switched on the fly during MMC fast boot.

In fast boot, host can use ADMA2 (Advanced DMA2) with two destinations.

The detail flow:

1. Software must configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software must configure MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot), to 1 (alternative boot); and bit 4 to select the acknowledge mode or not. In DMA mode, configure bit 7 to 1 for enable

automatically stop at block gap feature. Also configure bit31-bit16 to set the VALUE1 (value of block count that need to transfer first time), that host will stop at block gap when card block counter is equal to this value. Bit 3 through bit0 must be configured to select the acknowledge timeout value according to the SD clock frequency.

3. Software must then configure Block Attributes Register to set block size/no. In DMA mode, it is better to set block number to the max value (16'hFFFF).
4. Software must configure Protocol control register to set DTW (data transfer width).
5. Software enbables ADMA2 by configuring protocol control register bit9-bit8.
6. Software must set at least three pairs ADMA2 descriptor in boot memory (i.e. in IRAM, at least 6 words). The first pair descriptor define the start address (i.e. IRAM) and data length (that is, 512 bytes*VALUE1) of first part boot code. Software also must set the second pair descriptor, the second start address (any value that is writeable), data length is suggest to set 1~2word (record as VAULE2).

NOTE

The second couple descriptor also transfers useful data. (At lease 1 word because the ADMA2 can't support a 0 data_length data transfer descriptor.)

7. Software must configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot, and do not need set in normal fast boot.
8. Software must configure Transfer Type Register to start the boot process. CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, and MSBSEL is set to 1. DMAEN is configured as 1 in DMA mode. And if BCEN is configured as 1, it is best to configure block no in Bock Attributes Register to the max value.
9. When the step 8 is configured, boot process will begin. The first VAULE1 block number data has transfer. Software must poll TC bit (bit1 in Interrupt Status Register) to determine if the first transfer has ended. Software must poll BGE bit (bit2 in Interrupt Status Register) to determine if first transfer has stopped at block gap.
10. When TC, BGE bit is 1, SW can analyze the first code of VAULE1 block, initialize the new memory device, if required, and set the third pair of descriptors to define the start address and length of the remaining part of boot code (VAULE3 the remain boot code block). Remember to set the last descriptor with END.
11. Software must configure MMC Boot Register (ESDHC_MMCBOOT) (offset 0xc4) again. Set bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot), to 1 (alternative boot); and bit 4 to select the acknowledge mode or not. In DMA mode, configure bit 7 to 1 to enable automatic stop at block gap feature. Bit31-bit16 must be configured to set the (VAULE1+1+VAULE3) so that host will stop at block gap when card

block counter is equal to this value. And must configure bit 3-bit0 to select the acknowledge timeout value according to the SD clock frequency.

12. Software must clear TC and BGE bit. It must also clear SABGREQ (bit 16 in Protocol control register), and set CREQ (bit17 Protocol control register) to 1 to resume the data transfer. Host will transfer the VALUE2 and VAULE3 data to the destination that is set by descriptor.
13. Software must poll BGE bit to determine if the fast boot is over.

NOTE

When ADMA boot flow is started, for ESDHC, it is like a normal ADMA read operation. In order to keep descriptor, must have a memory length of at least a few words.For the 1~2 word data in second descriptor setting, it is the useful data, so software need to deal the data due to the application case.

26.5 Commands for MMC/SD/SDIO

See table below for the list of commands for the MMC/SD/SDIO cards.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

1. broadcast commands (bc), no response.
2. broadcast commands with response (bcr), response from all cards simultaneously.
3. addressed (point-to-point) commands (ac), no data transfer on the DAT.
4. addressed (point-to-point) data transfer commands (adtc).

The Access Bits for the EXT_CSD Access Modes are shown in table below.

Table 26-2. Commands for MMC/SD/SDIO Cards

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather that a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57~59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 shall be used to read and write these registers.
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62~63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac		R1	SET_WR_BLK_ERASE_COUNT	-
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac		R1	SET_CLR_CARD_DETECT	-
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed MMC cards . The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 26-3.
4. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this block).

Table 26-3. EXT_CSD Access Modes

Bits	Access Name	Operation
00	Command Set	The command set is changed according to the Cmd Set field of the argument
01	Set Bits	The bits in the pointed byte are set, according to the 1 bits in the Value field.
10	Clear Bits	The bits in the pointed byte are cleared, according to the 1 bits in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

26.6 Software Restrictions

26.6.1 Initialization Active

The driver cannot set INITA bit in System Control register when any of the command line or data lines is active, so the driver must ensure both CDIHB and CIHB bits are cleared. In order to auto clear the INITA bit, the SDCLKEN bit must be '1', otherwise no clocks can go out to the card and INITA will never clear.

26.6.2 Softeware Polling Procedure

For polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not the times of the value in Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block. For example, for read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

26.6.3 Suspend Operation

In order to suspend the data transfer, the software must inform ESDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform ESDHC that the transfer is suspended.

If software needs to resume the suspended trasnfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending a 'suspend' command, ESDHC will regard the current transfer as aborted and change BLKCNT register to its original value, rather than keeping the remaining number of blocks.

26.6.4 Data Length Setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

26.6.5 (A)DMA Address Setting

To configure ADMA1/ADMA2/DMA address register, when TC bit is set, the register will always update itself with the internal address value to support dynamic address synchronization, so software must make sure TC bit is cleared prior to configuring ADMA1/ADMA2/DMA address register.

26.6.6 Data Port Access

Data Port does not support parallel access. For example, during an external DMA access, it is not allowed to write any data to the Data Port by ARM platform; or during a ARM read operation, it is also prohibited to write any data to the Data Port, by either ARM or external DMA. Otherwise the data would be corrupted inside the ESDHC buffer.

26.6.7 Change Clock Frequency

ESDHC does not automatically gates off the card clock when the Host Driver changes the clock frequency. To remove possible glitch on the card clock, clear SDCLKEN bit when changing clock divisor value and set SDCLKEN bit to '1' after SDSTB bit is '1' again.

26.6.8 Multi-block Read

For pre-defined multi-block read operation, that is, the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by ESDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode. In this case, the card may not respond to this extra abort command and ESDHC will get Response Timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

26.7 Programmable Registers

This section includes the block memory map and detailed descriptions of all registers. Each of these registers support only 32-bit accesses.

NOTE

Addresses greater than 0x44, except 0x50, 0x54, 0x58, 0x60, 0x64, 0xC0, 0xC4 and 0xFC, are reserved and read as all 0s. Write to these registers is ignored.

ESDHCV2 memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5000_4000	DMA System Address (ESDHCV2-1_DSADDR)	32	R/W	0000_0000h	26.7.1/ 1302
5000_4004	Block Attributes (ESDHCV2-1_BLKATTR)	32	R/W	0000_0000h	26.7.2/ 1303
5000_4008	Command Argument (ESDHCV2-1_CMDARG)	32	R/W	0000_0000h	26.7.3/ 1304
5000_400C	Command Transfer Type (ESDHCV2-1_XFERTYP)	32	R/W	0000_0000h	26.7.4/ 1305
5000_4010	Command Response 0 (ESDHCV2-1_CMDRSP0)	32	R	0000_0000h	26.7.5/ 1309
5000_4014	Command Response 1 (ESDHCV2-1_CMDRSP1)	32	R	0000_0000h	26.7.6/ 1310
5000_4018	Command Response 2 (ESDHCV2-1_CMDRSP2)	32	R	0000_0000h	26.7.7/ 1310
5000_401C	Command Response 3 (ESDHCV2-1_CMDRSP3)	32	R	0000_0000h	26.7.8/ 1310
5000_4020	Data Buffer Access Port (ESDHCV2-1_DATPORT)	32	R/W	0000_0000h	26.7.9/ 1312
5000_4024	Present State (ESDHCV2-1_PRSSTAT)	32	R	0000_0000h	26.7.10/ 1313
5000_4028	Protocol Control (ESDHCV2-1_PROCTL)	32	R/W	0000_0000h	26.7.11/ 1318
5000_402C	System Control (ESDHCV2-1_SYSCTL)	32	R/W	0000_8008h	26.7.12/ 1322
5000_4030	Interrupt Status (ESDHCV2-1_IRQSTAT)	32	w1c	0000_0000h	26.7.13/ 1326
5000_4034	Interrupt Status Enable (ESDHCV2-1_IRQSTATEN)	32	R/W	117F_013Fh	26.7.14/ 1331
5000_4038	Interrupt Signal Enable (ESDHCV2-1_IRQSIGEN)	32	R/W	0000_0000h	26.7.15/ 1334
5000_403C	Auto CMD12 Status (ESDHCV2-1_AUTOC12ERR)	32	R	0000_0000h	26.7.16/ 1336
5000_4040	Host Controller Capabilities (ESDHCV2-1_HOSTCAPBLT)	32	R	07F3_0000h	26.7.17/ 1339
5000_4044	Watermark Level (ESDHCV2-1_WML)	32	R/W	0810_0810h	26.7.18/ 1341
5000_4050	Force Event (ESDHCV2-1_FEVT)	32	W (always reads zero)	0000_0000h	26.7.19/ 1342
5000_4054	ADMA Error Status Register (ESDHCV2-1_ADMAES)	32	R	0000_0000h	26.7.20/ 1344

Table continues on the next page...

ESDHCV2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5000_4058	ADMA System Address (ESDHCV2-1_ADSADDR)	32	R/W	0000_0000h	26.7.21/ 1346
5000_40C0	Vendor Specific Register (ESDHCV2-1_VENDOR)	32	R/W	0000_0001h	26.7.22/ 1347
5000_40C4	MMC Boot Register (ESDHCV2-1_MMCBOOT)	32	R/W	0000_0000h	26.7.23/ 1348
5000_40FC	Host Controller Version (ESDHCV2-1_HOSTVER)	32	R	0000_1201h	26.7.24/ 1349
5000_8000	DMA System Address (ESDHCV2-2_DSADDR)	32	R/W	0000_0000h	26.7.1/ 1302
5000_8004	Block Attributes (ESDHCV2-2_BLKATTR)	32	R/W	0000_0000h	26.7.2/ 1303
5000_8008	Command Argument (ESDHCV2-2_CMDARG)	32	R/W	0000_0000h	26.7.3/ 1304
5000_800C	Command Transfer Type (ESDHCV2-2_XFERTYP)	32	R/W	0000_0000h	26.7.4/ 1305
5000_8010	Command Response 0 (ESDHCV2-2_CMDRSP0)	32	R	0000_0000h	26.7.5/ 1309
5000_8014	Command Response 1 (ESDHCV2-2_CMDRSP1)	32	R	0000_0000h	26.7.6/ 1310
5000_8018	Command Response 2 (ESDHCV2-2_CMDRSP2)	32	R	0000_0000h	26.7.7/ 1310
5000_801C	Command Response 3 (ESDHCV2-2_CMDRSP3)	32	R	0000_0000h	26.7.8/ 1310
5000_8020	Data Buffer Access Port (ESDHCV2-2_DATPORT)	32	R/W	0000_0000h	26.7.9/ 1312
5000_8024	Present State (ESDHCV2-2_PRSSTAT)	32	R	0000_0000h	26.7.10/ 1313
5000_8028	Protocol Control (ESDHCV2-2_PROCTL)	32	R/W	0000_0000h	26.7.11/ 1318
5000_802C	System Control (ESDHCV2-2_SYSCTL)	32	R/W	0000_8008h	26.7.12/ 1322
5000_8030	Interrupt Status (ESDHCV2-2_IRQSTAT)	32	w1c	0000_0000h	26.7.13/ 1326
5000_8034	Interrupt Status Enable (ESDHCV2-2_IRQSTATEN)	32	R/W	117F_013Fh	26.7.14/ 1331
5000_8038	Interrupt Signal Enable (ESDHCV2-2_IRQSIGEN)	32	R/W	0000_0000h	26.7.15/ 1334
5000_803C	Auto CMD12 Status (ESDHCV2-2_AUTOC12ERR)	32	R	0000_0000h	26.7.16/ 1336

Table continues on the next page...

ESDHCV2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5000_8040	Host Controller Capabilities (ESDHCV2-2_HOSTCAPBLT)	32	R	07F3_0000h	26.7.17/ 1339
5000_8044	Watermark Level (ESDHCV2-2_WML)	32	R/W	0810_0810h	26.7.18/ 1341
5000_8050	Force Event (ESDHCV2-2_FEVT)	32	W (always reads zero)	0000_0000h	26.7.19/ 1342
5000_8054	ADMA Error Status Register (ESDHCV2-2_ADMAES)	32	R	0000_0000h	26.7.20/ 1344
5000_8058	ADMA System Address (ESDHCV2-2_ADSADDR)	32	R/W	0000_0000h	26.7.21/ 1346
5000_80C0	Vendor Specific Register (ESDHCV2-2_VENDOR)	32	R/W	0000_0001h	26.7.22/ 1347
5000_80C4	MMC Boot Register (ESDHCV2-2_MMCBOOT)	32	R/W	0000_0000h	26.7.23/ 1348
5000_80FC	Host Controller Version (ESDHCV2-2_HOSTVER)	32	R	0000_1201h	26.7.24/ 1349
5002_4000	DMA System Address (ESDHCV2-4_DSADDR)	32	R/W	0000_0000h	26.7.1/ 1302
5002_4004	Block Attributes (ESDHCV2-4_BLKATTR)	32	R/W	0000_0000h	26.7.2/ 1303
5002_4008	Command Argument (ESDHCV2-4_CMDARG)	32	R/W	0000_0000h	26.7.3/ 1304
5002_400C	Command Transfer Type (ESDHCV2-4_XFERTYP)	32	R/W	0000_0000h	26.7.4/ 1305
5002_4010	Command Response 0 (ESDHCV2-4_CMDRSP0)	32	R	0000_0000h	26.7.5/ 1309
5002_4014	Command Response 1 (ESDHCV2-4_CMDRSP1)	32	R	0000_0000h	26.7.6/ 1310
5002_4018	Command Response 2 (ESDHCV2-4_CMDRSP2)	32	R	0000_0000h	26.7.7/ 1310
5002_401C	Command Response 3 (ESDHCV2-4_CMDRSP3)	32	R	0000_0000h	26.7.8/ 1310
5002_4020	Data Buffer Access Port (ESDHCV2-4_DATPORT)	32	R/W	0000_0000h	26.7.9/ 1312
5002_4024	Present State (ESDHCV2-4_PRSSTAT)	32	R	0000_0000h	26.7.10/ 1313
5002_4028	Protocol Control (ESDHCV2-4_PROCTL)	32	R/W	0000_0000h	26.7.11/ 1318
5002_402C	System Control (ESDHCV2-4_SYSCTL)	32	R/W	0000_8008h	26.7.12/ 1322

Table continues on the next page...

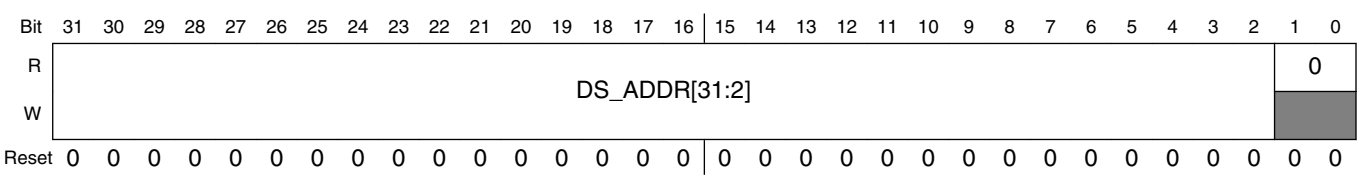
ESDHCV2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5002_4030	Interrupt Status (ESDHCV2-4_IRQSTAT)	32	w1c	0000_0000h	26.7.13/ 1326
5002_4034	Interrupt Status Enable (ESDHCV2-4_IRQSTATEN)	32	R/W	117F_013Fh	26.7.14/ 1331
5002_4038	Interrupt Signal Enable (ESDHCV2-4_IRQSIGEN)	32	R/W	0000_0000h	26.7.15/ 1334
5002_403C	Auto CMD12 Status (ESDHCV2-4_AUTOC12ERR)	32	R	0000_0000h	26.7.16/ 1336
5002_4040	Host Controller Capabilities (ESDHCV2-4_HOSTCAPBLT)	32	R	07F3_0000h	26.7.17/ 1339
5002_4044	Watermark Level (ESDHCV2-4_WML)	32	R/W	0810_0810h	26.7.18/ 1341
5002_4050	Force Event (ESDHCV2-4_FEVT)	32	W (always reads zero)	0000_0000h	26.7.19/ 1342
5002_4054	ADMA Error Status Register (ESDHCV2-4_ADMAES)	32	R	0000_0000h	26.7.20/ 1344
5002_4058	ADMA System Address (ESDHCV2-4_ADSADDR)	32	R/W	0000_0000h	26.7.21/ 1346
5002_40C0	Vendor Specific Register (ESDHCV2-4_VENDOR)	32	R/W	0000_0001h	26.7.22/ 1347
5002_40C4	MMC Boot Register (ESDHCV2-4_MMCBOOT)	32	R/W	0000_0000h	26.7.23/ 1348
5002_40FC	Host Controller Version (ESDHCV2-4_HOSTVER)	32	R	0000_1201h	26.7.24/ 1349

26.7.1 DMA System Address (ESDHCV2x_DSADDR)

This register contains the physical system memory address used for DMA transfers.

Addresses: ESDHCV2-1_DSADDR is 5000_4000h base + 0h offset = 5000_4000h
ESDHCV2-2_DSADDR is 5000_8000h base + 0h offset = 5000_8000h
ESDHCV2-4_DSADDR is 5002_4000h base + 0h offset = 5002_4000h



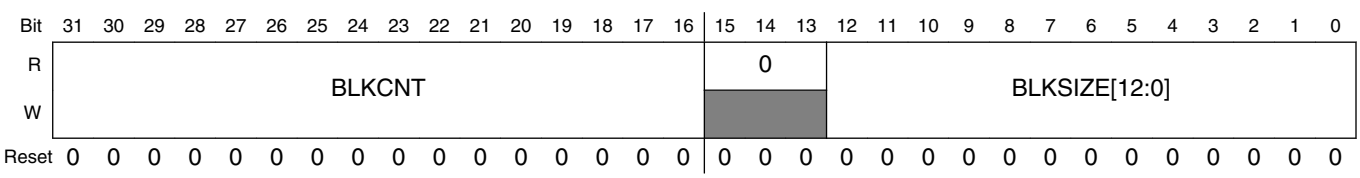
ESDHCV2x_DSADDR field descriptions

Field	Description
31–2 DS_ADDR[31:2]	DMA System Address: This register contains the 32-bit system memory address for a DMA transfer. As the address must be word (4 bytes) align, the least 2 bits are reserved, always 0. When the ESDHC stops a DMA transfer, this register points to the system address of the next contiguous data position. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operation during transfers may return an invalid value. The Host Driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The Host driver shall wait, until ESDHCV2_PRSSTAT[DLA] bit is cleared, before writing to this register. The ESDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, ESDHC will automatically change SEQ burst type to NSEQ. As this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions.
1–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

26.7.2 Block Attributes (ESDHCV2x_BLKATTR)

This register is used to configure the number of data blocks and the number of bytes in each block.

Addresses: ESDHCV2-1_BLKATTR is 5000_4000h base + 4h offset = 5000_4004h
ESDHCV2-2_BLKATTR is 5000_8000h base + 4h offset = 5000_8004h
ESDHCV2-4_BLKATTR is 5002_4000h base + 4h offset = 5002_4004h



ESDHCV2x_BLKATTR field descriptions

Field	Description
31–16 BLKCNT	Blocks Count For Current Transfer: This register is enabled when ESDHCV2_XFERTYP[BCEN] bit is set to 1 and is valid only for multiple block transfers. For single block transfer, this register will always read as 1. The Host Driver shall set this register to a value between 1 and the maximum block count. The ESDHC decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred. This register should be accessed only when no transaction is executing (that is, after transactions have stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored.

Table continues on the next page...

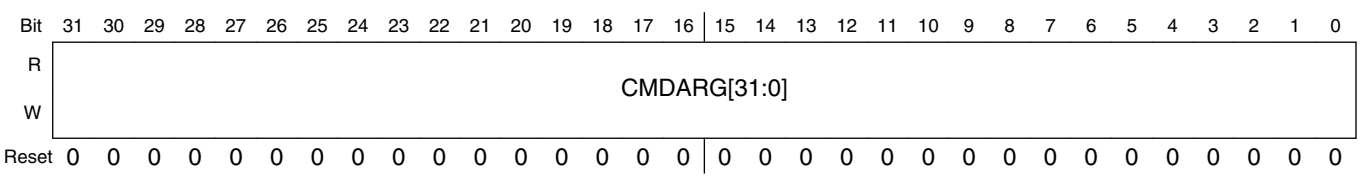
ESDHCV2x_BLKATTR field descriptions (continued)

Field	Description
	When saving transfer content as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend. This is because when Suspend command is sent out, ESDHC will regard the current transfer is aborted and change BLKCNT register back to its original value instead of keeping the dynamical indicator of remained block count. When restoring transfer content prior to issuing a Resume command, the Host Driver shall restore the previously saved block count. 0xFFFF 65535 blocks 0x0002 2 blocks 0x0001 1 block 0x0000 Stop Count
15–13 Reserved	This read-only field is reserved and always has the value zero. Reserved
12–0 BLKSIZE[12:0]	Transfer Block Size: This register specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations will be ignored. 0x1000 4096 Bytes 0x800 2048 Bytes 0x200 512 Bytes 0x1FF 511 Bytes 0x004 4 Bytes 0x003 3 Bytes 0x002 2 Bytes 0x001 1 Byte 0x000 No data transfer

26.7.3 Command Argument (ESDHCV2x_CMDARG)

This register contains the SD/MMC Command Argument.

Addresses: ESDHCV2-1_CMDARG is 5000_4000h base + 8h offset = 5000_4008h
ESDHCV2-2_CMDARG is 5000_8000h base + 8h offset = 5000_8008h
ESDHCV2-4_CMDARG is 5002_4000h base + 8h offset = 5002_4008h



ESDHCV2x_CMDARG field descriptions

Field	Description
31–0 CMDARG[31:0]	Command Argument: The SD/MMC Command Argument is specified as bits 39-8 of the Command Format in the SD or MMC Specification.This register is write protected when the Command Inhibit (CMD) bit in the Present State register is set.

26.7.4 Command Transfer Type (ESDHCV2x_XFERTYP)

This register is used to control the operation of data transfers. The Host Driver sets this register before issuing a command followed by a data transfer, or before issuing a Resume command. To prevent data loss, the ESDHC prevents writing to the bits, that are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

The Host Driver shall check the Command Inhibit DAT bit (CDIHB) and the Command Inhibit CMD bit (CIHB) in the Present State register before writing to this register. When the CDIHB bit in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB bit is set, any write to this register is ignored.

On sending commands with data transfer involved, it is mandatory that the block size is non-zero. Besides, block count must also be non-zero, or indicated as single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise ESDHC will ignore the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active (WPSPL bit of Present State Register is '1), otherwise ESDHC will also ignore the command.

If the commands with data transfer does not receive the response in 64 clock cycles, that is, response time-out, ESDHC will regard the external device does not accept the command and abort the data transfer. In this scenario, the driver should issue the command again to re-try the transfer. It is also possible that for some reason the card responds the command but ESDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The CMDTYP field of this register is used to initiate three special commands these are:

- Suspend: ESDHC does not monitor the content of the Suspend command response and therefore assumes that the commnand succeeds when issued. It then operates assuming the card bus has been released and that it is permissible to issue the next command using the DAT line. It is the responsibility of S/W to check the status of

the Suspend command and send another command marked as Suspend to inform the ESDHC that a Suspend command was successfully issued. Refer to Suspend Resume for more details. After the end bit of command is sent, ESDHC de-asserts Read Wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the Suspend command fails, the ESDHC will maintain its current state and the Host Driver shall restart the transfer by setting the Continue Request bit in the Protocol Control register.

- Resume: S/W re-starts the data transfer by restoring the registers saved before sending the Suspend Command and then sends the Resume Command. The ESDHC will check for a pending busy state before starting write transfers.
- Abort: If this command is set when executing a read transfer, ESDHC will stop reads to the buffer. If this command is set when executing a write transfer, ESDHC will stop driving the DAT line. After issuing the Abort command, the Host Driver should issue a software reset (Abort Transaction).

Command Transfer Type (ESDHCV2_XFERTYP) shows the summary of how register settings determine the type of data transfer.

Table 26-42. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

Table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, in regards to the Response Type bits and the name of the response type.

Table 26-43. Relationship Between Parameters and the Name of the Response Type

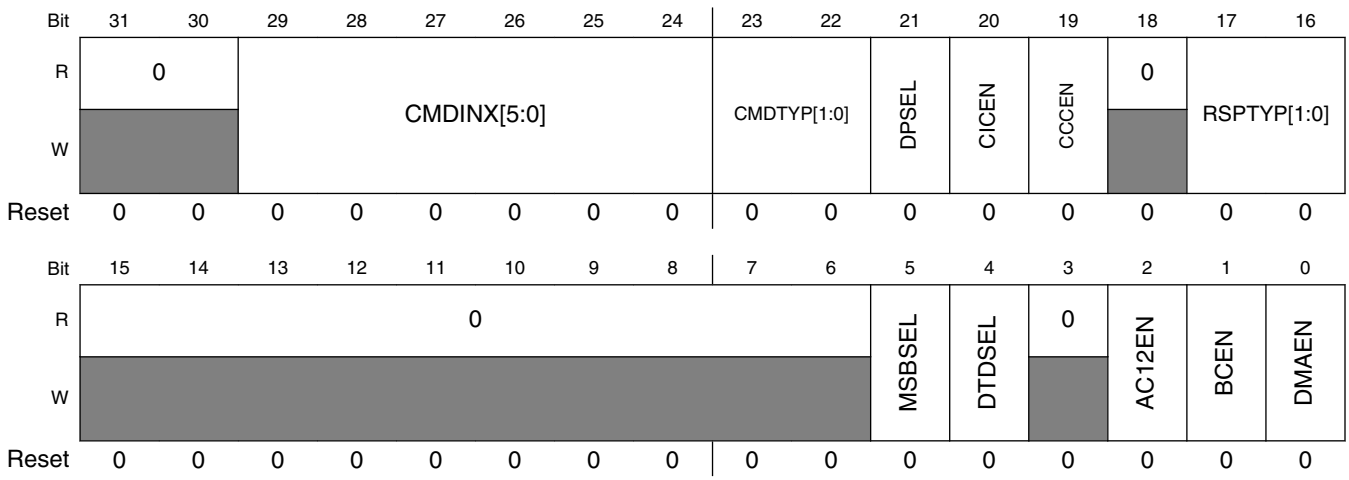
Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3,R4
10	1	1	R1,R5,R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to indicate

that ESDHC will check the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command will be used with R5b.

- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check shall be disabled for these response types.

Addresses: ESDHCV2-1_XFERTYP is 5000_4000h base + Ch offset = 5000_400Ch
ESDHCV2-2_XFERTYP is 5000_8000h base + Ch offset = 5000_800Ch
ESDHCV2-4_XFERTYP is 5002_4000h base + Ch offset = 5002_400Ch



ESDHCV2x_XFERTYP field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value zero. Reserved
29–24 CMDINX[5:0]	Command Index: These bits shall be set to the command number that is specified in bits 45-40 of the Command-Format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
23–22 CMDTYP[1:0]	Command Type: 11 Abort CMD12, CMD52 for writing I/O Abort in CCCR 10 Resume CMD52 for writing Function Select in CCCR 01 Suspend CMD52 for writing Bus Suspend in CCCR 00 Normal mode. Used for all other commands
21 DPSEL	Data Present Select: This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. It is set to 0 for the following: • Commands using only the CMD line (for example, CMD52). • Commands with no data transfer, but using the busy signal on DAT[0] line (R1b or R5b such as, CMD38) **NOTE:** In resume command, this bit shall be set, and other bits in this register shall be set the same as when the transfer was initially launched. When the Write Protect switch is on, (that is, the WPSPL bit is active as '0'), any command with a write operation will be ignored. That is to say, when this bit is set, while the DTDSEL bit is 0, writes to the register Transfer Type are ignored.

Table continues on the next page...

ESDHCV2x_XFERTYP field descriptions (continued)

Field	Description
	1 Data Present 0 No Data Present
20 CICEN	Command Index Check Enable: If this bit is set to 1, the ESDHC will check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked. 1 Enable 0 Disable
19 CCCEN	Command CRC Check Enable: If this bit is set to 1, the ESDHC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. (Refer to RSPTYP[1:0] and Command Transfer Type (ESDHCV2_XFERTYP).) 1 Enable 0 Disable
18 Reserved	This read-only field is reserved and always has the value zero. Reserved
17–16 RSPTYP[1:0]	Response Type Select: 00 No Response 01 Response Length 136 10 Response Length 48 11 Response Length 48, check Busy after response
15–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5 MSBSEL	Multi / Single Block Select: This bit enables multiple block DAT line data transfers. For any other commands, this bit shall be set to 0. If this bit is 0, it is not necessary to set the Block Count register. (See Command Transfer Type (ESDHCV2_XFERTYP).) 1 Multiple Blocks 0 Single Block
4 DTDSEL	Data Transfer Direction Select: This bit defines the direction of DAT line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the ESDHC and is set to 0 for all other commands. 1 Read (Card to Host) 0 Write (Host to Card)
3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2 AC12EN	Auto CMD12 Enable:

Table continues on the next page...

ESDHCV2x_XFERTYP field descriptions (continued)

Field	Description
	Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the ESDHC will issue a CMD12 automatically when the last block transfer has completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, the ESDHC will ignore this bit no matter if it is set or not. 1 Enable 0 Disable
1 BCEN	Block Count Enable: This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 1 Enable 0 Disable
0 DMAEN	DMA Enable: This bit enables DMA functionality. If this bit is set to 1, a DMA operation shall begin when the Host Driver sets the DPSEL bit of this register. Whether the Simple DMA, or the Advanced DMA, is active depends on the DMA Select field of the Protocol Control register. 1 Enable 0 Disable

26.7.5 Command Response 0 (ESDHCV2x_CMDRSP0)

This register is used to store part 0 of the response bits from the card.

Addresses: ESDHCV2-1_CMDRSP0 is 5000_4000h base + 10h offset = 5000_4010h
ESDHCV2-2_CMDRSP0 is 5000_8000h base + 10h offset = 5000_8010h
ESDHCV2-4_CMDRSP0 is 5002_4000h base + 10h offset = 5002_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP0[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2x_CMDRSP0 field descriptions

Field	Description
31–0 CMDRSP0[31:0]	Command Response 0: Refer to Command Response 3 (ESDHCV2_CMDRSP3) for the mapping of command responses from the SD Bus to this register for each response type.

26.7.6 Command Response 1 (ESDHCV2x_CMDRSP1)

This register is used to store part 1 of the response bits from the card.

Addresses: ESDHCV2-1_CMDRSP1 is 5000_4000h base + 14h offset = 5000_4014h
ESDHCV2-2_CMDRSP1 is 5000_8000h base + 14h offset = 5000_8014h
ESDHCV2-4_CMDRSP1 is 5002_4000h base + 14h offset = 5002_4014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP1[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2x_CMDRSP1 field descriptions

Field	Description
31–0 CMDRSP1[31:0]	Command Response 1: Refer to Command Response 3 (ESDHCV2_CMDRSP3) for the mapping of command responses from the SD Bus to this register for each response type.

26.7.7 Command Response 2 (ESDHCV2x_CMDRSP2)

This register is used to store part 2 of the response bits from the card.

Addresses: ESDHCV2-1_CMDRSP2 is 5000_4000h base + 18h offset = 5000_4018h
ESDHCV2-2_CMDRSP2 is 5000_8000h base + 18h offset = 5000_8018h
ESDHCV2-4_CMDRSP2 is 5002_4000h base + 18h offset = 5002_4018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP2[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2x_CMDRSP2 field descriptions

Field	Description
31–0 CMDRSP2[31:0]	Command Response 2: Refer to Command Response 3 (ESDHCV2_CMDRSP3) for the mapping of command responses from the SD Bus to this register for each response type.

26.7.8 Command Response 3 (ESDHCV2x_CMDRSP3)

This register is used to store part 3 of the response bits from the card.

Command Response 3 (ESDHCV2_CMDRSP3) describes the mapping of command responses from the SD Bus to Command Response registers for each response type. In the table, R[] refers to a bit range within the response data as transmitted on the SD Bus.

Table 26-48. Response Bit Definition for Each Response Type

Response Type	Meaning of Response	Response Field	Response Register
R1,R1b (normal response)	Card Status	R[39:8]	ESDHCV2_CMDRSP0
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	ESDHCV2_CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{ESDHCV2_CMDRSP3[23:0], ESDHCV2_CMDRSP2, ESDHCV2_CMDRSP1, ESDHCV2_CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	ESDHCV2_CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	ESDHCV2_CMDRSP0
R5, R5b	SDIO response	R[39:8]	ESDHCV2_CMDRSP0
R6 (Publish RCA)	New Published RCA[31:16] and card status[15:0]	R[39:9]	ESDHCV2_CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the ESDHCV2_CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the ESDHCV2_CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the ESDHCV2_CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, the ESDHC only stores part of the response data in the Command Response registers. This enables the Host Driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by the ESDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the ESDHC will check R[47:1], and if the response length is 136 the ESDHC will check R[119:1].

As ESDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, the ESDHC stores the Auto CMD12 response in the ESDHCV2_CMDRSP3 register. The CMD_wo_DAT response is stored in ESDHCV2_CMDRSP0. This allows the ESDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the ESDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Addresses: ESDHCV2-1_CMDRSP3 is 5000_4000h base + 1Ch offset = 5000_401Ch
ESDHCV2-2_CMDRSP3 is 5000_8000h base + 1Ch offset = 5000_801Ch
ESDHCV2-4_CMDRSP3 is 5002_4000h base + 1Ch offset = 5002_401Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP3[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2x_CMDRSP3 field descriptions

Field	Description
31–0 CMDRSP3[31:0]	Command Response 3: Refer to Command Response 3 (ESDHCV2_CMDRSP3) for the mapping of command responses from the SD Bus to this register for each response type.

26.7.9 Data Buffer Access Port (ESDHCV2x_DATPORT)

This is a 32-bit data port register used to access the internal buffer.

Addresses: ESDHCV2-1_DATPORT is 5000_4000h base + 20h offset = 5000_4020h
ESDHCV2-2_DATPORT is 5000_8000h base + 20h offset = 5000_8020h
ESDHCV2-4_DATPORT is 5002_4000h base + 20h offset = 5002_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATCONT[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2x_DATPORT field descriptions

Field	Description
31–0 DATCONT[31:0]	Data Content: The Buffer Data Port register is for 32-bit data access by the ARM platform or the external DMA. When the internal DMA is enabled, any write to this register is ignored, and any read from this register will always yield 0s.

26.7.10 Present State (ESDHCV2x_PRSSTAT)

The Host Driver can get status of the ESDHC from this 32-bit read only register. It issues CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DAT lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands will be issued when Command Inhibit (DAT) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

NOTE

The reset value of Present State Register depend on testbench connectivity.

Addresses: ESDHCV2-1_PRSSTAT is 5000_4000h base + 24h offset = 5000_4024h
ESDHCV2-2_PRSSTAT is 5000_8000h base + 24h offset = 5000_8024h
ESDHCV2-4_PRSSTAT is 5002_4000h base + 24h offset = 5002_4024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLSL[7:0]								CLSL	0			WPSPL	CDPL	0	CINS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				BREN	BWEN	RTA	WTA	SDOFF	PEROFF	HCKOFF	IPGOFF	SDSTB	DLA	CDIHB	CIHB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2x_PRSSTAT field descriptions

Field	Description
31–24 DLSL[7:0]	DAT[7:0] Line Signal Level: This status is used to check the DAT line level to recover from errors, and for debugging.This is especially useful in detecting the busy signal level from DAT[0]. The reset value is effected by the external pull-up/ pull-down resistors. By default, the read value of this bit field after reset is 8'b11110111, when DAT[3] is pulled down and the other lines are pulled up. DAT[7]: Data 7 line signal level DAT[6]: Data 6 line signal level

Table continues on the next page...

ESDHCV2x_PRSSTAT field descriptions (continued)

Field	Description
	DAT[5]: Data 5 line signal level DAT[4]: Data 4 line signal level DAT[3]: Data 3 line signal level DAT[2]: Data 2 line signal level DAT[1]: Data 1 line signal level DAT[0]: Data 0 line signal level
23 CLSL	CMD Line Signal Level: This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is effected by the external pull-up/pull-down resistor, by default, the read value of this bit after reset is 1'b1, when the command line is pulled up.
22–20 Reserved	This read-only field is reserved and always has the value zero. Reserved
19 WPSPL	Write Protect Switch Pin Level: The Write Protect Switch is supported for memory and combo cards.This bit reflects the inverted value of the SD_WP pin of the card socket. A software reset does not affect this bit. The reset value is effected by the external write protect switch. If the SD_WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled. 1 Write enabled (SD_WP=0) 0 Write protected (SD_WP=1)
18 CDPL	Card Detect Pin Level: This bit reflects the inverse value of the SD_CD# pin for the card socket. Debouncing is not performed on this bit. This bit may be valid, but is not guaranteed, because of propagation delay. Use of this bit is limited to testing As it must be debounced by software. A software reset does not effect this bit. A write to the Force Event Register does not effect this bit. The reset value is effected by the external card detection pin. This bit shows the value on the SD_CD# pin (that is, when a card is inserted in the socket, it is 0 on the SD_CD# input, and consequently the CDPL reads 1.) 1 Card present (SD_CD#=0) 0 No card present (SD_CD#=1)
17 Reserved	This read-only field is reserved and always has the value zero. Reserved
16 CINS	Card Inserted: This bit indicates whether a card has been inserted. The ESDHC debounces this signal so that the Host Driver will not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not effect this bit. The Software Reset For All in the System Control register does not effect this bit.A software reset does not effect this bit. 1 Card Inserted 0 Power on Reset or No Card
15–12 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

ESDHCV2x_PRSSTAT field descriptions (continued)

Field	Description
11 BREN	Buffer Read Enable: This status bit is used for non-DMA read transfers. The ESDHC may implement multiple buffers to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this bit is high, valid data greater than the watermark level exist in the buffer. A change of this bit from 1 to 0 occurs when any read from the buffer is made. A change of this bit from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 1 Read enable 0 Read disable
10 BWEN	Buffer Write Enable: This status bit is used for non-DMA write transfers. The ESDHC can implement multiple buffers to transfer data efficiently. This read only flag indicates if space is available for write data. If this bit is 1, valid data greater than the watermark level can be written to the buffer. A change of this bit from 1 to 0 occurs when any write to the buffer is made. A change of this bit from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled. 1 Write enable 0 Write disable
9 RTA	Read Transfer Active: This status bit is used for detecting completion of a read transfer. This bit is set for either of the following conditions: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. A Transfer Complete interrupt is generated when this bit changes to 0. This bit is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, that is, all data are read away from ESDHC internal buffer. • When all valid data blocks have been transferred from ESDHC internal buffer to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1. 1 Transferring data 0 No valid data
8 WTA	Write Transfer Active: This status bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the ESDHC. This bit is set in either of the following cases: • After the end bit of the write command. • When writing 1 to the Continue Request bit in the Protocol Control register to restart a write transfer. This bit is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple). • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request.

Table continues on the next page...

ESDHCV2x_PRSSTAT field descriptions (continued)

Field	Description
	During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host Driver in determining when to issue commands during Write Busy state. 1 Transferring data 0 No valid data
7 SDOFF	SD Clock Gated Off Internally: This status bit indicates that the SD Clock is internally gated off, because of buffer over/under-run or read pause without read wait assertion, or the driver has cleared SDCLKEN bit to stop the SD clock. This bit is for the Host Driver to debug data transaction on the SD bus. 1 SD Clock is gated off 0 SD Clock is active
6 PEROFF	ipg_perclk Gated Off Internally: This status bit indicates that the ipg_perclk is internally gated off. This bit is for the Host Driver to debug transaction on the SD bus. When INITA bit is set, ESDHC sending 80 clock cycles to the card, the SDCLKEN bit must be '1' to enable the output card clock, otherwise the ipg_perclk will never be gate off, so ipg_perclk and ipg_clk will be always active. 1 ipg_perclk is gated off 0 ipg_perclk is active
5 HCKOFF	hclk Gated Off Internally: This status bit indicates that the hclk is internally gated off. This bit is for the Host Driver to debug during a data transfer. 1 hclk is gated off 0 hclk is active
4 IPGOFF	ipg_clk Gated Off Internally: This status bit indicates that the ipg_clk is internally gated off. This bit is for the Host Driver to debug. 1 ipg_clk is gated off 0 ipg_clk is active
3 SDSTB	SD Clock Stable This status bit indicates that the internal card clock is stable. This bit is for the Host Driver to poll clock status when changing the clock frequency. It is recommended to clear SDCLKEN bit in System Control register to remove glitch on the card clock when the frequency is changing. 1 clock is stable 0 clock is changing frequency and not stable
2 DLA	Data Line Active This status bit indicates whether one of the DAT lines on the SD Bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD Bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This bit will be set in either of the following cases:

Table continues on the next page...

ESDHCV2x_PRSSTAT field descriptions (continued)

Field	Description
	• After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. This bit will be cleared in either of the following cases: (1) When the end bit of the last data block is sent from the SD Bus to the ESDHC. (2) When the Read Wait state is stopped by a Suspend command and the DAT2 line is released. The ESDHC will wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the ESDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function. This bit will remain 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD Bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the write command. • When writing to 1 to the Continue Request bit in the Protocol Control register to continue a write transfer. This bit will be cleared in either of the following cases: • When the SD card releases Write Busy of the last data block, the ESDHC will also detect if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the ESDHC will assume the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and as a result of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This bit will be cleared when the DAT0 line is released. 1 DAT Line Active 0 DAT Line Inactive
1 CDIHB	Command Inhibit (DAT): This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates that the ESDHC can issue the next SD/MMC Command. Commands with a busy signal belong to Command Inhibit (DAT) (for example, R1b, R5b type). Except in the case when the command busy is finished, changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. **NOTE:** The SD Host Driver can save registers for a suspend transaction after this bit has changed from 1 to 0. 1 Cannot issue command which uses the DAT line 0 Can issue command which uses the DAT line
0 CIHB	Command Inhibit (CMD): If this status bit is 0, it indicates that the CMD line is not in use and the ESDHC can issue a SD/MMC Command using the CMD line.

Table continues on the next page...

ESDHCV2x_PRSSTAT field descriptions (continued)

Field	Description
	This bit is set also immediately after the Transfer Type register is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DAT) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If the ESDHC cannot issue the command because of a command conflict error (Refer to Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this bit will remain 1 and the Command Complete is not set. The Status of issuing an Auto CMD12 does not show on this bit. 1 Cannot issue command 0 Can issue command using only CMD line

26.7.11 Protocol Control (ESDHCV2x_PROCTL)

There are three cases to restart the transfer after stop at the block gap. The appropriate case depends on whether ESDHC issues a Suspend command or the SD card accepts the Suspend command.

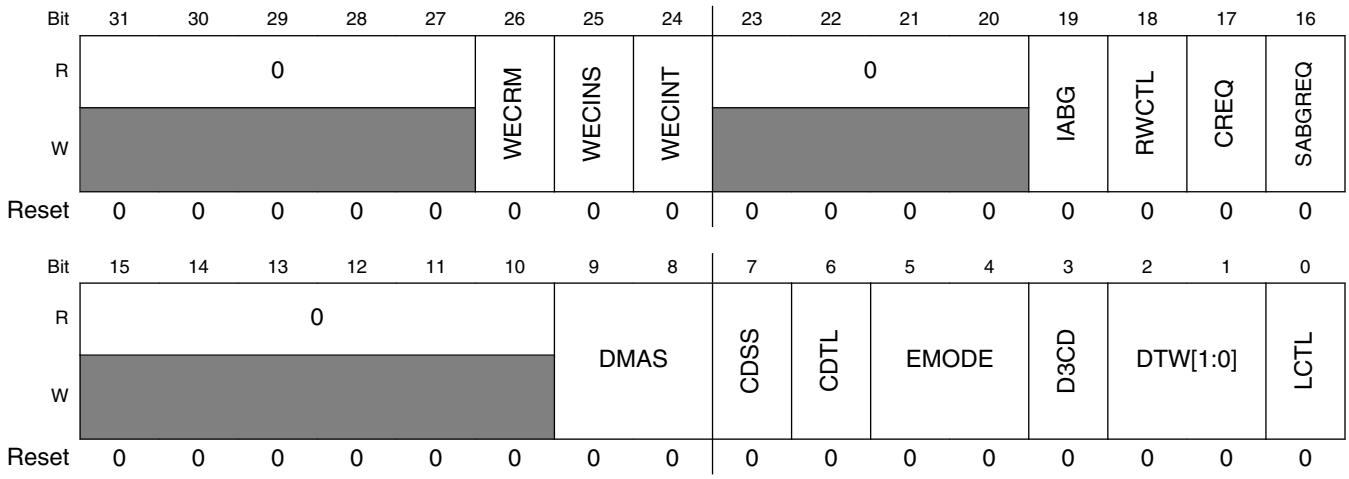
1. If the Host Driver does not issue a Suspend command, the Continue Request will be used to restart the transfer.
2. If the Host Driver issues a Suspend command and the SD card accepts it, a Resume command will be used to restart the transfer.
3. If the Host Driver issues a Suspend command and the SD card does not accept it, the Continue Request will be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the Host Driver will wait for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the Host Driver will clear the Stop At Block Gap Request before or simultaneously.

Addresses: ESDHCV2-1_PROCTL is 5000_4000h base + 28h offset = 5000_4028h

ESDHCV2-2_PROCTL is 5000_8000h base + 28h offset = 5000_8028h

ESDHCV2-4_PROCTL is 5002_4000h base + 28h offset = 5002_4028h



ESDHCV2x_PROCTL field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved
26 WECRM	Wakeup Event Enable On SD Card Removal: This bit enables a wakeup event, through a Card Removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Removal Status and the ESDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active in order to assert the Card Removal Status and the ESDHC interrupt. 1 Enable 0 Disable
25 WECINS	Wakeup Event Enable On SD Card Insertion: This bit enables a wakeup event, through a Card Insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Insertion Status and the ESDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active in order to assert the Card Insertion Status and the ESDHC interrupt. 1 Enable 0 Disable
24 WECINT	Wakeup Event Enable On Card Interrupt: This bit enables a wakeup event, through a Card Interrupt, in the Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this bit is set, the Card Interrupt Status and the ESDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active in order to assert the Card Interrupt Status and the ESDHC interrupt. 1 Enable 0 Disable
23–20 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

ESDHCV2x_PROCTL field descriptions (continued)

Field	Description
19 IABG	Interrupt At Block Gap: This bit is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0 to avoid an inadvertent interrupt. When the Host Driver detects an SDIO card insertion, it sets this bit according to the CCCR of the card. 1 Enabled 0 Disabled
18 RWCTL	Read Wait Control: The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DAT[2] line. Otherwise the ESDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SDIO card insertion, it sets this bit according to the CCCR of the card. If the card does not support read wait, this bit will never be set to 1, otherwise DAT line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the ESDHC will stop the SD Clock to pause reading operation. 1 Enable Read Wait Control, and assert Read Wait without stopping SD Clock at block gap when SABGREQ bit is set 0 Disable Read Wait Control, and stop SD Clock at block gap when SABGREQ bit is set
17 CREQ	Continue Request: This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. When a Suspend operation is not accepted by the card, it is also by setting this bit to restart the paused transfer. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit to 1 to restart the transfer. The ESDHC automatically clears this bit, therefore it is not necessary for the Host Driver to set this bit to 0. If both Stop At Block Gap Request and this bit are 1, the continue request is ignored. 1 Restart 0 No effect
16 SABGREQ	Stop At Block Gap Request: This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the Transfer Complete is set to 1, indicating a transfer completion, the Host Driver will leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The ESDHC will honor the Stop At Block Gap Request for write transfers, but for read transfers it requires that the SDIO card support Read Wait. Therefore, the Host Driver will not set this bit during read transfers unless the SDIO card supports Read Wait and has set the Read Wait Control to 1, otherwise the ESDHC will stop the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the Host Driver writes data to the Data Port register, the Host Driver will set this bit after all block data is written. If this bit is set to 1, the Host Driver will not write data to the Data Port register after a block is sent. Once this bit is set, the Host Driver will not clear this bit before the Transfer Complete bit in Interrupt Status Register is set, otherwise the ESDHCs behavior is undefined. This bit effects Read Transfer Active, Write Transfer Active, DAT Line Active and Command Inhibit (DAT) in the Present State register. 1 Stop 0 Transfer

Table continues on the next page...

ESDHCV2x_PROCTL field descriptions (continued)

Field	Description
15–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9–8 DMAS	DMA Select: This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00 No DMA or Simple DMA is selected 01 ADMA1 is selected 10 ADMA2 is selected 11 reserved
7 CDSS	Card Detect Signal Selection: This bit selects the source for the card detection. 1 Card Detection Test Level is selected (for test purpose) 0 Card Detection Level is selected (for normal purpose)
6 CDTL	Card Detect Test Level: This is bit is enabled while the Card Detection Signal Selection is set to 1 and it indicates card insertion. 1 Card Detect Test Level is 1, card inserted 0 Card Detect Test Level is 0, no card inserted
5–4 EMODE	Endian Mode: The ESDHC supports all four endian modes in data transfer. Refer to Data Buffer for more details. 00 Big Endian Mode 01 Half Word Big Endian Mode 10 Little Endian Mode 11 Reserved
3 D3CD	DAT3 as Card Detection Pin: If this bit is set, DAT3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DAT3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 may set the card into the SPI mode, which the ESDHC does not support. In case of SDIO application, if SD_CD signal is not multiplexed to ESDHC block, S/W should set D3CD bit. 1 DAT3 as Card Detection Pin 0 DAT3 does not monitor Card Insertion
2–1 DTW[1:0]	Data Transfer Width: This bit selects the data width of the SD bus for a data transfer. The Host Driver sets it to match the data width of the card. Possible Data transfer Width is 1-bit, 4-bits or 8-bits. 10 8-bit mode 01 4-bit mode 00 1-bit mode 11 Reserved
0 LCTL	LED Control:

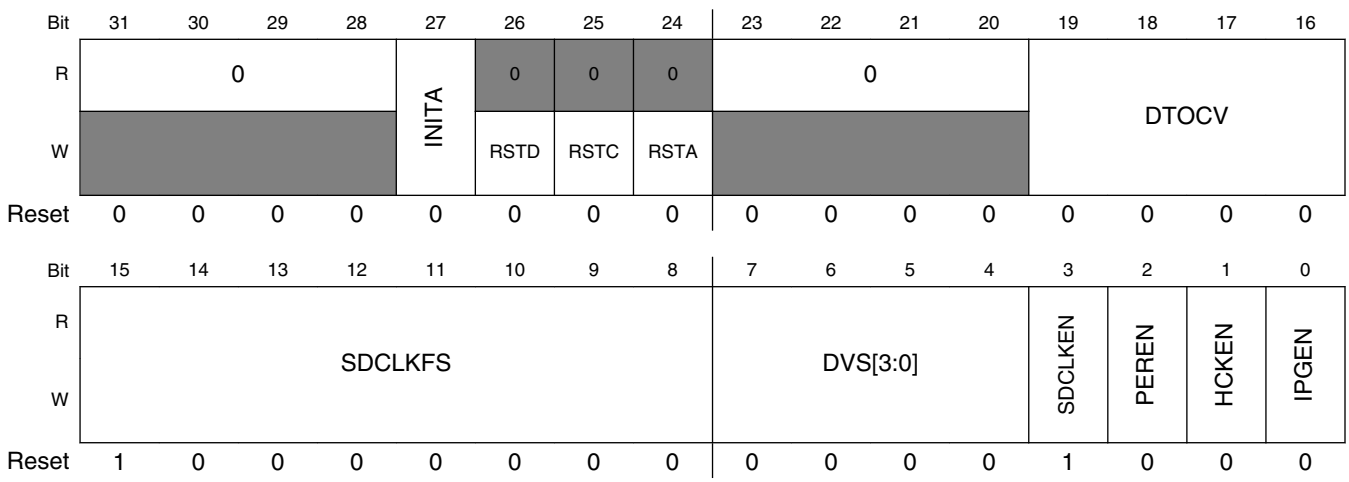
Table continues on the next page...

ESDHCV2x_PROCTL field descriptions (continued)

Field	Description
	This bit, fully controlled by the Host Driver, is used to caution the user not to remove the card while the card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction. When the software issues multiple SD commands, setting the bit once before the first command is sufficient; it is not necessary to reset the bit between commands. 1 LED on 0 LED off

26.7.12 System Control (ESDHCV2x_SYSCTL)

Addresses: ESDHCV2-1_SYSCTL is 5000_4000h base + 2Ch offset = 5000_402Ch
ESDHCV2-2_SYSCTL is 5000_8000h base + 2Ch offset = 5000_802Ch
ESDHCV2-4_SYSCTL is 5002_4000h base + 2Ch offset = 5002_402Ch



ESDHCV2x_SYSCTL field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value zero. Reserved
27 INITA	Initialization Active: When this bit is set, 80 SD-Clocks are sent to the card. After the 80 clocks are sent, this bit is self cleared. This bit is very useful during the card power-up period when 74 SD-Clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the CIHB and CDIHB bits in the Present State Register are set, writing 1 to this bit is ignored (that is, when command line or data lines are active, write to this bit is not allowed). On the other hand, when this bit is set, that is, during initialization active period, it is allowed to issue commands, and the command bit stream will appear on the CMD pad after all 80 clock cycles are completer. When this command ends, the driver can make sure that the 80 clock cycles are sent out, which is very useful when the driver needs to send 80 cycles to the card and does not want to wait until this bit is self cleared.

Table continues on the next page...

ESDHCV2x_SYSCTL field descriptions (continued)

Field	Description
26 RSTD	Software Reset For DAT Line: Only part of the data circuit is reset. DMA circuit is also reset. The following registers and bits are cleared by this bit: • Data Port register • Buffer is cleared and initialized.Present State register • Buffer Read Enable • Buffer Write Enable • Read Transfer Active • Write Transfer Active • DAT Line Active • Command Inhibit (DAT) Protocol Control register • Continue Request • Stop At Block Gap Request Interrupt Status register • Buffer Read Ready • Buffer Write Ready • DMA Interrupt • Block Gap Event • Transfer Complete 1 Reset 0 No Reset
25 RSTC	Software Reset For CMD Line: Only part of the command circuit is reset. The following registers and bits are cleared by this bit: • Present State register Command Inhibit (CMD) • Interrupt Status register Command Complete 1 Reset 0 No Reset
24 RSTA	Software Reset For ALL: This reset effects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the Host Driver will set this bit to 1 to reset the ESDHC. The ESDHC will reset this bit to 0 when the capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All does not affect the value of the Capabilities registers. After this bit is set, it is recommended that the Host Driver reset the external card and re-initialize it. b Reset 0 No Reset
23–20 Reserved	This read-only field is reserved and always has the value zero. Reserved
19–16 DTOCV	Data Timeout Counter Value: This value determines the interval by which DAT line timeouts are detected. Refer to the Data Timeout Error bit in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock SDCLK value by this value. The Host Driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events.

Table continues on the next page...

ESDHCV2x_SYSCTL field descriptions (continued)

Field	Description
	1111 Reserved 1110 SDCLK x 2^{27} 0001 SDCLK x 2^{14} 0000 SDCLK x 2^{13}
15–8 SDCLKFS	SDCLK Frequency Select: This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (this register) and divisor (next register) of the Base Clock Frequency register. Setting 00h bypasses the frequency prescaler of the SD Clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h will yield 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this bit field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD Clock after reset is 375 kHz. According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD Clock frequency is 50 MHz and will never exceed this limit. Only the following settings are allowed: 0x80 Base clock divided by 256 0x40 Base clock divided by 128 0x20 Base clock divided by 64 0x10 Base clock divided by 32 0x08 Base clock divided by 16 0x04 Base clock divided by 8 0x02 Base clock divided by 4 0x01 Base clock divided by 2 0x00 Base clock (10 MHz-63 MHz)
7–4 DVS[3:0]	Divisor: This register is used to provide a more exact divisor to generate the desired SD clock frequency. **NOTE:** The divider can even support odd divisor without deterioration of duty cycle. The setting are as following: 0x0 Divisor by 1 0x1 Divisor by 2 0xe Divisor by 15 0xf Divisor by 16

Table continues on the next page...

ESDHCV2x_SYSCTL field descriptions (continued)

Field	Description
3 SDCLKEN	SD Clock Enable The Host Controller will stop SDCLK when writing this bit to 0. SDCLK Frequency can be changed when this bit is 0. Then, the Host Controller will maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK=0). If the Card Inserted in the Present State register is cleared, this bit should be cleared by the Host Driver to save power.
2 PEREN	Peripheral Clock Enable: If this bit is set, ipg_perclk will always be active and no automatic gating is applied. Thus the SDCLK is active except when auto gating-off during buffer danger (buffer about to over-run or under-run). When this bit is cleared, the ipg_perclk will be automatically off when there is no transaction on the SD bus. As this bit is only a feature enabling bit, clearing this bit does not stop SDCLK immediately. The ipg_perclk will be internally gated off, if none of the following factors are met: • The cmd part is reset, or • Data part is reset, or • A soft reset, or • The cmd is about to send, or • Clock divisor is just updated, or • Continue request is just set, or • This bit is set, or • Card insertion is detected, or • Card removal is detected, or • Card external interrupt is detected, or • 80 clocks for initialization phase is ongoing 1 ipg_perclk will not be automatically gated off 0 ipg_perclk will be internally gated off
1 HCKEN	HCLK Enable: If this bit is set, hclk will always be active and no automatic gating is applied. When this bit is cleared, hclk will be automatically off when no data transfer is on the SD bus. 1 hclk will not be automatically gated off 0 hclk will be internally gated off
0 IPGEN	IPG Clock Enable: If this bit is set, ipg_clk will always be active and no automatic gating is applied. The ipg_clk will be internally gated off, if none of the following factors are met: • The cmd part is reset, or • Data part is reset, or • Soft reset, or • The cmd is about to send, or • Clock divisor is just updated, or • Continue request is just set, or • This bit is set, or • Card insertion is detected, or • Card removal is detected, or • Card external interrupt is detected, or • The ipg_perclk is not gated off **NOTE:** The ipg_clk will not be auto gated off if the ipg_perclk is not gated off. Clearing only this bit has no effect unless the PEREN bit is also cleared.

Table continues on the next page...

ESDHCV2x_SYSCTL field descriptions (continued)

Field	Description
	1 ipg_clk will not be automatically gated off 0 ipg_clk will be internally gated off

26.7.13 Interrupt Status (ESDHCV2x_IRQSTAT)

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. Before clearing the Card Interrupt, make sure that the card stops asserting the interrupt, that is, the Card Driver stops servicing the interrupt condition. Otherwise, the CINT bit will be asserted again.

Interrupt Status (ESDHCV2_IRQSTAT) below shows the relationship between the Command Timeout Error and the Command Complete.

Table 26-54. ESDHC Status for Command Timeout Error/Command Complete Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

Table below shows the relationship between the Transfer Complete and the Data Timeout Error.

Table 26-55. ESDHC Status for Data Timeout Error/Transfer Complete Bit Combinations

Transfer Complete	Data Timeout Error	Meaning of the Status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data Transfer Complete

Table below shows the relationship between the Command CRC Error and Command Timeout Error.

Table 26-56. ESDHC Status for Command CRC Error/Command Timeout Error Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	No error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Addresses: ESDHCV2-1_IRQSTAT is 5000_4000h base + 30h offset = 5000_4030h
ESDHCV2-2_IRQSTAT is 5000_8000h base + 30h offset = 5000_8030h
ESDHCV2-4_IRQSTAT is 5002_4000h base + 30h offset = 5002_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAE	0			AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				w1c				w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2*x*_IRQSTAT field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 DMAE	DMA Error: Occurs when an Internal DMA transfer has failed. This bit is set to 1, when some error occurs in the data transfer. This error can be caused by either Simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. As any error corrupts the whole data block, the Host Driver will re-start the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 1 Error 0 No Error
27–25 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

ESDHCV2x_IRQSTAT field descriptions (continued)

Field	Description
24 AC12E	Auto CMD12 Error: Occurs when detecting that one of the bits in the Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1, not only when the errors in Auto CMD12 occur, but also when the Auto CMD12 is not executed due to the previous command error. 1 Error 0 No Error
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 DEBE	Data End Bit Error: Occurs either when detecting 0 at the end bit position of read data, which uses the DAT line, or at the end bit position of the CRC. 1 Error 0 No Error
21 DCE	Data CRC Error: Occurs when detecting a CRC error when transferring read data, which uses the DAT line, or when detecting the Write CRC status having a value other than 010. 1 Error 0 No Error
20 DTOE	Data Timeout Error: Occurs when detecting one of following time-out conditions. • Busy time-out for R1b,R5b type • Busy time-out after Write CRC status • Read Data time-out. 1 Time out 0 No Error
19 CIE	Command Index Error: Occurs if a Command Index error occurs in the command response. 1 Error 0 No Error
18 CEBE	Command End Bit Error: Occurs when detecting that the end bit of a command response is 0. 1 End Bit Error Generated 0 No Error
17 CCE	Command CRC Error: Command CRC Error is generated in two cases. • If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this bit is set when detecting a CRC error in the command response. • The ESDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the ESDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then

Table continues on the next page...

ESDHCV2x_IRQSTAT field descriptions (continued)

Field	Description
	the ESDHC will abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error will also be set to 1 to distinguish CMD line conflict. 1 CRC Error Generated. 0 No Error
16 CTOE	Command Timeout Error: Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the ESDHC detects a CMD line conflict, in which case a Command CRC Error will also be set (as shown in Interrupt Status (ESDHCV2_IRQSTAT)), this bit will be set without waiting for 64 SDCLK cycles. Because the command will be aborted by the ESDHC. 1 Time out 0 No Error
15–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 CINT	Card Interrupt: This status bit is set when an interrupt signal is detected from the external card. In 1-bit mode, the ESDHC will detect the Card Interrupt without the SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the Host System. Writing 1 to this bit will clear it, but as the interrupt factor from the SDIO card is not clear, this bit is set again. In order to clear this bit, reset the interrupt factor from the external card and then clear this bit. When this status bit is set, and the Host Driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be cleared to deactivate the interrupt signal to the Host System. After completion of the card interrupt service, (it should reset the interrupt factors in the SDIO card and the interrupt signal should not be asserted), write 1 to clear this bit, enable the Card Interrupt Signal Enable and start sampling the interrupt signal again. 1 Generate Card Interrupt 0 No Card Interrupt
7 CRM	Card Removal: This status bit is set if the Card Inserted bit (PRSSTNT[CINS]) changes from 1 to 0. When the Host Driver writes this bit to 1 to clear this status, the status of CINS should be confirmed, because the card state may possibly change when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if no card is inserted. In order to leave it cleared, clear the Card Removal Status Enable bit IRQSTATEN[CRMSEN]in Interrupt Status Enable register. 1 Card removed 0 Card state unstable or inserted
6 CINS	Card Insertion: This status bit is set if the Card Inserted bit in the Present State register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if a card is inserted. In order to leave it cleared, clear the Card Inserted Status Enable bit in Interrupt Status Enable register.

Table continues on the next page...

ESDHCV2x_IRQSTAT field descriptions (continued)

Field	Description
	1 Card inserted 0 Card state unstable or removed
5 BRR	Buffer Read Ready: This status bit is set if the Buffer Read Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Read Enable bit in the Present State register for additional information. 1 Ready to read buffer 0 Not ready to read buffer
4 BWR	Buffer Write Ready: This status bit is set if the Buffer Write Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Write Enable bit in the Present State register for additional information. 1 Ready to write buffer: 0 Not ready to write buffer
3 DINT	DMA Interrupt: Occurs only when the internal DMA finishes the data transfer successfully. When errors occur during data transfer, this bit will not be set. Instead, the DMAE bit will be set. Either Simple DMA or ADMA finishes data transferring, this bit will be set. 1 DMA Interrupt is generated 0 No DMA Interrupt
2 BGE	Block Gap Event: If the Stop At Block Gap Request bit in the Protocol Control register is set, this bit is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1. In the case of a Read Transaction: This bit is set at the falling edge of the DAT Line Active Status (When the transaction is stopped at SD Bus timing). The Read Wait must be supported in order to use this function. In the case of Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 1 Transaction stopped at block gap 0 No block gap event
1 TC	Transfer Complete: This bit is set when a read or write transfer is completed. In the case of a Read Transaction: This bit is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request bit in the Protocol Control register (after valid data has been read to the Host System). In the case of a Write Transaction: This bit is set at the falling edge of the DAT Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request bit in the Protocol Control register, and the data transfers are completed. (After valid data is written to the SD card and the busy signal released.)

Table continues on the next page...

ESDHCV2x_IRQSTAT field descriptions (continued)

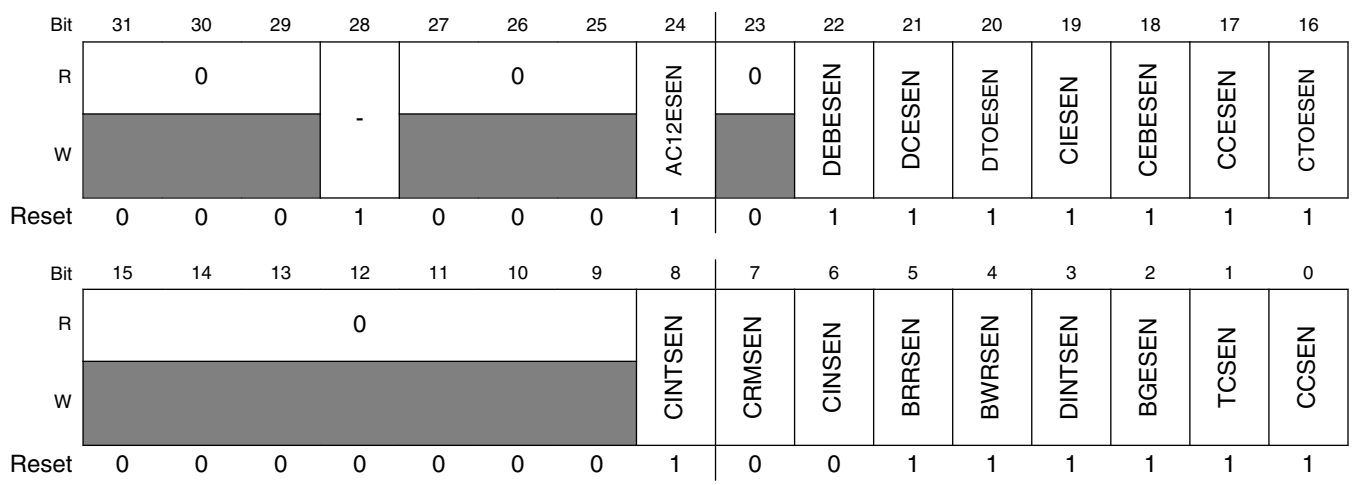
Field	Description
	1 Transfer complete 0 Transfer not complete
0 CC	Command Complete: This bit is set when you receive the end bit of the command response (except Auto CMD12). Refer to the Command Inhibit (CMD) in the Present State register. 1 Command complete 0 Command not complete

26.7.14 Interrupt Status Enable (ESDHCV2x_IRQSTATEN)

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any bit is cleared, the corresponding Interrupt Status bit is also cleared (that is, when the bit in this register is cleared, the corresponding bit in Interrupt Status Register is always 0).

- Depending on IABG bit setting, ESDHC may be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There will be some delays on the Card Interrupt, asserted from the card, to the time the Host System is informed.
- To detect a CMD line conflict, the Host Driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

Addresses: ESDHCV2-1_IRQSTATEN is 5000_4000h base + 34h offset = 5000_4034h
ESDHCV2-2_IRQSTATEN is 5000_8000h base + 34h offset = 5000_8034h
ESDHCV2-4_IRQSTATEN is 5002_4000h base + 34h offset = 5002_4034h



ESDHCV2x_IRQSTATEN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 -	DMA Error Status Enable: 1 Enabled 0 Masked
27–25 Reserved	This read-only field is reserved and always has the value zero. Reserved
24 AC12ESEN	Auto CMD12 Error Status Enable: 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 DEBESEN	Data End Bit Error Status Enable: 1 Enabled 0 Masked
21 DCESEN	Data CRC Error Status Enable: 1 Enabled 0 Masked
20 DTOESEN	Data Timeout Error Status Enable: 1 Enabled 0 Masked
19 CIESEN	Command Index Error Status Enable: 1 Enabled 0 Masked
18 CEBESEN	Command End Bit Error Status Enable: 1 Enabled 0 Masked
17 CCESEN	Command CRC Error Status Enable: 1 Enabled 0 Masked
16 CTOESEN	Command Timeout Error Status Enable: 1 Enabled 0 Masked
15–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 CINTSEN	Card Interrupt Status Enable:

Table continues on the next page...

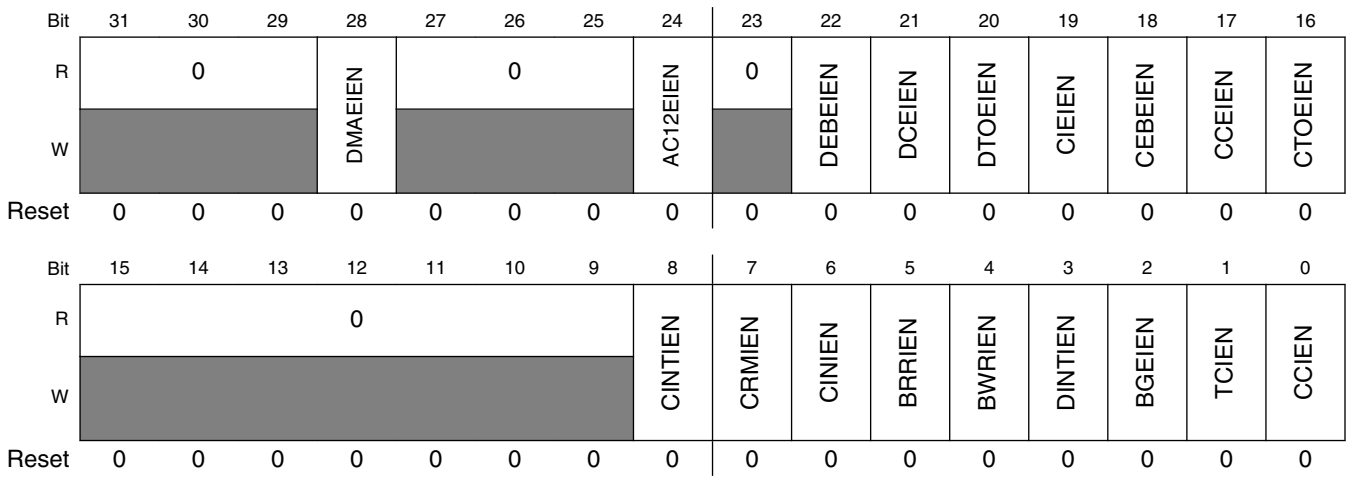
ESDHCV2x_IRQSTATEN field descriptions (continued)

Field	Description
	If this bit is set to 0, the ESDHC will clear the interrupt request to the System. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 1 Enabled 0 Masked
7 CRMSEN	Card Removal Status Enable: 1 Enabled 0 Masked
6 CINSEN	Card Insertion Status Enable: 1 Enabled 0 Masked
5 BRRSEN	Buffer Read Ready Status Enable: 1 Enabled 0 Masked
4 BWRSEN	Buffer Write Ready Status Enable: 1 Enabled 0 Masked
3 DINTSEN	DMA Interrupt Status Enable: 1 Enabled 0 Masked
2 BGESEN	Block Gap Event Status Enable: 1 Enabled 0 Masked
1 TCSEN	Transfer Complete Status Enable: 1 Enabled 0 Masked
0 CCSEN	Command Complete Status Enable: 1 Enabled 0 Masked

26.7.15 Interrupt Signal Enable (ESDHCV2x_IRQSIGEN)

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding Status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

Addresses: ESDHCV2-1_IRQSIGEN is 5000_4000h base + 38h offset = 5000_4038h
ESDHCV2-2_IRQSIGEN is 5000_8000h base + 38h offset = 5000_8038h
ESDHCV2-4_IRQSIGEN is 5002_4000h base + 38h offset = 5002_4038h



ESDHCV2x_IRQSIGEN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 DMAEIEN	DMA Error Interrupt Enable: 1 Enable 0 Masked
27–25 Reserved	This read-only field is reserved and always has the value zero. Reserved
24 AC12EIEN	Auto CMD12 Error Interrupt Enable: 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 DEBEIEN	Data End Bit Error Interrupt Enable: 1 Enabled 0 Masked

Table continues on the next page...

ESDHCV2x_IRQSIGEN field descriptions (continued)

Field	Description
21 DCEIEN	Data CRC Error Interrupt Enable: 1 Enabled 0 Masked
20 DTOEIEN	Data Timeout Error Interrupt Enable: 1 Enabled 0 Masked
19 CIEIEN	Command Index Error Interrupt Enable: 1 Enabled 0 Masked
18 CEBEIEN	Command End Bit Error Interrupt Enable: 1 Enabled 0 Masked
17 CCEIEN	Command CRC Error Interrupt Enable: 1 Enabled 0 Masked
16 CTOEIEN	Command Timeout Error Interrupt Enable 1 Enabled 0 Masked
15–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 CINTIEN	Card Interrupt Interrupt Enable: 1 Enabled 0 Masked
7 CRMIEN	Card Removal Interrupt Enable: 1 Enabled 0 Masked
6 CINIEN	Card Insertion Interrupt Enable: 1 Enabled 0 Masked
5 BRRIEN	Buffer Read Ready Interrupt Enable: 1 Enabled 0 Masked
4 BWRIEN	Buffer Write Ready Interrupt Enable: 1 Enabled 0 Masked

Table continues on the next page...

ESDHCV2x_IRQSIGEN field descriptions (continued)

Field	Description
3 DINTIEN	DMA Interrupt Enable: 1 Enabled 0 Masked
2 BGEIEN	Block Gap Event Interrupt Enable: 1 Enabled 0 Masked
1 TCIEN	Transfer Complete Interrupt Enable: 1 Enabled 0 Masked
0 CCIEN	Command Complete Interrupt Enable: 1 Enabled 0 Masked

26.7.16 Auto CMD12 Status (ESDHCV2x_AUTOC12ERR)

When the Auto CMD12 Error Status bit in the Status register is set, the Host Driver will check this register to identify what kind of error the Auto CMD12 indicated. This register is valid only when the Auto CMD12 Error status bit is set.

Table below shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 26-60. Relationship Between Command CRC Error and Command Timeout Error for Auto CMD12

Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Type of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

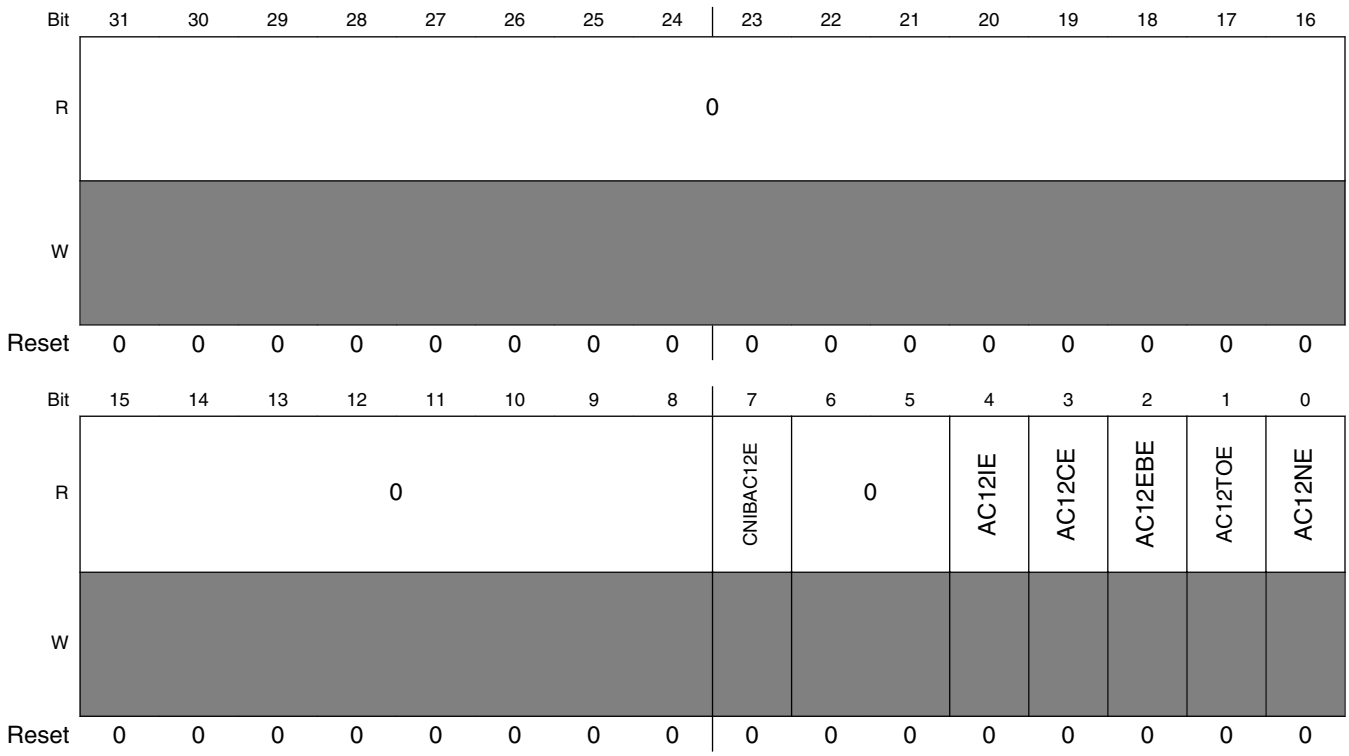
1. When the ESDHC is going to issue an Auto CMD12.
2. Set bit 0 to 1 if the Auto CMD12 cannot be issued due to an error in the previous command
3. Set bit 0 to 0 if the Auto CMD12 is issued
4. At the end bit of an Auto CMD12 response:
    - Check errors correspond to bits 1-4.

- Set bits 1-4 corresponding to detected errors.
- Clear bits 1-4 corresponding to detected errors

5. Before reading the Auto CMD12 Error Status bit 7:
   - Set bit 7 to 1 if there is a command that cannot be issued
   - Clear bit 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, bit 7 will be sampled when the driver is not writing to the Command register. It is suggested to read this register only when the AC12E bit in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error bits (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Addresses: ESDHCV2-1_AUTOC12ERR is 5000_4000h base + 3Ch offset = 5000_403Ch
ESDHCV2-2_AUTOC12ERR is 5000_8000h base + 3Ch offset = 5000_803Ch
ESDHCV2-4_AUTOC12ERR is 5002_4000h base + 3Ch offset = 5002_403Ch



ESDHCV2x_AUTOC12ERR field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 CNIBAC12E	Command Not Issued By Auto CMD12 Error: Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register.

Table continues on the next page...

ESDHCV2x_AUTOC12ERR field descriptions (continued)

Field	Description
	1 Not Issued 0 No error
6–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4 AC12IE	Auto CMD12 Index Error: Occurs if the Command Index error occurs in response to a command. 1 Error, the CMD index in response is not CMD12 0 No error
3 AC12CE	Auto CMD12 CRC Error: Occurs when detecting a CRC error in the command response. 1 CRC Error Met in Auto CMD12 Response 0 No CRC error
2 AC12EBE	Auto CMD12 End Bit Error: Occurs when detecting that the end bit of command response is 0 which should be 1. 1 End Bit Error Generated 0 No error
1 AC12TOE	Auto CMD12 Timeout Error: Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to1, the other error status bits (2-4) have no meaning. 1 Time out 0 No error
0 AC12NE	Auto CMD12 Not Executed: If memory multiple block data transfer is not started, due to a command error, this bit is not set because it is not necessary to issue an Auto CMD12. Setting this bit to 1 means the ESDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (1-4) have no meaning. 1 Not executed 0 Executed

26.7.17 Host Controller Capabilities (ESDHCV2x_HOSTCAPBLT)

This register provides the Host Driver with information specific to the ESDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset. Any write to this register is ignored.

Addresses: ESDHCV2-1_HOSTCAPBLT is 5000_4000h base + 40h offset = 5000_4040h
ESDHCV2-2_HOSTCAPBLT is 5000_8000h base + 40h offset = 5000_8040h
ESDHCV2-4_HOSTCAPBLT is 5002_4000h base + 40h offset = 5002_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0					VS18	VS30	VS33	SRS	DMAS	HSS	ADMAS	0	MBL[2:0]		
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2x_HOSTCAPBLT field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved
26 VS18	Voltage Support 1.8V: This bit depends on the Host System ability. 1 1.8V supported 0 1.8V not supported
25 VS30	Voltage Support 3.0V: This bit depends on the Host System ability. 1 3.0V supported 0 3.0V not supported
24 VS33	Voltage Support 3.3V: This bit depends on the Host System ability. 1 3.3V supported 0 3.3V not supported

Table continues on the next page...

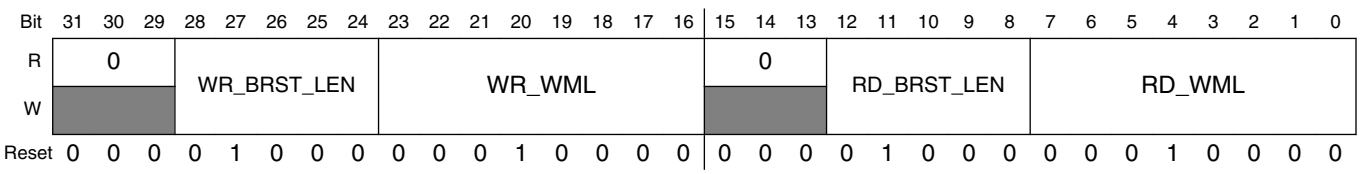
ESDHCV2x_HOSTCAPBLT field descriptions (continued)

Field	Description
23 SRS	Suspend / Resume Support: This bit indicates whether the ESDHC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism, as well as the Read Wait, are not supported, and the Host Driver will not issue either Suspend or Resume commands. 1 Supported 0 Not supported
22 DMAS	DMA Support: This bit indicates whether the ESDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly. 1 DMA Supported 0 DMA not supported
21 HSS	High Speed Support: This bit indicates whether the ESDHC supports High Speed mode and the Host System can supply a SD Clock frequency from 25 MHz to 50 MHz. 1 High Speed Supported 0 High Speed Not Supported
20 ADMAS	ADMA Support: This bit indicates whether the ESDHC supports the ADMA feature. 1 Advanced DMA Supported 0 Advanced DMA Not supported
19 Reserved	This read-only field is reserved and always has the value zero. Reserved
18–16 MBL[2:0]	Max Block Length: This value indicates the maximum block size that the Host Driver can read and write to the buffer in the ESDHC. The buffer will transfer block size without wait cycles. 000 512 bytes 001 1024 bytes 010 2048 bytes 011 4096 bytes
15–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

26.7.18 Watermark Level (ESDHCV2*x*_WML)

Both write and read watermark levels (FIFO threshold) are configurable. There value can range from 1 to 128 words. Both write and read burst lengths are also configurable. There value can range from 1 to 31 words.

Addresses: ESDHCV2-1_WML is 5000_4000h base + 44h offset = 5000_4044h
ESDHCV2-2_WML is 5000_8000h base + 44h offset = 5000_8044h
ESDHCV2-4_WML is 5002_4000h base + 44h offset = 5002_4044h



ESDHCV2*x*_WML field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28–24 WR_BRST_LEN	Write Burst Length:[1] The number of words the ESDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (that is, it is not able to clear this field).
23–16 WR_WML	Write Watermark Level: The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15–13 Reserved	This read-only field is reserved and always has the value zero. Reserved
12–8 RD_BRST_LEN	Read Burst Length:[2] The number of words the ESDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (that is, it is not able to clear this field.
7–0 RD_WML	Read Watermark Level: The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

1. Due to system restriction, the actual burst length may not exceed 16.
2. Due to system restriction, the actual burst length may not exceed 16.

26.7.19 Force Event (ESDHCV2x_FEVT)

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status Register can be written if the corresponding bit of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register sets the corresponding bit of Interrupt Status Register. A read from this register always results in 0's. In order to change the corresponding status bits in the Interrupt Status Register, set IPGEN bit in System Control Register so that ipg_clk is always active.

Forcing a card interrupt will generate a short pulse on the DAT[1] line, and the driver will treat this interrupt as a normal interrupt. The interrupt service routine will skip polling the card interrupt factor as the interrupt is self cleared.

Addresses: ESDHCV2-1_FEVT is 5000_4000h base + 50h offset = 5000_4050h
ESDHCV2-2_FEVT is 5000_8000h base + 50h offset = 5000_8050h
ESDHCV2-4_FEVT is 5002_4000h base + 50h offset = 5002_4050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0		0	0			0	0	0	0	0	0	0	0	0
W	FEVTCINT			FEVTDMAE				FEVTAC12E		FEVTDEBE	FEVTDCE	FEVTDTOE	FEVTCIE	FEVTCEBE	FEVTCCE	FEVTCCE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0	0		0	0	0	0	0
W									FEVTCNIBAC12E			FEVTAC12IE	FEVTAC12EBE	FEVTAC12CE	FEVTAC12TOE	FEVTAC12NE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV2x_FEVT field descriptions

Field	Description
31 FEVTCINT	Force Event Card Interrupt: Writing 1 to this bit generates a short low-level pulse on the internal DAT[1] line, as if a self clearing interrupt was received from the external card. If enabled, the CINT bit will be set and the interrupt service routine will treat this interrupt as a normal interrupt from the external card.
30–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 FEVTDMAE	Force Event DMA Error: Forces the DMAE bit of Interrupt Status Register to be set
27–25 Reserved	This read-only field is reserved and always has the value zero. Reserved
24 FEVTAC12E	Force Event Auto Command 12 Error: Forces the AC12E bit of Interrupt Status Register to be set
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 FEVTDEBE	Force Event Data End Bit Error: Forces the DEBE bit of Interrupt Status Register to be set
21 FEVTDCE	Force Event Data CRC Error: Forces the DCE bit of Interrupt Status Register to be set
20 FEVTDTOE	Force Event Data Time Out Error: Force the DTOE bit of Interrupt Status Register to be set
19 FEVTCIE	Force Event Command Index Error: Forces the CCE bit of Interrupt Status Register to be set
18 FEVTCEBE	Force Event Command End Bit Error: Forces the CEBE bit of Interrupt Status Register to be set
17 FEVTCCE	Force Event Command CRC Error: Forces the CCE bit of Interrupt Status Register to be set
16 FEVTCCE	Force Event Command Time Out Error: Forces the CTOE bit of Interrupt Status Register to be set
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 FEVTCNIBAC12E	Force Event Command Not Executed By Auto Command 12 Error: Forces the CNIBAC12E bit in the Auto Command12 Error Status Register to be set
6–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4 FEVTAC12IE	Force Event Auto Command 12 Index Error: Forces the AC12IE bit in the Auto Command12 Error Status Register to be set
3 FEVTAC12EBE	Force Event Auto Command 12 End Bit Error: Forces the AC12EBE bit in the Auto Command12 Error Status Register to be set

Table continues on the next page...

ESDHCV2x_FEVT field descriptions (continued)

Field	Description
2 FEVTAC12CE	Force Event Auto Command 12 CRC Error: Forces the AC12CE bit in the Auto Command12 Error Status Register to be set
1 FEVTAC12TOE	Force Event Auto Command 12 Time Out Error: Forces the AC12TOE bit in the Auto Command12 Error Status Register to be set
0 FEVTAC12NE	Force Event Auto Command 12 Not Executed: Forces the AC12NE bit in the Auto Command12 Error Status Register to be set

26.7.20 ADMA Error Status Register (ESDHCV2x_ADMAES)

When an ADMA Error Interrupt occurrs, the ADMA Error States field (ADMAES) in this register holds the ADMA state and the ADMA System Address register (ADSADDR) holds the address of the error descriptor.

To recover from this error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address
- ST_CADR: This state is never set because it only increments the descriptor pointer and doesn't generate an ADMA error
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address

In case of a write operation, the Host Driver should use ACMD22 command (see field descriptions table) to get the number of the written block, because unwritten data may exist in the Host Controller.

The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The Host Driver can distinguish this error by reading the Valid bit of the error descriptor.

Table 26-65. ADMA Error State Coding

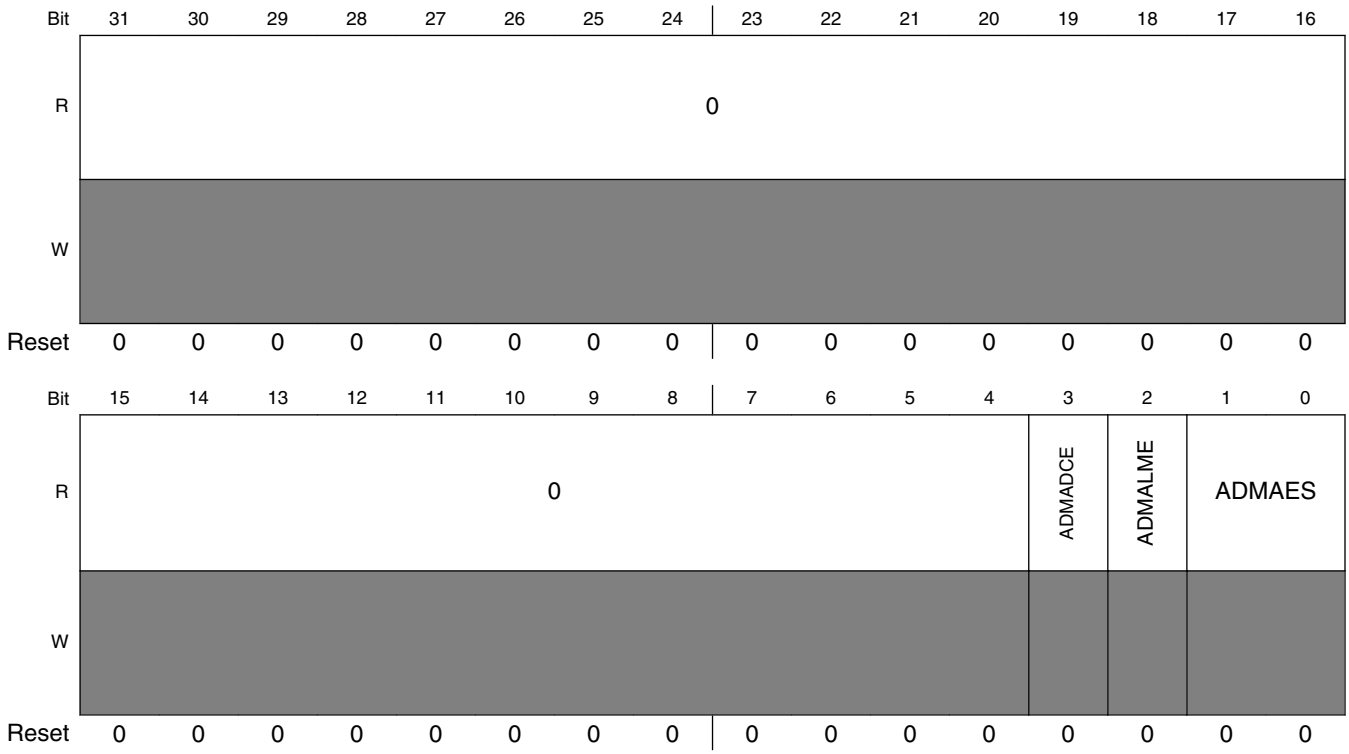
D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
00	ST_STOP (Stop DMA)	Holds the address of the next executable Descriptor command
01	ST_FDS (Fetch Descriptor)	Holds the valid Descriptor address

Table continues on the next page...

Table 26-65. ADMA Error State Coding (continued)

D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
10	ST_CADR (Change Address)	No ADMA Error is generated
11	ST_TFR (Transfer Data)	Holds the address of the next executable Descriptor command

Addresses: ESDHCV2-1_ADMAES is 5000_4000h base + 54h offset = 5000_4054h
ESDHCV2-2_ADMAES is 5000_8000h base + 54h offset = 5000_8054h
ESDHCV2-4_ADMAES is 5002_4000h base + 54h offset = 5002_4054h



ESDHCV2*x*_ADMAES field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value zero. Reserved
3 ADMADCE	ADMA Descritor Error: This error occurs when invalid descriptor fetched by ADMA: 1 Error 0 No Error
2 ADMALME	ADMA Length Mismatch Error: This error occurs in the following 2 cases:

Table continues on the next page...

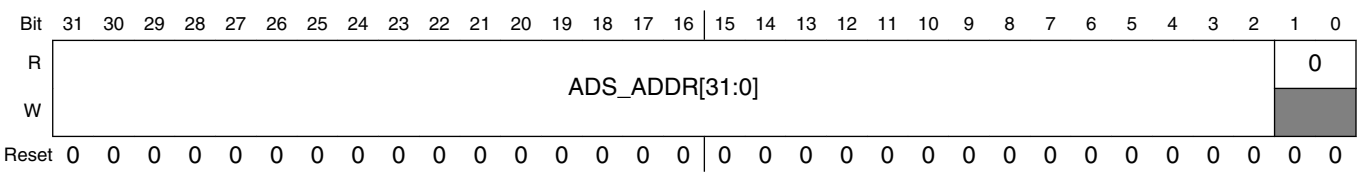
ESDHCV2x_ADMAES field descriptions (continued)

Field	Description
	• While the Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length • Total data length can not be divided by the block length 1 Error 0 No Error
1–0 ADMAES	ADMA Error State (when ADMA Error is occurred.): This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. Refer to ADMA Error Status Register (ESDHCV2_ADMAES) for more details.

26.7.21 ADMA System Address (ESDHCV2x_ADSADDR)

This register contains the physical system memory address used for ADMA transfers.

Addresses: ESDHCV2-1_ADSADDR is 5000_4000h base + 58h offset = 5000_4058h
ESDHCV2-2_ADSADDR is 5000_8000h base + 58h offset = 5000_8058h
ESDHCV2-4_ADSADDR is 5002_4000h base + 58h offset = 5002_4058h



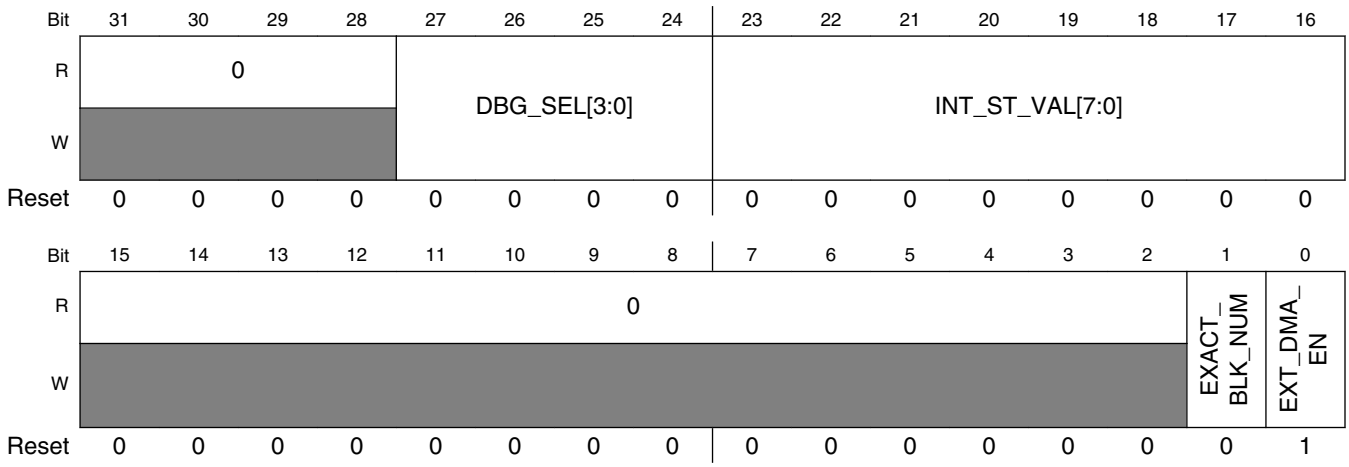
ESDHCV2x_ADSADDR field descriptions

Field	Description
31–2 ADS_ ADDR[31:0]	ADMA System Address: This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the Host Driver will set the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register will hold the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned. As this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so this register value cannot be changed when the TC bit is set. Such restriction is also listed in Software Restrictions.
1–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

26.7.22 Vendor Specific Register (ESDHCV2x_VENDOR)

This register contains the vendor specific control/status register.

Addresses: ESDHCV2-1_VENDOR is 5000_4000h base + C0h offset = 5000_40C0h
ESDHCV2-2_VENDOR is 5000_8000h base + C0h offset = 5000_80C0h
ESDHCV2-4_VENDOR is 5002_4000h base + C0h offset = 5002_40C0h



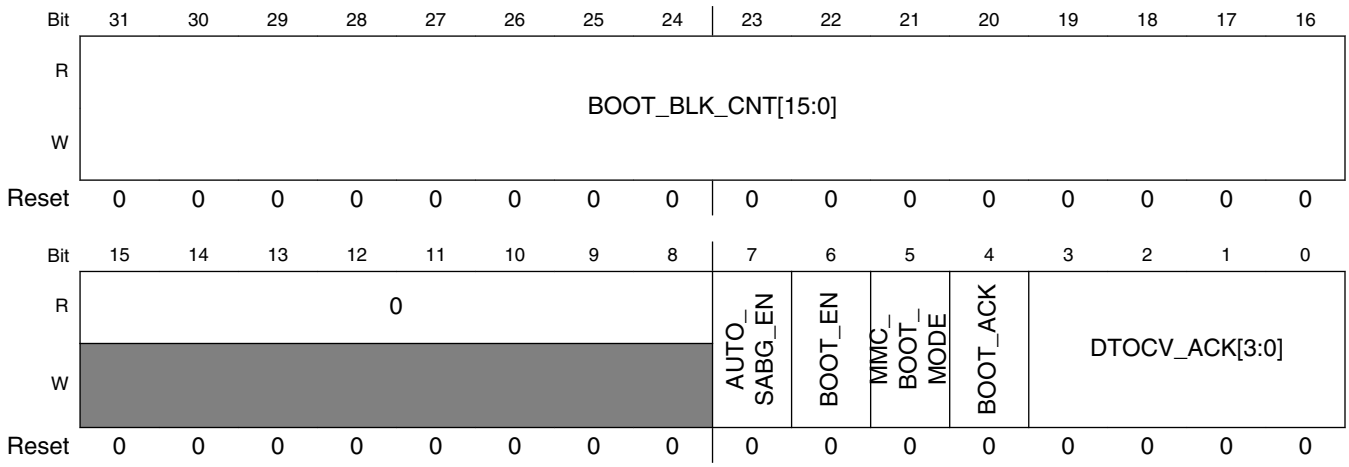
ESDHCV2x_VENDOR field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value zero. Reserved
27–24 DBG_SEL[3:0]	Debug Select Select the internal sub-block to show its internal state value.
23–16 INT_ST_VAL[7:0]	Internal State Value Internal state value, reflecting the corresponding state value selected by Debug Select field. This field is read-only and write to this field does not have effect.
15–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 EXACT_BLK_NUM	Exact block number block read enable for SDIO CMD53 This bit should be set before S/W issues CMD53 multi-block read with exact block number. This bit should not be set if the CMD53 multi-block read is not exact block number. 0 none exact block read 1 exact block read for SDIO CMD53
0 EXT_DMA_EN	External DMA Request Enable Enable the request to external DMA. When the internal DMA (either Simple DMA or Advanced DMA) is not in use, and this bit is set, ESDHC will send out DMA request when the internal buffer is ready. This bit is particularly useful when transferring data by ARM platform polling mode, and it is not allowed to send out the external DMA request. By default, this bit is set. 0 In any scenario, ESDHC does not send out external DMA request 1 When internal DMA is not active, the external DMA request will be sent out

26.7.23 MMC Boot Register (ESDHCV2x_MMCBOOT)

This register contains the MMC Fast Boot control register.

Addresses: ESDHCV2-1_MMCBOOT is 5000_4000h base + C4h offset = 5000_40C4h
ESDHCV2-2_MMCBOOT is 5000_8000h base + C4h offset = 5000_80C4h
ESDHCV2-4_MMCBOOT is 5002_4000h base + C4h offset = 5002_40C4h



ESDHCV2x_MMCBOOT field descriptions

Field	Description
31–16 BOOT_BLK_CNT[15:0]	The value defines the Stop At Block Gap value of automatic mode. When received card block cnt is equal to BOOT_BLK_CNT and AUTO_SABG_EN is 1, then Stop At Block Gap.
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 AUTO_SABG_EN	When boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when received card block cnt is equal to BOOT_BLK_CNT.
6 BOOT_EN	Boot mode enable 0: fast boot disable 1: fast boot enable
5 MMC_BOOT_MODE	Boot mode select 0: normal boot 1: alternative boot
4 BOOT_ACK	Boot ack mode select 0: no ack 1: ack
3–0 DTOCV_ACK[3:0]	Boot ACK time out counter value. 0000 SDCLK x 2^8 0001 SDCLK x 2^9 0010 SDCLK x 2^10 0011 SDCLK x 2^11 0100 SDCLK x 2^12 0101 SDCLK x 2^13

Table continues on the next page...

ESDHCV2x_MMCBOOT field descriptions (continued)

Field	Description
	0110 SDCLK x 2^14 0111 SDCLK x 2^15 1110 SDCLK x 2^22 1111 Reserved

26.7.24 Host Controller Version (ESDHCV2x_HOSTVER)

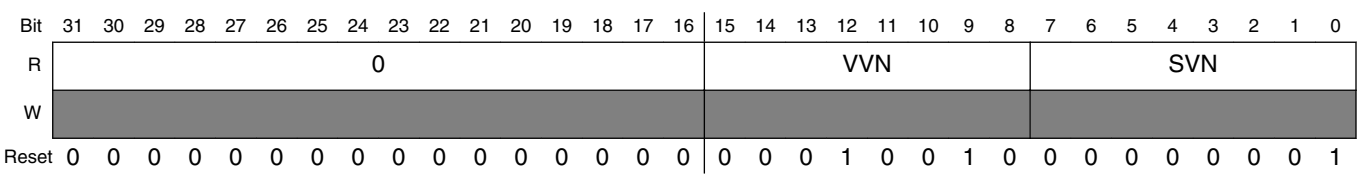
This register contains the vendor Host Controller version information. All bits are read only and when read return the reset value.

NOTE

The VVN field is 0x12 for both ESDHCV2 and ESDHCV3. Todetermine whether the controller is ESDHCV2 or ESDHCV3 do the following:

Set DLLCTRL [10]. This bit only exists in ESDHCV3. If it is set to 1 successfully, then the controller is ESDHCV3, otherwise it is ESDHCV2.

Addresses: ESDHCV2-1_HOSTVER is 5000_4000h base + FCh offset = 5000_40FCh
ESDHCV2-2_HOSTVER is 5000_8000h base + FCh offset = 5000_80FCh
ESDHCV2-4_HOSTVER is 5002_4000h base + FCh offset = 5002_40FCh



ESDHCV2x_HOSTVER field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–8 VVN	Vendor Version Number: These status bits are reserved for the vendor version number. The Host Driver will not use this status. Settings not shown are reserved. 00 Freescale ESDHC Version 1.0 10 Freescale ESDHC Version 2.0 11 Freescale ESDHC Version 2.1 12 Freescale ESDHC Version 2.2
7–0 SVN	Specification Version Number: These status bits indicate the Host Controller Specification Version.

Table continues on the next page...

ESDHCV2x_HOSTVER field descriptions (continued)

Field	Description
	Settings not shown are reserved. — 01 SD Host Specification Version 2.0, supports Test Event Register and ADMA.

Chapter 27
Enhanced Secured Digital Host Controller (eSDHCv3)

27.1 Overview

The Enhanced Secured Digital Host Controller Version 3 (ESDHCV3, referred to as ESDHC hereafter) provides the interface between the host system and the SD/SDIO/MMC cards, as depicted in Figure 27-1. The ESDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards. It handles the SD/SDIO/MMC protocols at the transmission level.

The types of cards supported by the ESDHC are described briefly as follows:

The Multi Media Card (MMC) is a universal low cost data storage and communication media that is designed to cover a wide area of applications including mobile video and gaming. Old MMC cards are based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low voltage range.

The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment and data transfer protocol are forward compatible with the old MMC (with some additions).

Under the SD protocol, it can be categorized into Memory card, I/O card and Combo card, which has both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, Figure 27-1 does not show cards with reduced size or mini cards.

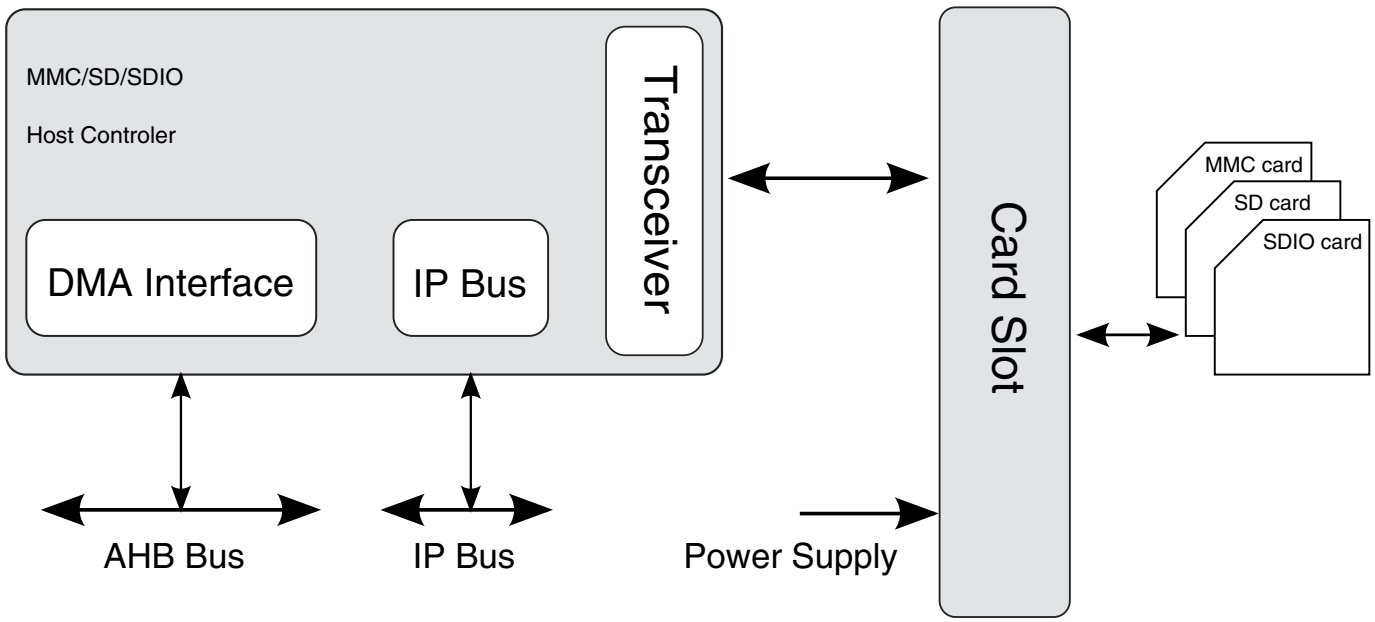


Figure 27-1. System Connection of the ESDHC

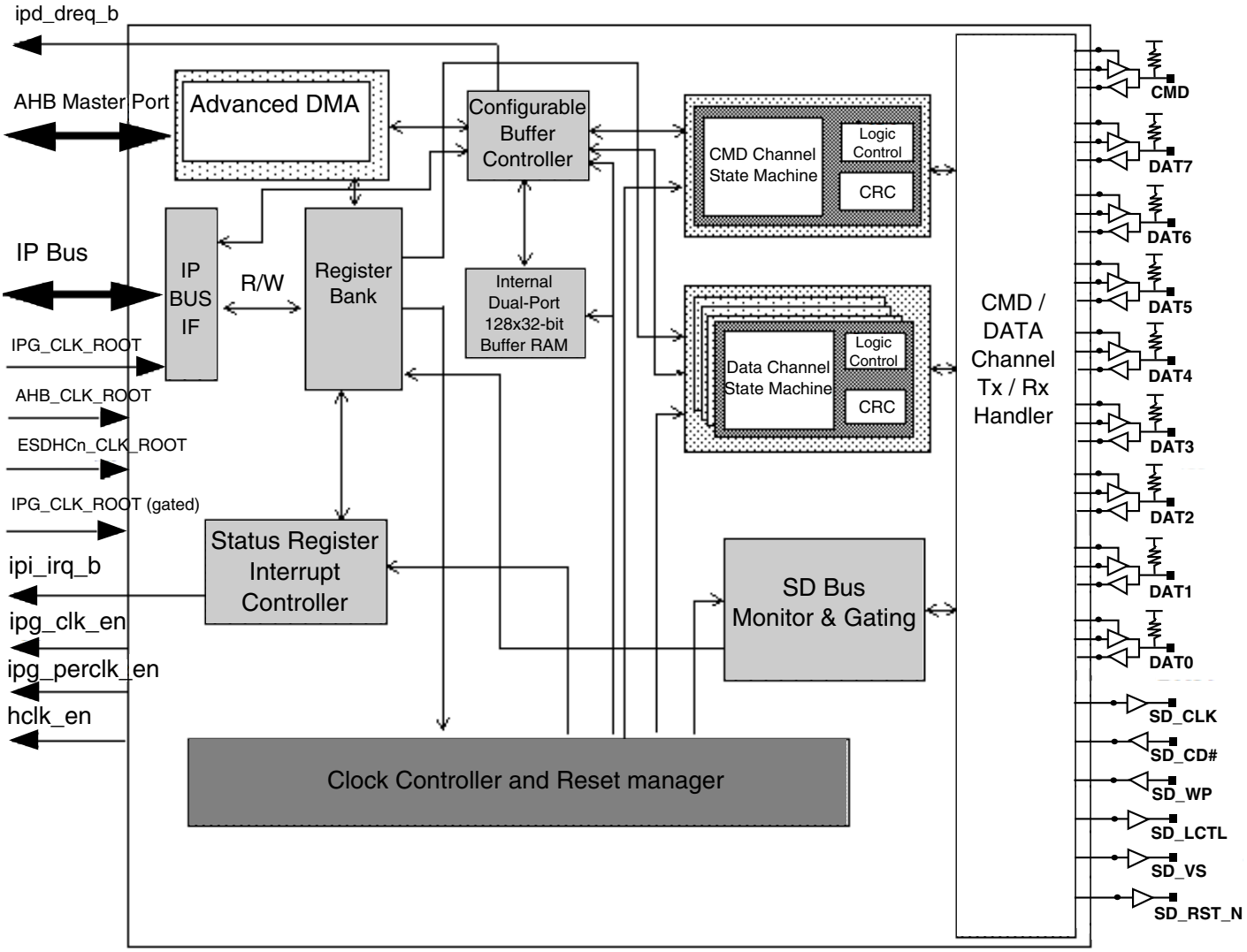


Figure 27-2. enhanced Secure Digital Host Controller Block Diagram

27.1.1 Features

The features of the ESDHC include the following:

- Conforms to the SD Host Controller Standard Specification version 2.0 including Test Event register support
- Compatible with the MMC System Specification version 4.2/4.3/4.4
- Compatible with the SD Memory Card Specification version 2.0 and supports the High Capacity SD Memory Card
- Compatible with the SDIO Card Specification version 2.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards

- Card bus clock frequency up to 52 MHz
- Supports 1-bit / 4-bit SD and SDIO modes, 1-bit / 4-bit / 8-bit MMC modes devices
  - Up to 200 Mbps of data transfer for SD/SDIO cards using 4 parallel data lines
  - Up to 416 Mbps of data transfer for MMC cards using 8 parallel data lines in SDR (Single Data Rate) mode
  - Up to 832 Mbps of data transfer for MMC cards using 8 parallel data lines in DDR (Dual Data Rate) mode
- Supports Single Block, Multi Block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports the write protection switch for write operations
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal and external DMA capabilities
- Supports Advanced DMA to perform linked memory access

27.1.2 Modes and Operations

27.1.2.1 Data transfer Modes

The ESDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- Identification Mode (up to 400 kHz)
- MMC full speed mode (up to 20 MHz)
- MMC high speed mode (up to 52 MHz)
- MMC DDR mode (up to 52MHz, sampled on both clock edges)
- SD/SDIO full speed mode (up to 25 MHz)
- SD/SDIO high speed mode (up to 50 MHz)

27.2 External Signals

27.2.1 Signals Overview

The ESDHC has 15 associate I/O signals.

- The SD_CLK is an internally generated clock used to drive the MMC, SD, SDIO cards.
- The CMD I/O is used to send commands and receive responses to/from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the ESDHC and the card.
- The SD_CD# and SD_WP are card detection and write protection signals directly routed from the socket. A low on SD_CD# means that a card is inserted and a high on SD_WP means that the write protect switch is active.
- SD_OD is an output signal generated in SoC level outside ESDHC and is used to select the external open drain resistor.
- SD_LCTL is an output signal used to drive an external LED to indicate that the SD interface is busy.
- SD_RST_N is an output signal used to reset MMC card. This should be supported by card.

SD_CD#, SD_WP, SD_OD, SD_LCTL and SD_RST_N are all optional for system implementation. If the ESDHC is desired to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high.

27.2.2 Ports Table

See Table 26-1 for the signal properties of the I/Os.

Table 27-1. Properties of I/O Signals

Name	Port	Function	Reset State	Pull up
SD_CLK	O	Clock for MMC/SD/SDIO card	0	N/A
SD_CMD	I/O	CMD line connect to card	1	Pull up
SD_DAT7	I/O	DAT7 line in 8-bit mode Not used in other modes	1	Pull up
SD_DAT6	I/O	DAT6 line in 8-bit mode Not used in other modes	1	Pull up

Table continues on the next page...

Table 27-1. Properties of I/O Signals (continued)

Name	Port	Function	Reset State	Pull up
SD_DAT5	I/O	DAT5 line in 8-bit mode Not used in other modes	1	Pull up
SD_DAT4	I/O	DAT4 line in 8-bit mode Not used in other modes	1	Pull up
SD_DAT3	I/O	DAT3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	0	Should be pull-up/pull-down configurable as this port may be used for card detection
SD_DAT2	I/O	DAT2 line or Read Wait in 4-bit mode Read Wait in 1-bit mode	1	Pull up
SD_DAT1	I/O	DAT1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	1	Pull up
SD_DAT0	I/O	DAT0 line in all modes Also used to detect busy state	1	Pull up
SD_CD#	I	Card detection pin If not used tie high	N/A	N/A
SD_WP	I	Card write protect detect If not used tie low	N/A	N/A
SD_OD	O	Open drain select (not generated within the ESDHC). Optional output	N/A	N/A
SD_LCTL	O	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	0	N/A
SD_RST_N	O	Card hardware reset signal, active LOW	1	N/A

27.3 Functional Description

The following sections provide a brief functional description of the major system blocks, including the Data Buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock & reset manager and clock generator.

27.3.1 Data Buffer

The ESDHC uses one configurable data buffer, so that data can be transferred between the system bus (IP Bus or AHB Bus) and the SD card, with an optimized manner to maximize throughput between the two clock domains (that is, the IP peripheral clock, and the master clock). See the table below for illustration of the buffer scheme. The buffer is used as temporary storage for data being transferred between the host system and the card. The watermark levels for read and write are both configurable, and can be any number from 1 to 128 words. The burst lengths for read and write are also configurable, and can be any number from 1 to 31 words.

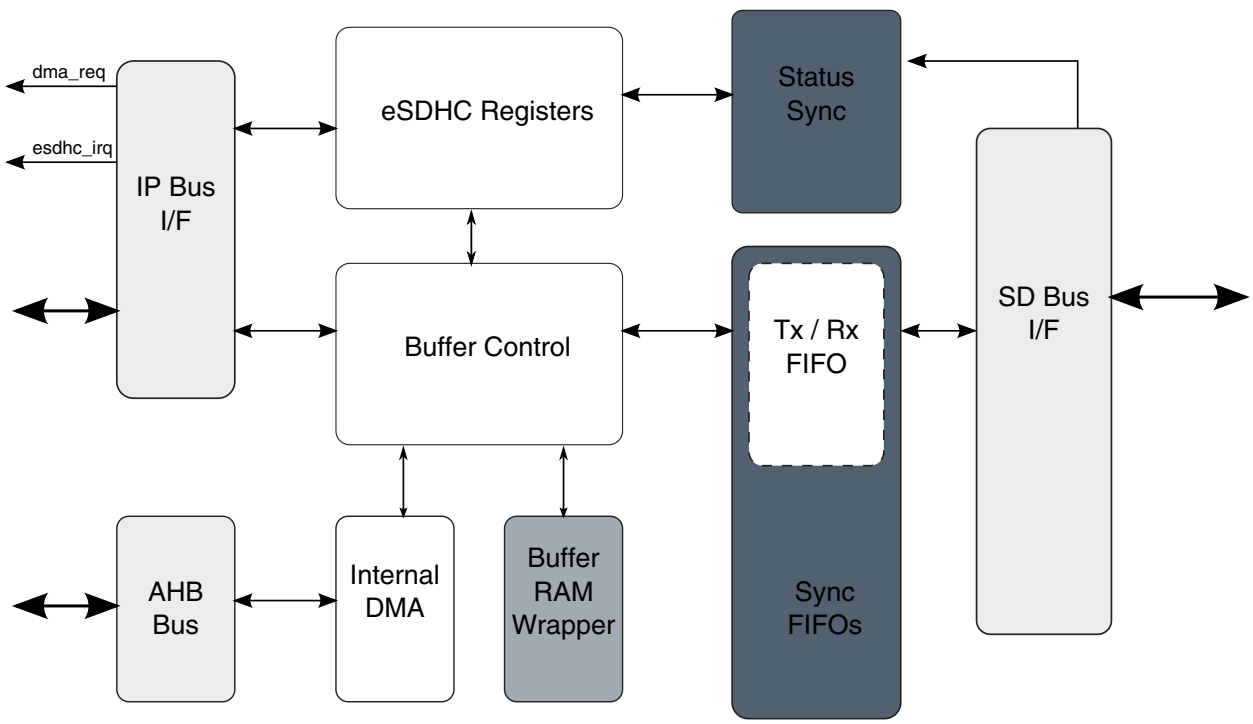


Figure 27-3. ESDHC Buffer Scheme

There are 3 transfer modes to access the data buffer:

- ARM platform polling mode:
  - For a host read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, then by polling the BRR bit the Host Driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar.
- External DMA mode:
  - For a read operation, when there are more words received in the buffer than the amount set in the RD_WML register, a DMA request is sent out to inform the external DMA to fetch the data. The request will be immediately de-asserted

when there is an access on the Buffer Data Port register. If the number of words in the buffer after the current burst meets or exceeds RD_WML value, then the DMA request is asserted again. For instance, if there are twice as many words in the buffer than the RD_WML value, there are two successive DMA requests with only one cycle of de-assertion between. The write operation is similar.

NOTE

Data buffer accesses by the ARM platform polling mode and external DMA mode both use the IP Bus. In these mode, if the external DMA is enabled, an external DMA request is sent out every time the buffer is ready.

- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus. For internal DMA access mode, the external DMA request will never be sent out.

    For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except INCR4 and INCR8, the burst type is always INCR mode and the burst length depends on the shortest of following factors:

1. Burst length configured in the burst length field of the Watermark Level register
2. Watermark Level boundary
3. Block size boundary
4. Data boundary configured in the current descriptor (if the ADMA is active)
5. 1 Kbyte address boundary defined in the AHB protocol

Write operation is similar.

Sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actually byte order is swapped inside the buffer, according to the endian mode configured by software. See Figure 26-4 and Figure 26-5. For a host write operation, byte order is swapped after data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, byte order is swapped before the data is stored into the buffer.

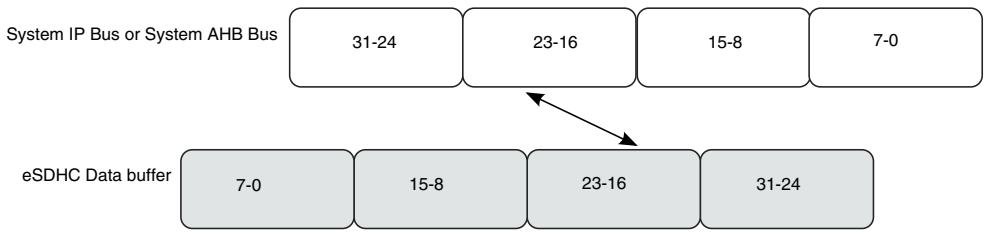


Figure 27-4. Data Swap between System Bus and ESDHC Data Buffer in Byte Little Endian Mode

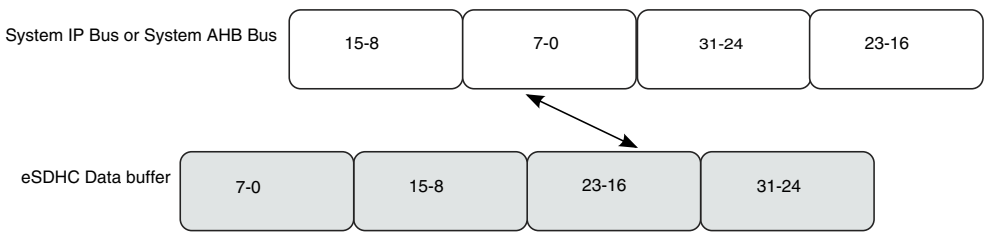


Figure 27-5. Data Swap between System Bus and ESDHC Data Buffer in Half Word Big Endian Mode

27.3.1.1 Write Operation Sequence

There are three ways to write data into the buffer when the user transfers data to the card:

1. By using external DMA through the ESDHC DMA request signal.
2. By processor core polling through the BWR bit in Interrupt Status register (interrupt or polling).
3. By using the internal DMA.

When the internal DMA is not used, (i.e. the DMAEN bit in the Transfer Type register is not set when the command is sent), the ESDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register, and is ready for receiving new data. At the same time, the ESDHC would set the BWR bit. The buffer write ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the ESDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the ESDHC will abort the data transfer and abandon the current block. The Host Driver should read the contents of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi block mode, the ESDHC will not automatically send CMD12, even though the

AC12EN bit in the Transfer Type register is set. The Host Driver shall send CMD12 in this scenario and re-start the write operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started after error recovery.

The ESDHC will not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the Host System does not write data in time, the ESDHC will stop the SD_CLK to avoid the data buffer under-run situation.

27.3.1.2 Read Operation Sequence

There are three ways to read data from the buffer when the user transfers data from the card:

1. By using the external DMA through the ESDHC DMA request signal
2. By processor core polling through the BRR bit in Interrupt Status register (interrupt or polling)
3. By using the internal DMA

When internal DMA is not used (i.e. DMAEN bit in Transfer Type register is not set when the command is sent), the ESDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register (that is, available and ready for system fetching data). At the same time, the ESDHC would set the BRR bit. The buffer read ready interrupt will be generated if it is enabled by software.

When internal DMA is used, the ESDHC will not inform the system before all the required number of bytes are transferred (if no error was encountered). When an error occurs during the data transfer, the ESDHC will abort the data transfer and abandon the current block. The Host Driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi block mode, the ESDHC will not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The Host Driver shall send CMD12 in this scenario and re-start the read operation from that address. It is recommended that a Software Reset for Data be applied before the transfer is re-started after error recovery.

For any read transfer mode, the ESDHC will not start data transmission until the number of words set in the RD_WML register are in the buffer. If the buffer is full and the Host System does not read data in time, the ESDHC will stop the SD_CLK to avoid the data buffer over-run situation.

27.3.1.3 Data Buffer and Block Size

The user needs to know the buffer size, for the buffer operation during a data transfer, to utilize it in the most optimized way. In the ESDHC, the only data buffer can hold up to 128 words (32-bit), and the watermark levels for write and read can be configured respectively. For both read and write, the watermark level can be from 1 word to the maximum of 128 words. For both read and write, the burst length, can be from 1 word to the maximum of 31 words. The Host Driver may configure the value according to the system situation and requirement.

During a multi-block data transfer, the block length may be set to any value between 1 and 4096 bytes inclusive which satisfies the requirements of the external card. The only restriction is from the external card. It might not support that large of a block or it does not support a partial block access (which is not the integer times of 512 bytes).

For block size not times of 4, that is, not word aligned, ESDHC requires stuff bytes at the end of each block, because ESDHC treats each block individually. For example, if the block size is 7 bytes and there are 12 blocks to write, the system side must write two times for each block. For each block, the ending byte will be abandoned by ESDHC because it only sends 7 bytes to the card and picks data from the following system write, making a total of 24 beats of write access.

27.3.1.4 Dividing Large Data Transfer

This SDIO command CMD53 definition, limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in block size units. If the total data length can't be divided evenly into a multiple of the block size, then there are two ways to transfer the data which depend on the function and the card design. Option 1 is for the Host Driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See figure below for an example showing the dividing of large data transfers. Assuming a kind of WLAN SDIO card only supports block size up to 64 bytes. Although the ESDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size less than64 bytes, so the data must be divided (see example below).

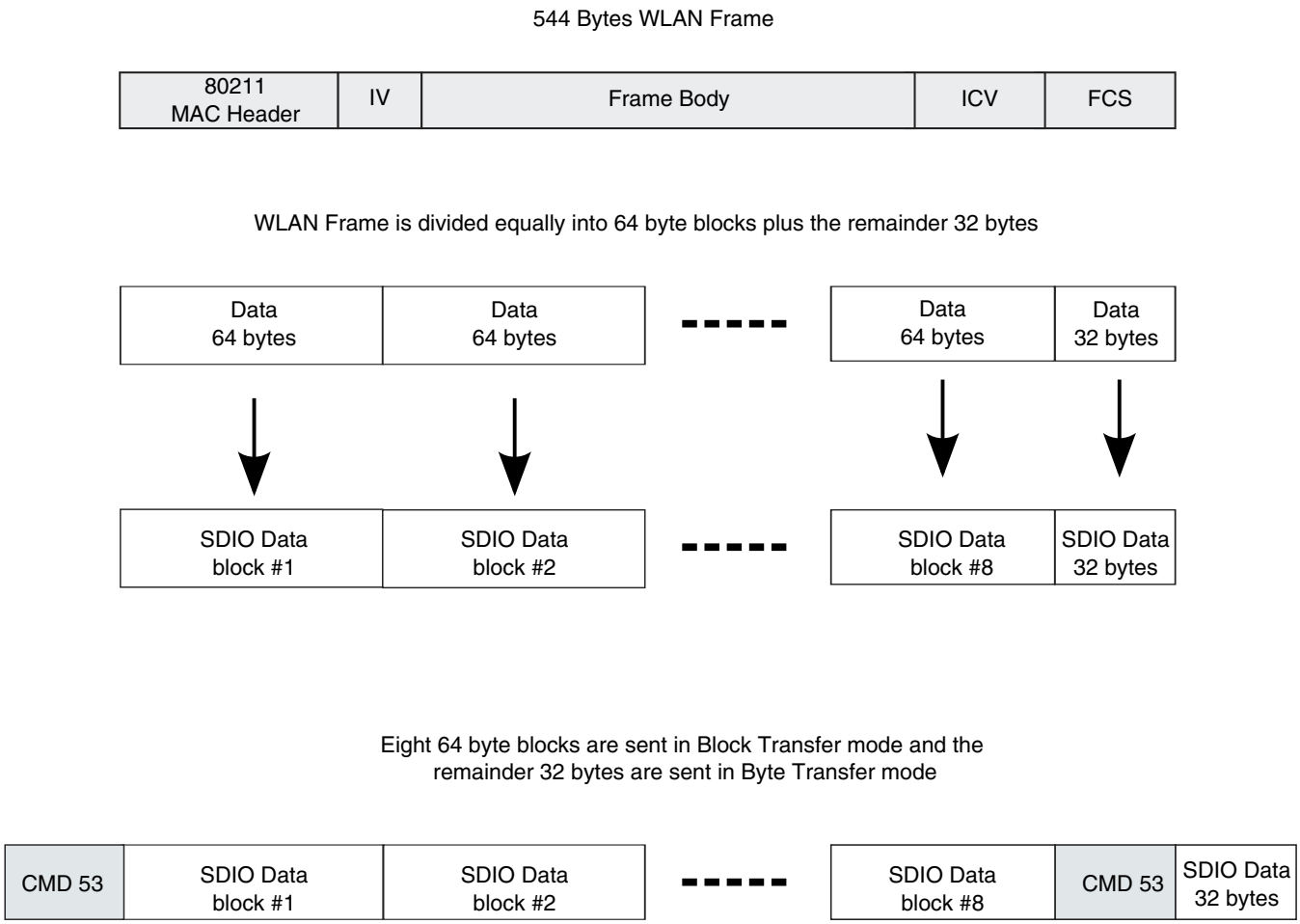


Figure 27-6. Example for Dividing Large Data Transfers

27.3.1.5 External DMA Request

When the internal DMA is not in use, and external DMA request is enabled, the Data Buffer will generate a DMA request to the system. During a write operation, when the number of WR_WML words can be held in the buffer free space, the signal esdhc_dreq_b is asserted to 0, informing the Host System of a DMA write. The BWR bit in the Interrupt Status register is also set, as long as the BWRSEN bit in the Interrupt Status Enable register is set. The DMA request is immediately de-asserted when an access to the Data Port register is made. If the buffer's free space still meets the watermark condition, the DMA request is asserted again after a cycle.

On read operation, when the number of RD_WML words are already in the buffer, the signal esdhc_dreq_b is asserted to 0, informing the Host System for a DMA read. The BRR bit in the Interrupt Status register is also set, as long as the BRRSEN bit in the

Interrupt Status Enable register is set. The DMA request is immediately de-asserted when an access to the Data Port register is made. If the buffer's data still meets the watermark condition, the DMA request is asserted again after a cycle.

Because the DMA burst length can not change during a data transfer for an external DMA transfer, the watermark level (read or write) must be a divisor of the block size. If it is not, transferring of the block may cause buffer under-run (read operation) or over-run (write operation). For example, if the block size is 512 bytes, the watermark level of read (or write) must be a power of two between 1 and 128. For processor core polling access, as the last access in the block transfer can be controlled by software, there is no such issue. The watermark level can be any value, even larger than the block size (but no greater than 128 words). This is because the actual number of bytes transferred by the software can be controlled and does not exceed the block size in each transfer.

The ESDHC also supports non-word aligned block size, as long as the card supports that block size. In this case, the watermark level should be set as the number of words. For example, if the block size is 31 bytes, the watermark level can be set to any number of word. For this case, the BLKSIZE bits of the Block Attribute register shall be set as 1fh. For the ARM platform polling access, the burst length can be 1 to 128 words, without restriction. This is because the software will transfer 8 words, and the ESDHC will also set the BWR or BRR bits when the remaining data does not violate data buffer. Refer to DMA Burst Length for more details about the dynamic watermark level of the data buffer. For the above example, even though 8 words are transferred through the Data Port register, the ESDHC will transfer only 31 bytes over the SD Bus, as required by the BLKSIZE bits. In this data transfer, with non-word aligned block size, the endian mode should be set cautiously, or invalid data will be transferred to/from the card.

27.3.2 DMA AHB Interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, the esdhc_dreq_b will not be asserted during the transfer, but the BWR and BRR bits will be set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register. See the figure below for an illustration of the DMA AHB interface block.

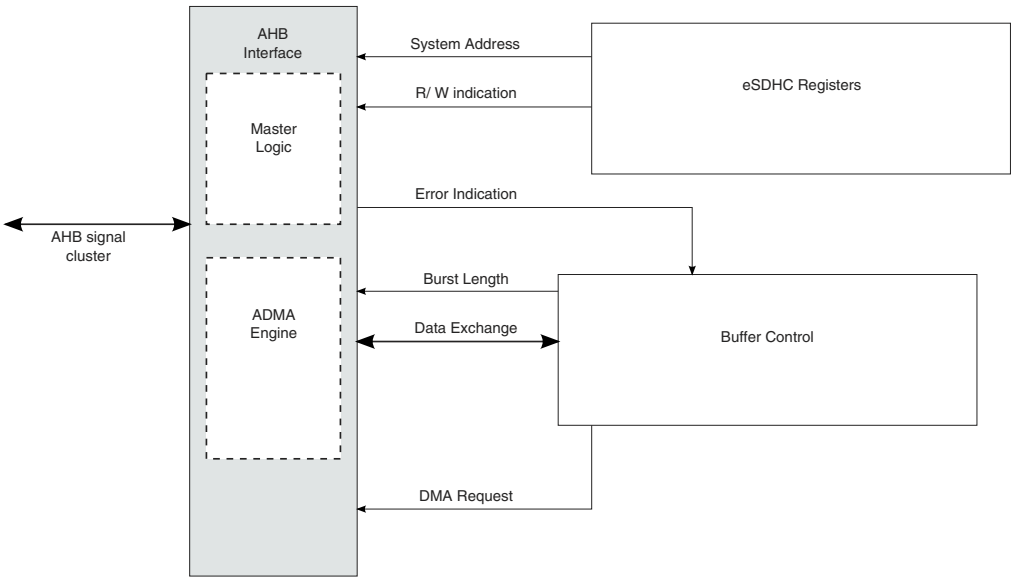


Figure 27-7. DMA AHB Interface Block

27.3.2.1 Internal DMA Request

If the watermark level requirement is met in data transfer, and the Internal DMA is enabled, the Data Buffer block will send a DMA request to AHB interface. Meanwhile, the external DMA request signal (esdhc_dreq_b) is disabled. The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to the ESDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The Data Buffer de-asserts the request once an access to the buffer is made. Upon access to the buffer by internal DMA, the Data Buffer updates its internal buffer pointer, and when the watermark level is satisfied, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to 6 words. After the first burst transfer, if there are more than 2 words in the buffer (which might contain some data of the next block), another DMA request read is sent. This is because the remaining number of words to send for the current block is (31 - 6 * 4) / 4 = 2. The ESDHC will read 2 words out of the buffer, with 7 valid bytes and 1 stuff byte.

27.3.2.2 DMA Burst Length

Just like a ARM platform polling access, the DMA burst length for the internal DMA engine can be from 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block. Take the example in Internal DMA Request again. The following burst length after 6 words are read will be 2 words, and the next burst length will be 6 words again. This is because the next block starts, which is 31 bytes, more than 6 words. The Host Driver writer may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length will be the same.

27.3.2.3 AHB Master Interface

It is possible that the internal AHB DMA engine could fail during the data transfer. When this error occurs, the DMA engine stops the transfer and goes to the idle state as well as the internal data buffer stops accepting incoming data. The DMAE bit in the Interrupt Status register is set to inform the driver.

Once the DMAE interrupt is received, the software shall send a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and re-start the transfer from this address to recover the corrupted block.

27.3.2.4 ADMA Engine

In the SD Host Controller Standard, the new DMA transfer algorithm called the ADMA (Advanced DMA) is defined. For Simple DMA, once the page boundary is reached, a DMA interrupt will be generated and the new system address shall be programmed by the Host Driver. The ADMA defines the programmable descriptor table in the system memory. The Host Driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized since the Host MCU intervention would not be needed during long DMA based data transfers.

There are two types of ADMA: ADMA1 and ADMA2 in Host Controller. ADMA1 can support data transfer of 4KB aligned data in system memory. ADMA2 removes the restriction so that data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA engine can recognize all kinds of descriptors define in SD Host Controller Standard, and if 'End' flag is detected in the descriptor, ADMA engine will stop after this descriptor is processed.

27.3.2.4.1 ADMA Concept and Descriptor Format

For ADMA1, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Set data length descriptor.
- Set data address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

For ADMA2, including the following descriptors:

- Valid/Invalid descriptor.
- Nop descriptor.
- Rsv descriptor.
- Set data length & address descriptor.
- Link descriptor.
- Interrupt flag and End flag in descriptor.

ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it will ignore the high 32-bit. Address field shall be set on word aligned (lower 2-bit is always set to 0). Data length is in byte unit.

ADMA will start read/write operation after it reaches the Tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last Set type descriptor before Tran type descriptor. Every Tran type will trigger a transfer, and the transfer data length is extracted from the most recent Set type descriptor. If there is no Set type descriptor after the previous Trans descriptor, the data length will be the value for previous transfer, or 0 if no Set descriptor is ever met.

For ADMA2, Tran type descriptor contains both data length and transfer data address, so only a Tran type descriptor can start a data transfer.

Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 27-8. Format of the ADMA1 Descriptor Table

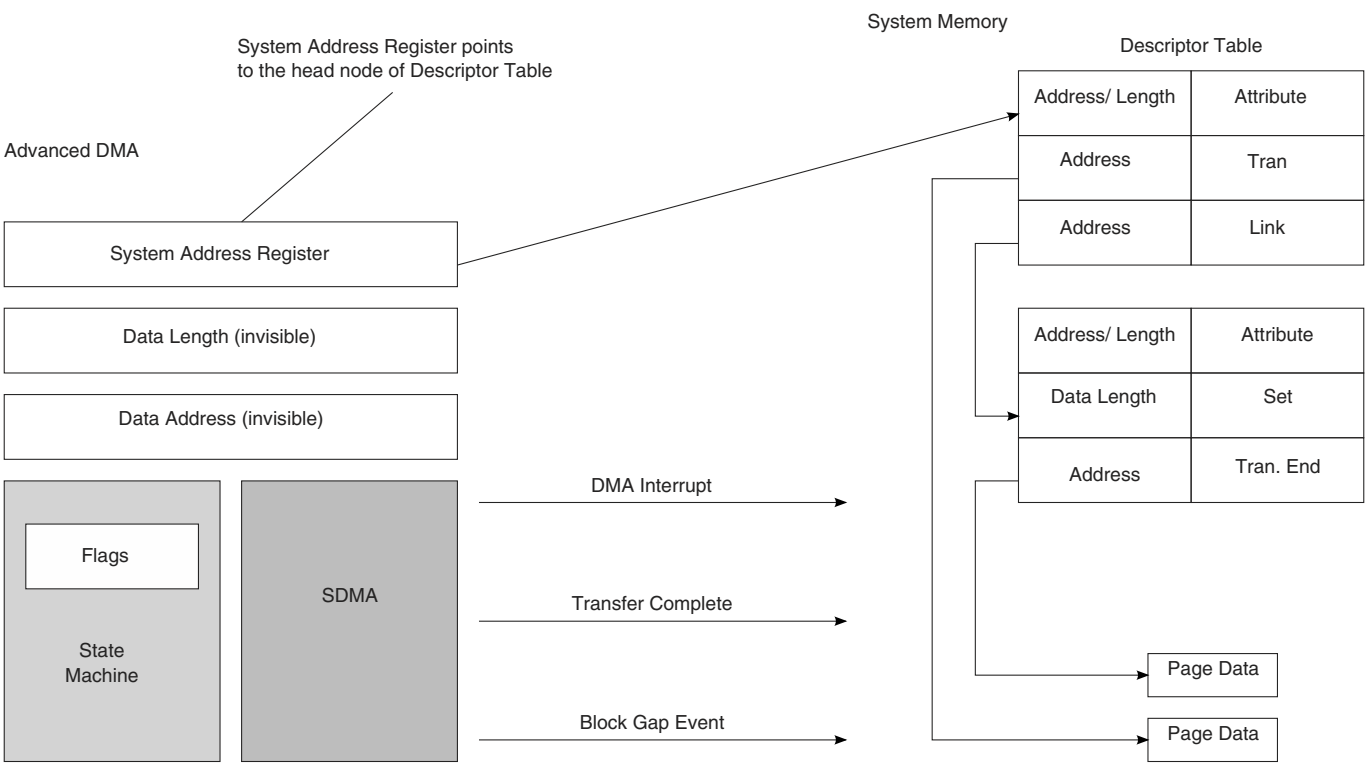


Figure 27-9. Concept and Access Method of ADMA1 Descriptor Table

Address/ Page Field		Address/ Page Field		Address/ Page Field		Attribute Field					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit Length		0000000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Don't Care
0	1	Rsv	Reserved	Same as Nop. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Valid	Valid = 1indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is done.

Figure 27-10. Format of the ADMA2 Descriptor Table

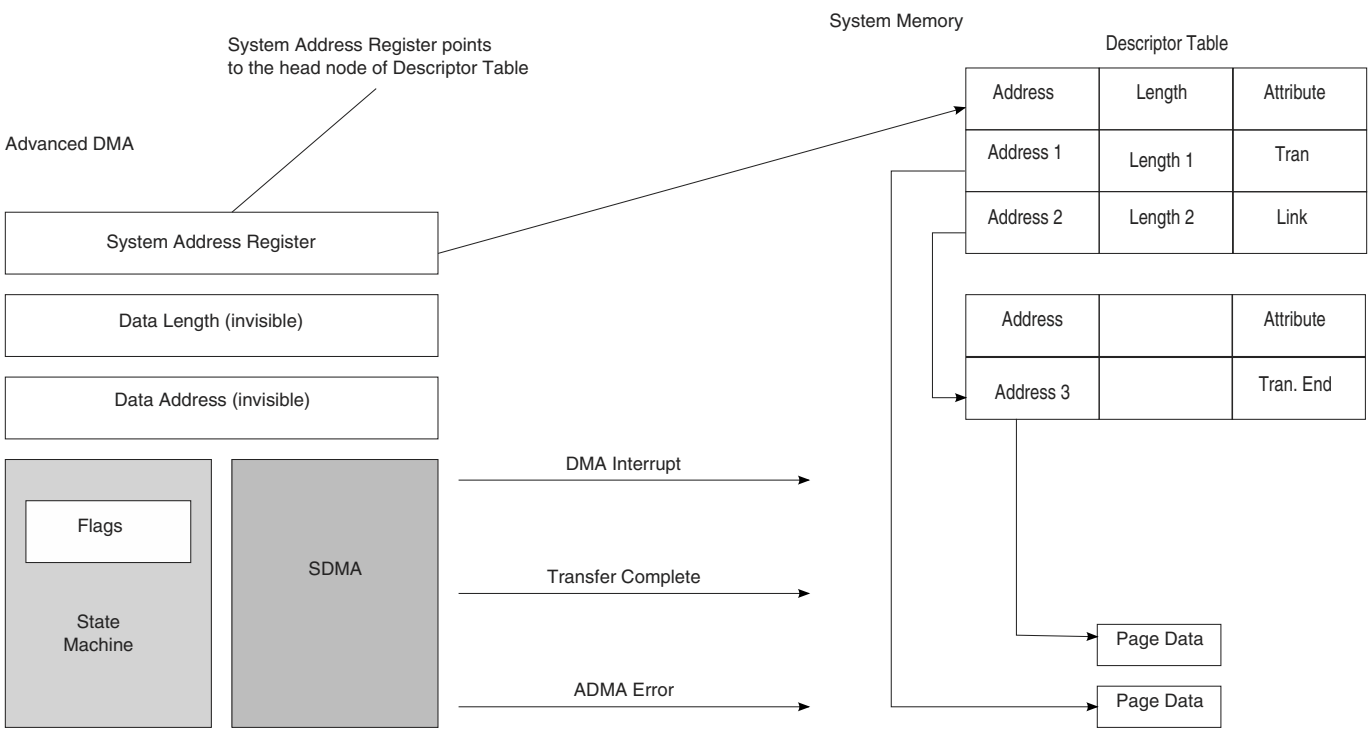


Figure 27-11. Concept and Access Method of ADMA2 Descriptor Table

27.3.2.4.2 ADMA Interrupt

If the 'Interrupt' flag of descriptor is set, ADMA will generate an interrupt according to different type descriptor:

For ADMA1:

- Set type descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type descriptor: interrupt is generated when new descriptor address is set.
- Nop type descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type descriptor: interrupt is generated just after fetch this descriptor.

27.3.2.4.3 ADMA Error-DMA

The ADMA will stop whenever any error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

ADMA descriptor error will be generated when it fails to detect 'Valid' flag in the descriptor. If adma descriptor error occurs, the interrupt is not generated even if the 'Interrupt' flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in Block Attributes register must equal to the whole data length set in descriptor nodes, otherwise data length mismatch error will be generated.

If BLKCNTEN bit is not set, the whole data length set in descriptor should be times of block length, otherwise, when all data set in the descriptor nodes are done not at block boundary, the data mismatch error will occur.

27.3.3 Register Bank with IP Bus Interface

Register accesses via the IP Bus interface are actually on the Register Bank. See the figure below for the block diagram.

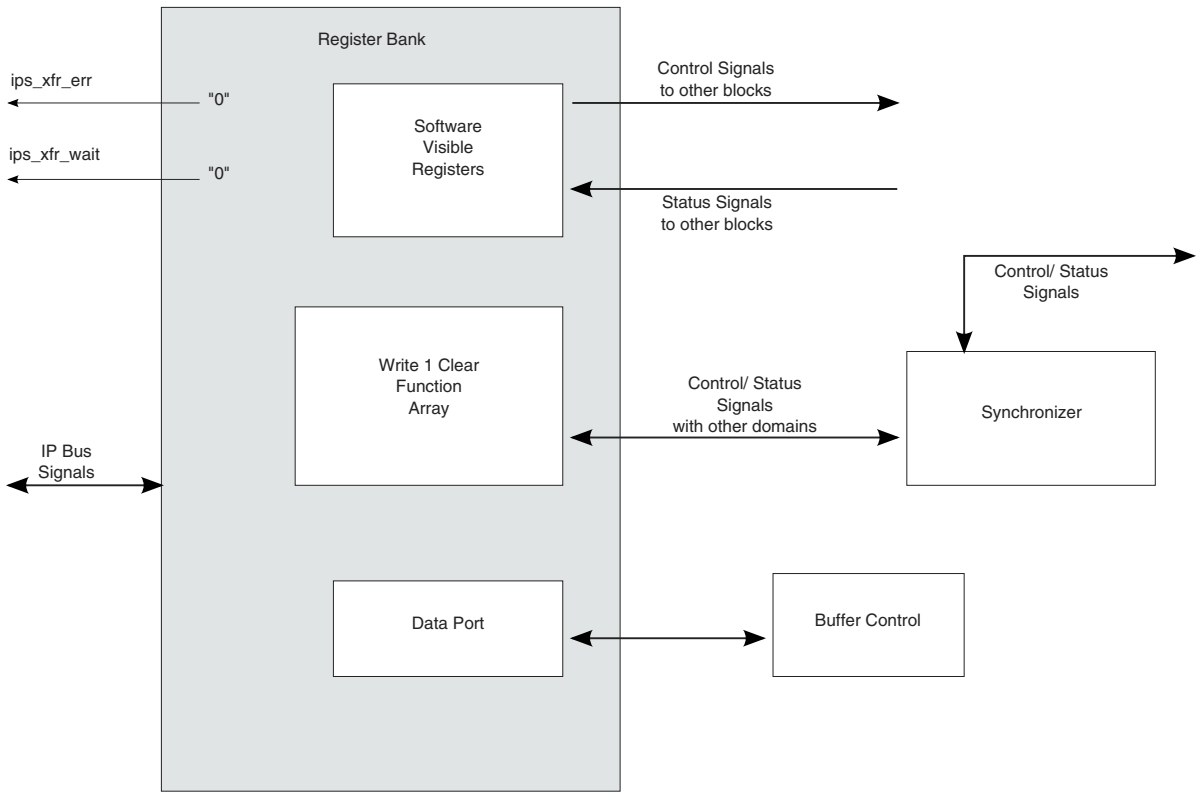


Figure 27-12. Register Bank Diagram

Only 32-bit access is allowed, and no partial read / write is supported, thus all accesses are word aligned.

27.3.4 SD Protocol Unit

The SD protocol unit deals with all SD protocol affairs.

The SD Protocol Unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the DAT[0] line
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status
- Detects the write protect state

The SD Protocol Unit consists of four sub-blocks:

1. SD transceiver.

2. SD clock and monitor.
3. Command agent.
4. Data agent.

27.3.4.1 SD Transceiver

In the SD protocol unit, the transceiver is the main control sub-block. It consists of an FSM and control sub-block, from which the control signals for all other three sub-blocks are generated.

27.3.4.2 SD Clock & Monitor

This sub-block monitors the signal level on all 8 data lines, the command lines, and directly routes the level values into the Register Bank. The driver can use this for debug purposes.

The sub-block also detects the Card Detection (CD) line as well as the DAT[3] line. The transceiver reports the card insertion state according to the CD state, or the signal level on the DAT[3] line, when the D3CD bit in the Protocol Control register is set.

The sub-block detects the Write Protect (WP) line. With the information of the WP state, the Register Bank will ignore the command, accompanied by a write operation, when the WP switch is on.

If the internal data buffer is in danger, and the SD clock must be gated off to avoid buffer over/under-run, this sub-block will assert the gate of the output SD clock to shut the clock off. After the buffer danger has recovered, and when the system access of the buffer catches up, the clock gate of this sub-block will open and the SD clock will be active again.

This sub-block also drive SD_LCTL output signal when the LCTL bit is set by the driver.

27.3.4.3 Command Agent

The Command Agent deals with the transactions on the CMD line. See figure below for an illustration of the structure for the Command CRC Shift Register.

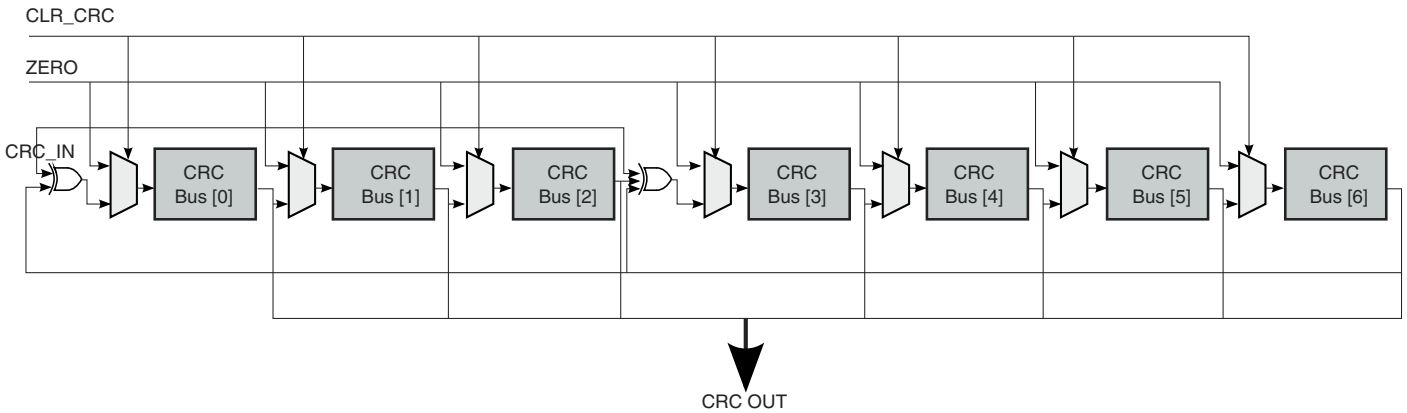


Figure 27-13. Command CRC Shift Register

The CRC polynomials for the CMD are as follows:

Generator polynomial: $G(x) = x^7 + x^3 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$

27.3.4.4 Data Agent

The Data Agent deals with the transactions on the eight data lines. Moreover, this sub-block also detects the busy state on the DAT[0] line, and generate the Read Wait state by the request from the Transceiver. The CRC polynomials for the DAT are as follows:

Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$
$CRC[15:0] = \text{Remainder } [(M(x) * x^{16}) / G(x)]$

27.3.5 Clock & Reset Manager

This sub-block controls all the reset signals within the ESDHC.

There are four kinds of reset signals within ESDHC:

1. Hardware reset.
2. Software reset for all.
3. Software reset for the data part.
4. Software reset for the command part.

All these signals are fed into this sub-block and stable signals are generated inside the module to reset all other modules. The sub-block also gates off all the inside signals.

There are three clocks inside the ESDHC:

1. ipg_clk.

2. ipg_perclk.
3. hclk.

The sub-block monitors the activities of all other sub-blocks, supplies the clocks for them, and when enabled, automatically gates off the corresponding clocks.

27.3.6 Clock Generator

The Clock Generator generates the SD_CLK by peripheral source clock in two stages. Refer to the figure below for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.

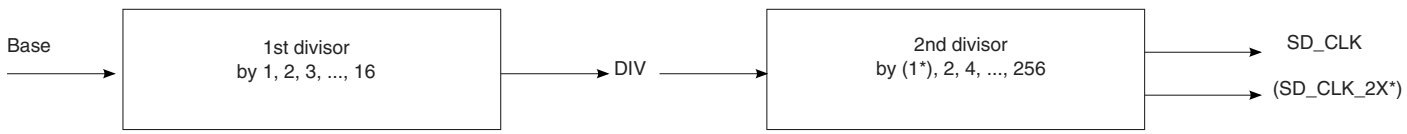


Figure 27-14. Two Stages of the Clock Divider

The first stage outputs an intermediate clock (DIV), which can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler, and outputs the actual clock (SD_CLK). Also it outputs DDR internal processing clock (SD_CLK_2X). These clocks are the driving clock for all sub-blocks of the SD Protocol Unit, and the sync FIFOs (see Figure 26-3) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage, can be DIV/2, DIV/4,..., or DIV/256. Thus the highest frequency of the SD_CLK is Base/2, while the lowest frequency is Base/4096. If the Base clock is of equal duty ratio (usually true), the duty cycle of SD_CLK/SD_CLK_2X is also 50%, even when the compound divisor is an odd value.

27.3.7 SDIO Card Interrupt

27.3.7.1 Interrupts in 1-bit Mode

In this case the DAT[1] pin is dedicated to providing the interrupt function. An interrupt is asserted by pulling the DAT[1] low from the SDIO card, until the interrupt service is finished to clear the interrupt.

27.3.7.2 Interrupt in 4-bit Mode

Since the interrupt and data line 1 share Pin 8 in 4-bit mode, an interrupt will only be sent by the card and recognized by the host during a specific time. This is known as the Interrupt Period. The ESDHC will only sample the level on Pin 8 during the Interrupt Period. At all other times, the host will ignore the level on Pin 8, and treat it as the data signal. The definition of the Interrupt Period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the Interrupt Period becomes active two clock cycles after the completion of a data packet. This Interrupt Period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in 4-bit mode, there is only a limited period of time that the Interrupt Period can be active due to the limited period of data line availability between the multiple blocks of data. This requires a more strict definition of the Interrupt Period. For this case, the Interrupt Period is limited to two clock cycles. This begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DAT[1] line will be held low for one clock cycle with the last clock cycle pulling DAT[1] high. On completion of the Interrupt Period, the card releases the DAT[1] line into the high Z state. The ESDHC samples the DAT[1] during the Interrupt Period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10f for further information about the SDIO card interrupt.

27.3.7.3 Card Interrupt Handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, the ESDHC clears the interrupt request to the Host System. The Host Driver should clear this bit before servicing the SDIO Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Status bit is cleared by resetting the SDIO interrupt. Writing to this bit would have no effects. In 1-bit mode, the ESDHC will detect the SDIO Interrupt with or without the SD clock (to support wakeup). In 4-bit mode, the interrupt signal is sampled during the Interrupt Period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status has been set, and the Host Driver needs to service this interrupt, so the SDIO bit in the Interrupt Control Register of SDIO card will be cleared. This is required to clear the SDIO interrupt status latched in the ESDHC and to stop driving the interrupt signal to the

System Interrupt Controller. The Host Driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Enable bit is set to 1, and the ESDHC starts sampling the interrupt signal again.

See figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

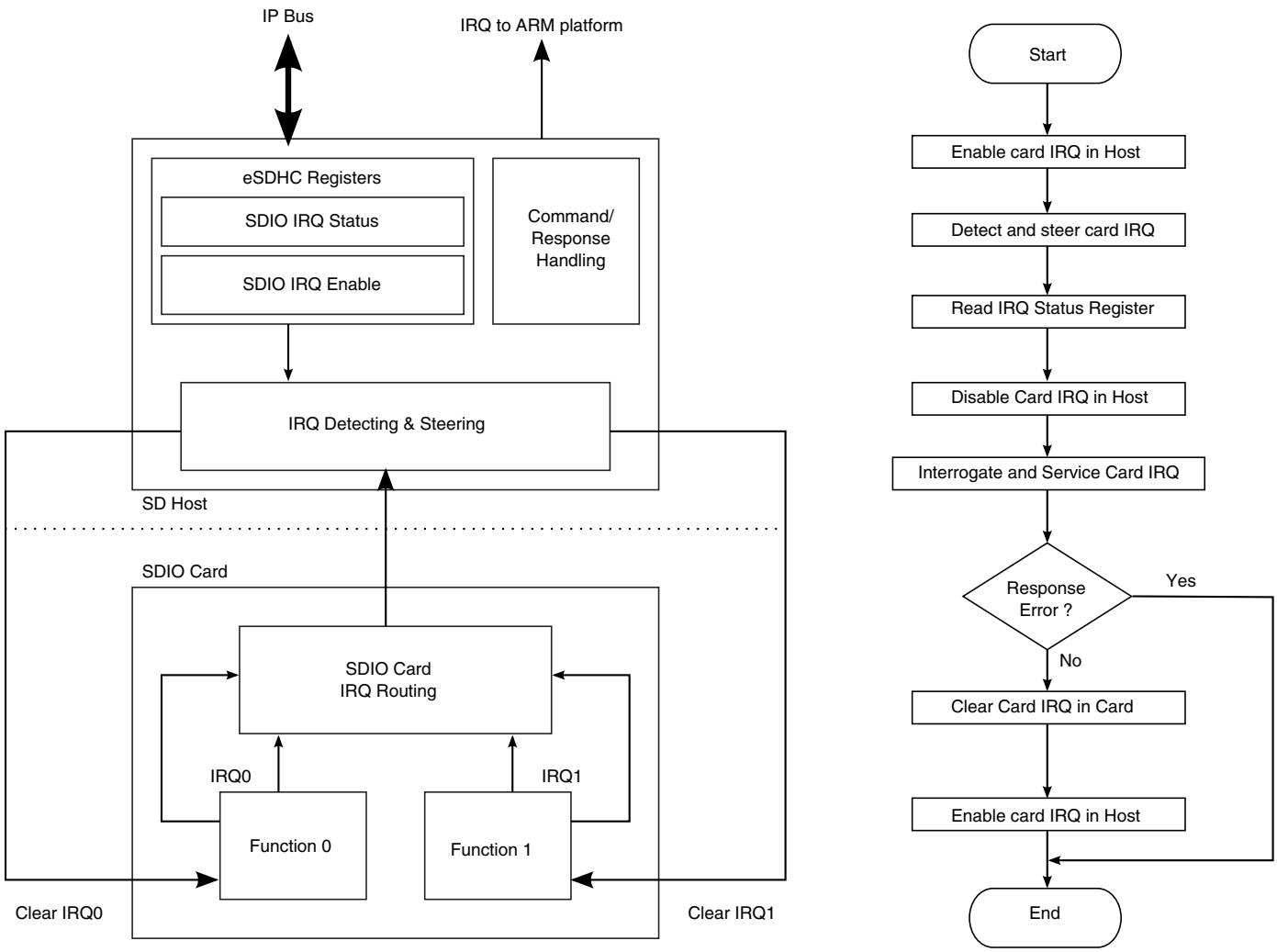


Figure 27-15. Card Interrupt Scheme and Card Interrupt Detection and Handling Procedure

27.3.8 Card Insertion and Removal Detection

The ESDHC uses either the DAT[3] pin or the CD pin to detect card insertion or removal. When there is no card on the MMC/SD bus, the DAT[3] will be pulled to a low voltage level by default. When any card is inserted to or removed from the socket, the ESDHC detects the logic value changes on the DAT[3] pin and generates an interrupt. When the DAT[3] pin is not used for card detection (for example, it is implemented in GPIO), the CD pin must be connected for card detection. Whether DAT[3] is configured for card detection or not, the CD pin is always a reference for card detection. Whether the DAT[3] pin or the CD pin is used to detect card insertion, the ESDHC will send an interrupt (if enabled) to inform the Host system that a card is inserted.

27.3.9 Power Management and Wake Up Events

When there is no operation between the ESDHC and the card through the SD bus, the user can completely disable the ipg_clk and ipg_perclk in the chip level clock control module to save power. When the user needs to use the ESDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to the ESDHC are disabled, for instance, when the system is in low power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The ESDHC can generate these interrupt even when there are no clocks enabled. The three interrupts which can be used as wake up events are:

1. Card Removal Interrupt
2. Card Insertion Interrupt
3. Interrupt from SDIO card

The ESDHC offers a power management feature. By clearing the clock enabled bits in the System Control Register (ESDHC_SYSCTL), the clocks are gated in the low position to the ESDHC. For maximum power saving, the user can disable all the clocks to the ESDHC when there is no operation in progress.

These three wake up events (or wakeup interrupts) can also be used to wake up the system from low-power modes.

NOTE

To make the interrupt a wakeup event, when all the clocks to the ESDHC are disabled or when the whole system is in low power mode, the corresponding wakeup enabled bit needs to be set. Refer to Protocol Control (ESDHCV3_PROCTL) for more information on the ESDHC Protocol Control register.

27.3.9.1 Setting Wake Up Events

For the ESDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the ARM platform enters sleep mode. Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No Read or Write Transfer is active
- Data and Command lines are not active
- No interrupts are pending
- Internal data buffer is empty

27.3.10 MMC Fast Boot

The Embedded Multimedia Card (eMMC 4.3) has a fast boot feature. In boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

There are two supported types of fast boot mode, boot operation and alternative boot operation. Each type has with-acknowledge and without-acknowledge modes.

NOTE

For the eMMC 4.3 card setting, please see the eMMC 4.3 spec.

27.3.10.1 Boot Operation

NOTE

In this block guide, this fast boot is called normal fast boot mode.

If the CMD line is held LOW for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after the CMD line goes LOW, the slave starts to send the first boot data to the master on the DAT line(s). The master must keep the CMD line LOW to read all of the boot data.

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes LOW. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode with the CMD line HIGH.

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

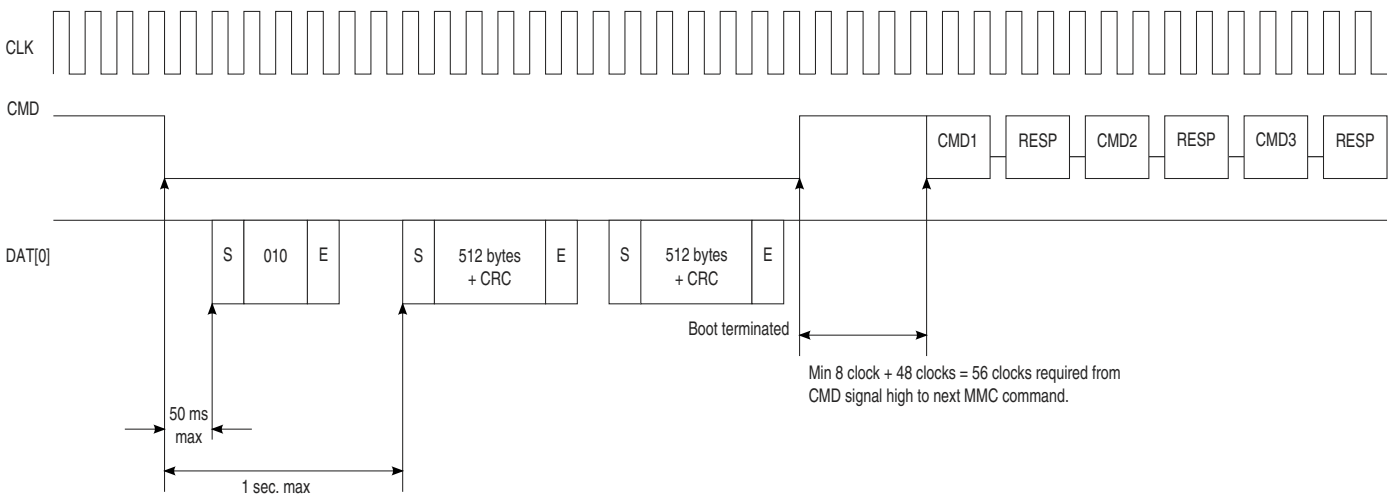


Figure 27-16. MultiMediaCard state diagram (normal boot mode)

27.3.10.2 Alternative Boot Operation

If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

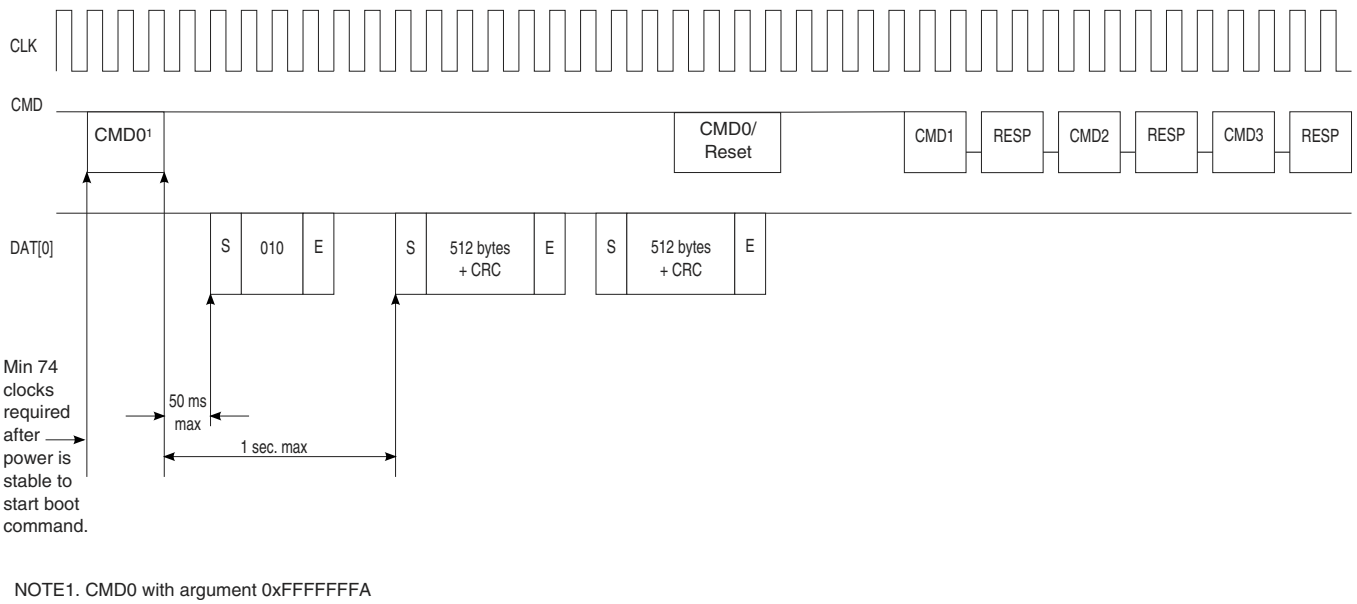
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within 1 second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DAT line(s).

If boot acknowledge is enabled, the slave has to send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, the slave will not send out acknowledge pattern '010'.

The master can terminate boot mode by issuing CMD0 (Reset).

Boot operation will be terminated when all contents of the enabled boot data are sent to the master. After boot operation is executed, the slave shall be ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



NOTE1. CMD0 with argument 0xFFFFFFFA

Figure 27-17. MultiMediaCard state diagram (alternative boot mode)

27.4 Initialization/Application of ESDHC

All communication between system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as "GO_IDLE_STATE", "SEND_OP_COND", "ALL_SEND_CID" and etc. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for MMC/SD/SDIO for the commands of bc and bcr categories.

After the Broadcast command CMD3 is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DAT I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO for the commands of ac and adtc categories.

27.4.1 Command Send & Response Receive Basic Operation

Assuming the data type WORD is an unsigned 32-bit integer, the below flow is a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
          set MSBSEL bit;
          if (finite block number) {
                    set BCEN bit;
                    if (auto12 command is to use) set AC12EN bit;
          }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. By doing this, make sure the corresponding interrupt status bits are enabled.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and go to the Standby State, no response to the Host when CMD2 is sent. The Host Driver shall deal with 'fake' errors like this with caution.

27.4.2 Card Identification Mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for the MMC cards.

27.4.2.1 Card Detect

See figure below for a flow diagram showing the detection of MMC, SD and SDIO cards using the ESDHC.

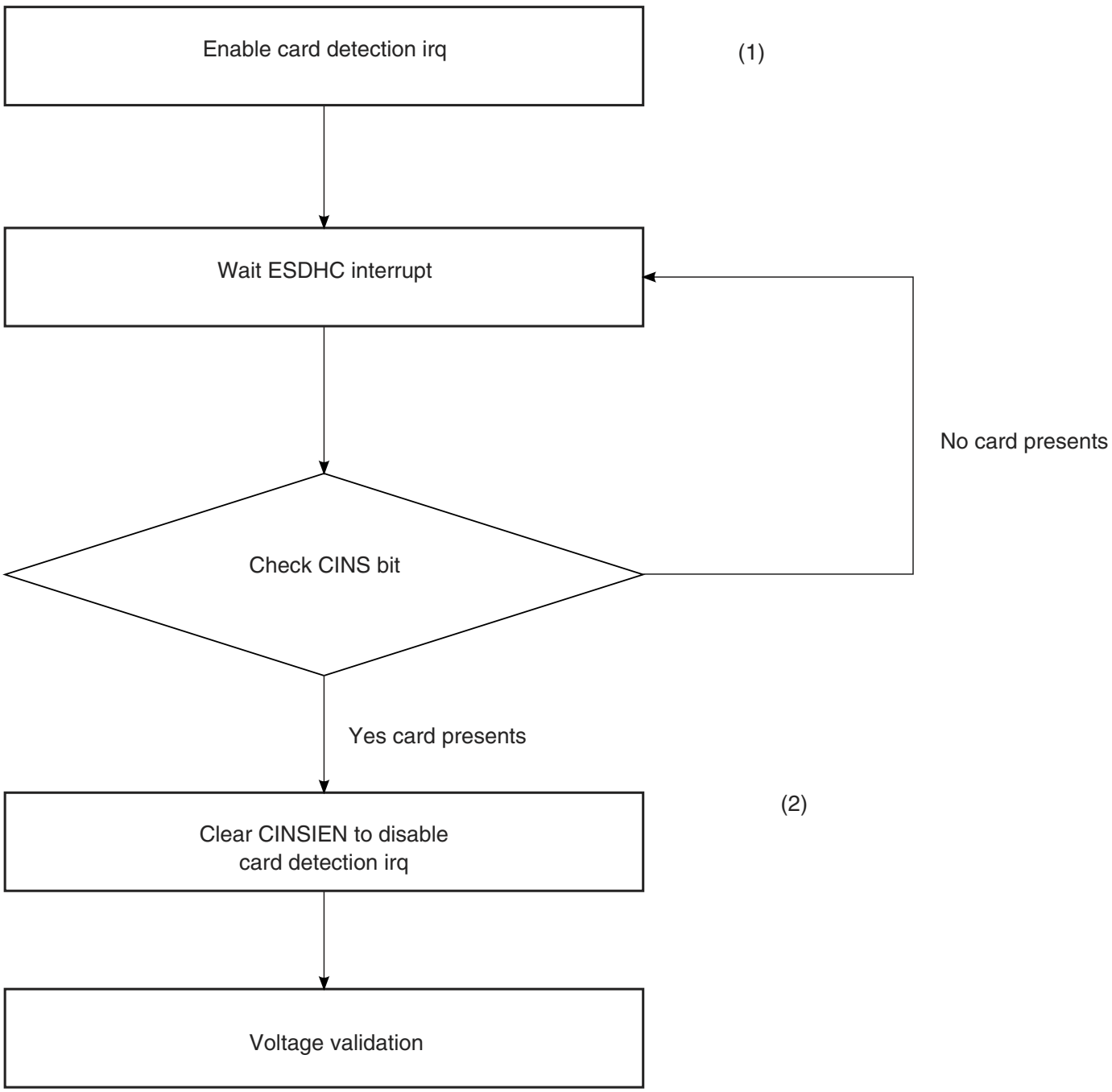


Figure 27-18. Flow Diagram for Card Detection

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt
- When an interrupt from the ESDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards

27.4.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) which is driven by POR (Power On Reset)
- Software reset (Host Only) is proceed by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the Host Controller, respectively
- Card reset (Card Only). The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards, SD Memory cards. This command sets each card into the Idle State regardless of the current card state. For an SD I/O Card, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See figure below for the software flow to reset both the ESDHC and the card.

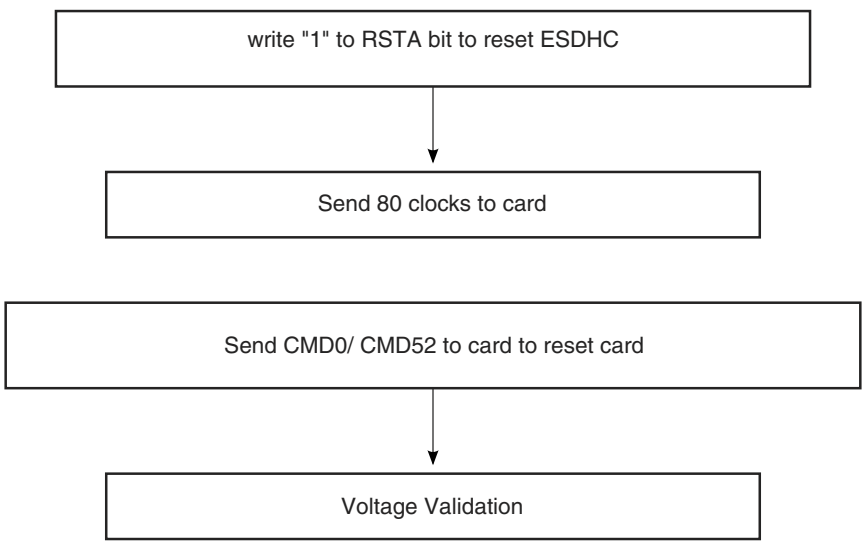


Figure 27-19. Flow Chart for Reset of the ESDHC and SD I/O Card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DVS and SDCLKFS bit fields to get the SD_CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

27.4.2.3 Voltage Validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for Vdd are defined in the Operation Conditions Register (OCR) and may not cover the whole range. Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer Vdd conditions. This means if the host and card have non-common Vdd ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the Vdd range(s) desired by the host. This is accomplished by the host sending the desired Vdd voltage window as the operand of this command. Cards that can't perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification shall be sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
operation voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
          if (0 < number of IO functions) {
                    label the card as SDIO;
                    IORDY = 0;
                    while (!(IORDY in IO OCR response)) { // set voltage range for each IO
function
                              send_command(IO_SEND_OP_COND, <voltage range>, <other
parameter>);
                              wait_for_response(IO_SEND_OP_COND);
                    } // end of while ...
          } // end of if (0 < ...
          if (memory part is present inside SDIO card) Label the card as SDCombo; // this is
an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range
is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific
CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
          send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage
```

```
range
for memory part or SD card
          wait_for_response(SD_APP_OP_COND); // voltage range is set
          if (card type is UNKNOWN) label the card as SD;
          return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
          deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refuse, it must be MMC card           if (card is already labelled as
SDCombo) { // change label
                    re-label the card as SDIO;
                    ignore the error or report it;
                    return; // card is identified as SDIO card
          } // of if (card is ...
          send_command(SEND_OP_COND, <voltage range>, <...>);
          if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted,
either
                    label the card as UNKNOWN;
                    return;
          } // of if (RESP_TIMEOUT ...
          if (check for CE-ATA signature succeeded) { // the card is CE-ATA
                    store CE-ATA specific info from the signature;
                    label the card as CE-ATA;
          } // of if (check for CE-ATA ...
          else label the card as MMC;
} // of else
}
```

27.4.2.4 Card Registry

Card registry for the MMC and SD/SDIO/SD Combo cards are different.

For the SD Card, the Identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the Card spec). At this time, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host will request the card to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are put into the Inactive State. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready State), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification State.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA will be used to address the card for future data transfer operations. Once the RCA is received, the card changes its state to the Standby State. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in system.

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (the cards in Ready State) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into the Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

For operation as MMC cards:

```
card_registry()
{
do { // decide RCA for each card until response time-out
          if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
                    send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to
publish its
RCA
                    retrieve RCA from response;
          } // end if (card is labelled as SDCombo ...
          else if (card is labelled as SD) { // for SD card
                    send_command(ALL_SEND_CID, <...>);
                    if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                    send_command(SET_RELATIVE_ADDR, <...>);
                    retrieve RCA from response;
          } // else if (card is labelled as SD ...
          else if (card is labelled as MMC or CE-ATA) { // treat CE-ATA as MMC
                    send_command(ALL_SEND_CID, <...>);
                    rca = 0x1; // arbitrarily set RCA, 1 here for example, this RCA is also
the
relative address to access the CE-ATA card
                    send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper
16
bits
          } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

27.4.3 Card Access

27.4.3.1 Block Write

27.4.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card shall indicate the failure on the DAT line. The transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card will respond with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby State and release the DAT line without interrupting the write operation. When re-selecting the card, it will reactivate the busy indication by pulling DAT to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other two methods (by means of external DMA or ARM platform polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   a. For SD/MMC cards, use SET_BLOCKLEN (CMD16)
   b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the ESDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the ESDHC block attribute register, block number is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the ESDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

27.4.3.1.2 DDR Write

ESDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described as below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the ESDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the ESDHC DMA when sending the command with data transfer.The DDR_EN_IPG bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

27.4.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the SD_CLK at any time to pause all the operations, which is also inaccessible to the Host Driver, the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer

between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DAT0 line is not required to de-assert to release the busy state, no suspend command is needed.

Like in the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a. For SD/MMC, use SET_BLOCKLEN (CMD16)
   b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the ESDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the ESDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the ESDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The Driver shall read the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the Driver should treat this as a non-pause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the Host System is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the Suspend Command for the SDIO card; when such a command is sent, the ESDHC thinks the System will switch to another

function on the SDIO card, and flush the data buffer. The ESDHC takes the Resume Command as a normal command with data transfer, and it is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the ESDHC will automatically send a CMD12 to mark the end of the multi-block transfer.

27.4.3.2 Block Read

27.4.3.2.1 Normal Read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer State. For multi blocks read, data blocks will be continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other two methods (by means of external DMA or ARM platform polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a. For SD/MMC, use SET_BLOCKLEN (CMD16)
   b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
3. Set the ESDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the ESDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the ESDHC DMA when sending the command with data transfer. The AC12EN bit should also be set:
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

27.4.3.2.2 DDR Read

ESDHC supports dual data rate mode.

The software flow to write to a card in ddr mode is described below:

1. Check the card status, wait until the card is ready for data.
2. For eMMC4.4 card, block length only can be set to 512byte.
3. Set the ESDHC number block register (NOB), nob is 5 (for instance).
4. Set eMMC4.4 card to high speed mode, use SWITCH(CMD6).
5. Set eMMC4.4 card bus with (4-bit /8-bit ddr mode), use SWITCH(CMD6).
6. Disable the buffer write ready interrupt, configure the DMA settings and enable the ESDHC DMA when sending the command with data transfer.The DDR_EN_IPG bit should be set. The AC12EN bit should also be set.
7. Wait for the Transfer Complete interrupt.
8. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

27.4.3.2.3 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the Driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks. Before setting the SABGREQ bit, make sure the RWCTL bit in the Protocol Control register is set, otherwise the ESDHC will not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended that the RWCTL bit be set once the Read Wait capability of the SDIO card is recognized.

Like in the flow described in Normal Read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports Read Wait.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
   a. For SD/MMC, use SET_BLOCKLEN (CMD16)
   b. For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7)
5. Set the ESDHC block length register to be the same as the block length set for the card in Step 2.

6. Set the ESDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the ESDHC DMA when sending the command with data transfer. The AC12EN bit should also be set
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the ESDHC will ignore the Stop At Block Gap Request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause will be transferred to the Host System. No matter if the Suspend Command is sent or not, the internal data buffer is not flushed.

If the Suspend Command is sent and the transfer is later resumed by means of a Resume Command, the ESDHC takes the command as a normal one accompanied with data transfer. It is left for the Driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the ESDHC will automatically send the CMD12 to mark the end of multi-block transfer.

27.4.3.2.4 DLL (Delay Line) in Read Path

The DLL (Delay Line) is added to assist in sampling read data. As in The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage and temperature (PVT). The reasons why the DLL is needed for ESDHC are 1.) the path of read data traveling from card to host varies. 2.) in MMC DDR mode the minimum input setup and hold time are both at 2.5 ns. The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided perclk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in override mode. The override value set is the number of delay cells. In override mode, the DLL_enable is no need to set. And another working mode of DLL is target value mode. In this mode, DLL will automatically adjust the number of delay cells according to the target value your set value and PVT changes. Be aware that target value is in the unit of 1/32 clock reference period. If the perclk is 100MHz, then the reference clock period is 10ns, setting target value of 16 means 5ns=(16/32)*10ns. Software can disable automatically update by setting dll_gate_update bit. Please refer to Figure 27-20.

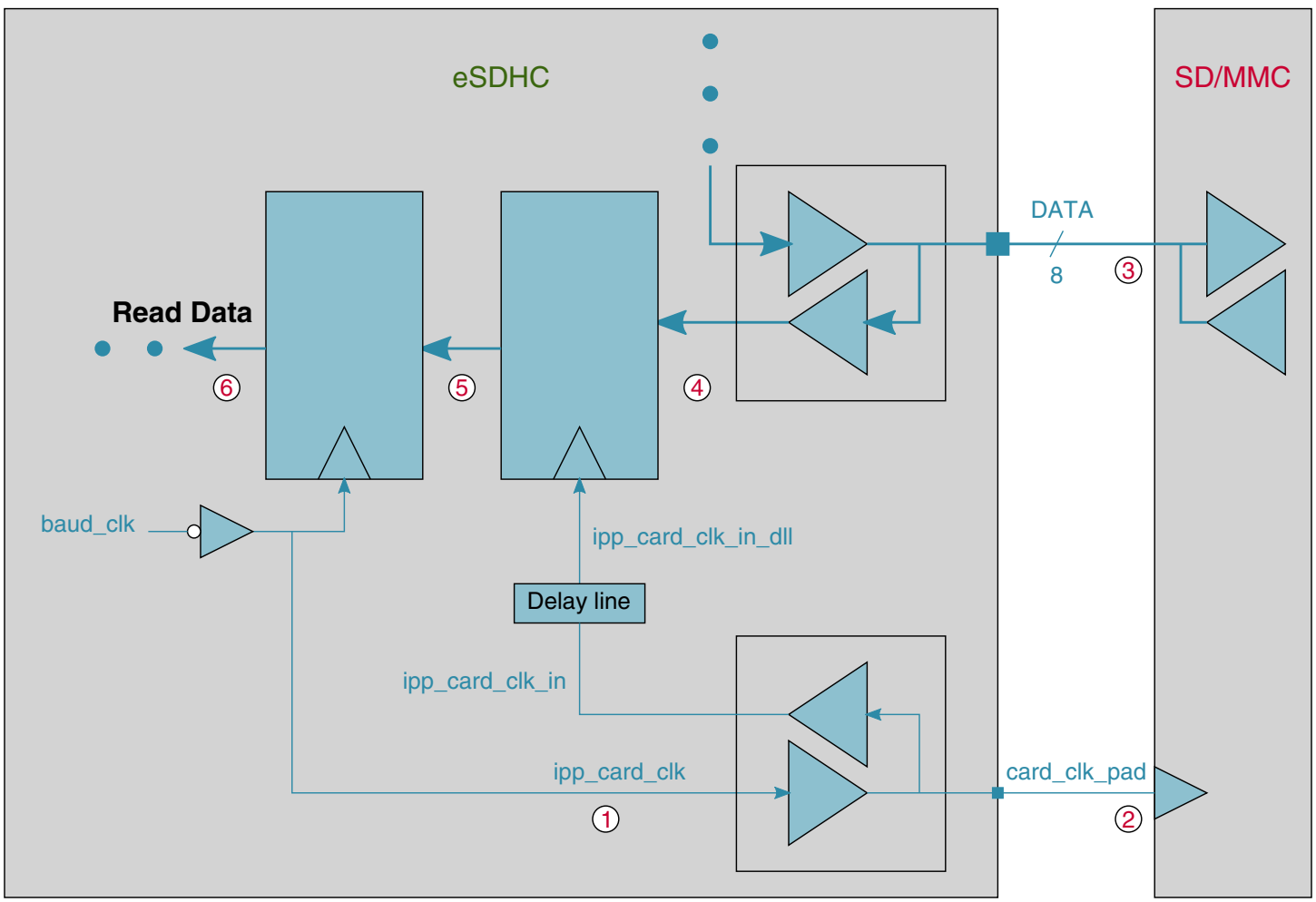


Figure 27-20. DLL (Delay Line) in Read Path

27.4.3.3 Suspend Resume

The ESDHC supports the Suspend Resume operations of SDIO cards, although slightly different than the suggested implementation of Suspend in the SDIO card specification.

Suspend

After setting the SABGREQ bit, the Host Driver may send a Suspend command to switch to another function of the SDIO card. The ESDHC does not monitor the content of the response, so it doe not know whether or not the Suspend command has succeeded. Accordingly, it does not de-assert Read Wait for read pause. To solve this problem, the Driver does not set the ESDHC_XFERTYP[CMDTYP] register to 01, that is, Suspend option. Instead, the Driver sends this command as if it were a normal command (that is, sets CMDTYP to b00). Only when the command succeeds, and the BS bit is set in the response, does the Driver send another command marked as "Suspend" to inform the ESDHC that the current transfer is suspended. This is shown in the following sequence for Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the ESDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the ESDHC stops driving DAT2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

27.4.3.3.1 Resume

To resume the data transfer, a Resume command shall be issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation above.
2. Send the Resume command. In the Transfer Type register, all bit fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer will be resumed.

27.4.3.4 ADMA1 Usage

To use the ADMA1 in a data transfer, the Host Driver must prepare the correct descriptor chain prior to sending the read/write command. The steps to accomplish this are:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4kB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.
4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors match the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

27.4.3.5 Transfer Error

27.4.3.5.1 CRC Error

It is possible at the end of a block transfer, that a write CRC status error or read CRC error occurs. For this type of error the latest block received shall be discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one. For a multi-block transfer, the Host Driver shall issue a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, the ESDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 will be sent by the ESDHC. In this case, the Driver shall re-send or re-obtain the last block with a single block transfer.

27.4.3.5.2 Internal DMA Error

During the data transfer with internal Simple DMA, if the DMA engine encounters some error on the AHB bus, the DMA operation is aborted and DMA Error interrupt is sent to the Host System. When acknowledged by such an interrupt, the Driver shall calculate the start address of data block in which the error occurs. The start address can be calculated by either:

1. Read the DMA System Address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the next burst transfer into account, it is straight forward to obtain the start address of the corrupted block.
2. Read the BLKCNT field of the Block Attribute register. By the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined. When the BCEN bit is not set, the contents of the Block Attribute register does not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and re-start the transfer from the corrupted block to recover from the error.

27.4.3.5.3 ADMA Error-Card Access

There are three kinds of possible ADMA errors. The AHB transfer, invalid descriptor, and data-length mismatch errors. When these errors occur, the DMA transfer stops and the corresponding error status bit is set. For acknowledging the status, the Host Driver should recover the error as shown below and re-transfer from the place of interruption.

1. AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or the next block if no block is corrupted.
2. Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and re-create the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the previous setting, the data length configured in the new descriptor chain should match the new value.
3. Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain, and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

27.4.3.5.4 Auto CMD12 Error

After the last block of the multi block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, the ESDHC automatically sends a CMD12 to the card to stop the transfer. When errors with this command occur, it is recommended to the Driver to deal with the situations in the following manner:

1. Auto CMD12 response time-out. It is not certain whether the command is accepted by the card or not. The Driver should clear the Auto CMD12 error status bits and re-send the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error. Since card responds to the CMD12, the card will abort the transfer. The Driver may ignore the error and clear the error status bit.
3. Auto CMD12 conflict error or not sent. The command is not sent, so the Driver shall send a CMD12 manually.

27.4.3.6 Card Interrupt

The external cards can inform the Host Controller by means of some special signals. For the SDIO card, it can be the low level on the DAT[1] line during some special period. The ESDHC only monitors the DAT[1] line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the ESDHC, and the Host System is informed by the ESDHC asserting the ESDHC interrupt line, the interrupt service from the Host Driver is called.

As the interrupt factor is controlled by the external card, the interrupt from the SDIO card must be served before the CINT bit is cleared by written 1. Refer to Card Interrupt Handling for the card interrupt handling flow.

27.4.4 Switch Function

MMC cards transferring data at bus widths other than 1-bit is a feature in MMC spec. The high speed timing mode for all card devices, was also defined in various card specifications. To enable these features, a "switch" command shall be issued by the Host Driver.

For SDIO cards, the high speed mode is enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode is queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards, the high speed mode is queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The 4-bit and 8-bit bus width of the MMC is also enabled by the SWITCH command, but with a different argument.

DDR mode is also selected by setting ddr mode bus width via SWITCH.

These new functions can also be disabled by a software reset. For SDIO cards it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following flowcharts do not show current capability check, which is recommended in the function switch process.

27.4.4.1 Query, Enable and Disable SDIO High Speed Mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
cleared;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

27.4.4.2 Query, Enable and Disable SD High Speed Mode

```
enable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFF1 and read 64 bytes of data accompanying the R1 response;
wait data transfer done bit is set;
check if the bit 401 of received 512 bit is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
send CMD6, with argument 0x80FFFFF1 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

27.4.4.3 Query, Enable and Disable MMC High Speed Mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into ESDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

27.4.4.4 Set MMC Bus Width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

27.4.5 ADMA Operation

27.4.5.1 ADMA1 Operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
```

```
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB align);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

27.4.5.2 ADMA2 Operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

27.4.6 Fast Boot Operation

27.4.6.1 Normal fast boot flow

1. Software need to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.

2. Software need to configure MMC Boot Register (offset 0xc4) bit 6 to 1(enable boot), and bit 5 to 0(normal fast boot), and bit 4 to select the ack mode or not. If need to send through DMA mode, need to configure bit 7 to enable automatically stop at block gap feature. And need to configure bit 3-bit0 to select the ack timeout value according to the sd clk frequence.
3. Software then need to configure Block Attributes Register to set block size/no. If in ddr fast boot mode, block size only can be configure to 512byte.
4. Software need to configure Protocol control register to set DTW (data transfer width). If in ddr fast boot mode, DTW only can be configure to 4-bit/8-bit dataline mode.
5. Software need to configure Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software need to configure Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.

NOTE

DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1, better to configure blk no in Bock Attributes Register to the max value. And if in ddr fast boot mode, DDR_EN need to be set to 1.

7. When the step 6 is configured, boot process will begin. Software need to poll the data buffer ready status to read the data from buffer in time. If boot time-out happened (ack time out or the first data read time out), Interrupt will be triggered, and software need to configure MMC Boot Register to bit 6 to 0 to disable boot. Thus will make CMD high, and then after at least 56 clocks, it is ready to begin normal initialization process.
8. If no time out, software need to decide the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This will make CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin normal initialization process.
9. Reset the host and then can begin the normal process.

27.4.6.2 Alternative fast boot flow

1. Software need to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software need to configure MMC Boot Register (offset 0xc4) bit 6 to 1(enable boot), and bit 5 to 1(alternative boot), and bit 4 to select the ack mode or not. If need to send through DMA mode, need to configure bit 7 to enable automatically stop at

block gap feature. And need to configure bit 3-bit0 to select the ack timeout value according to the sd clk frequence.

3. Software then need to configure Block Attributes Register to set block size/no. If in ddr fast boot mode, block size only can be configure to 512byte.
4. Software need to configure Protocol control register to set DTW(data transfer width). If in ddr fast boot mode, DTW only can be configure to 4-bit/8-bit dataline mode.
5. Software need to configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software need to configure Transfer Type Register to start the boot process by CMD0 with 0xFFFFFFFA argument . In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, MSBSEL is set to 1.

NOTE

DMAEN should be configured as 0 in polling mode. And if BCEN is configured as 1 in polling mode, better to configure blk no in Bock Attributes Register to the max value. And if in ddr fast boot mode, DDR_EN need to be set to 1.

7. When the step 6 is configured, boot process will begin. Software need to poll the data buffer ready status to read the data from buffer in time. If boot time out(ack data time out in 50ms or data time out in 1s), host will send out the interrupt and software need to send CMD0 with reset and then configure boot enable bit to 0 to stop this process. After command completed, configure MMC Boot Register bit 6 to 0 to disable boot. After at least 8 clocks from command completed, card is ready for identification step.
8. If no time out, software need to decide when to stop the boot process, and send out the CMD0 with reset and then after command completed, configure MMC Boot Register bit 6 to stop the process. After 8 clocks from command completed, slave(card) is ready for identification step.
9. Reset the host and then can begin the normal process.

27.4.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the

beginning of the image, need to switch DMA parameters on the fly during MMC fast boot.

In fast boot, host can use ADMA2 (Advanced DMA2) with two destinations.

The detail flow:

1. Software need to configure init_active bit (system control register bit 27) to make sure 74 card clocks are finished.
2. Software need to configure MMC Boot Register (offset 0xc4) bit 6 to 1(enable boot); and bit 5 to 0(normal fast boot), to 1(alternative boot); and bit 4 to select the acknowledge mode or not. In DMA mode, configure bit 7 to 1 for enable automatically stop at block gap feature. Also configure bit31-bit16 to set the VALUE1 (value of block count that need to transfer first time), that host will stop at block gap when card block counter is equal to this value. And need to configure bit 3-bit0 to select the acknowledge timeout value according to the SD clock frequency.
3. Software then need to configure Block Attributes Register to set block size/no. If in DDR fast boot mode, block size only can be configure to 512byte. In DMA mode, it is better to set block number to the max value (16'hFFFF).
4. Software need to configure Protocol control register to set DTW (data transfer width). If in DDR fast boot mode, DTW can be only configured to 4-bit/8-bit data line mode.
5. Software enbables ADMA2 by configuring protocol control register bit9-bit8.
6. Software need to set at least three pairs ADMA2 descriptor in boot memory (i.e. in IRAM, at least 6 words). The first pair descriptor define the start address (i.e. IRAM) and data length (i.e. 512 bytes*VALUE1) of first part boot code. Software also need to set the second pair descriptor, the second start address (any value that is writeable), data length is suggest to set 1~2word (record as VAULE2).

NOTE

The second couple descriptor also transfer useful data even at lease 1 word. Because our ADMA2 can't support 0 data_length data transfer descriptor.

7. Software need to configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot, and don't need set in normal fast boot.
8. Software need to configure Transfer Type Register to start the boot process. CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN and DMAEN are kept default value. DPSEL bit is set to 1, DTDSEL is set to 1, and MSBSEL is set to 1. DMAEN is configured as 1 in DMA mode. And if BCEN is configured as 1, better to configure block no in Bock Attributes Register to the max value. And if in DDR fast boot mode, DDR_EN need to be set to 1.
9. When the step 8 is configured, boot process will begin. The first VAULE1 block number data has transfer. Software need to polling TC bit (bit1 in Interrupt Status Register) to determine first transfer is end. Also software need to polling BGE bit (bit2 in Interrupt Status Register) to determine if first transfer stop at block gap.
10. When TC, BGE bit is 1, SW can analyzes the first code of VAULE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define

the start address and length of the remaining part of boot code(VAULE3 the remain boot code block). Remember set the last descriptor with END.

11. Software need to configure MMC Boot Register (offset 0xc4) again. Set bit 6 to 1(enable boot); and bit 5 to 0(normal fast boot), to 1(alternative boot); and bit 4 to select the acknowledge mode or not. In DMA mode, configure bit 7 to 1 for enable automatically stop at block gap feature. Also configure bit31-bit16 to set the (VAULE1+1+VAULE3), that host will stop at block gap when card block counter is equal to this value. And need to configure bit 3-bit0 to select the acknowledge timeout value according to the SD clock frequency.
12. Software need to clear TC and BGE bit. And software need to clear SABGREQ (bit 16 in Protocol control register), and set CREQ (bit17 Protocol control register) to 1 to resume the data transfer. Host will transfer the VALUE2 and VAULE3 data to the destination that is set by descriptor.
13. Software need to polling BGE bit to determine if the fast boot is over.

NOTE

- When ADMA boot flow is started, for ESDHC, it is like a normal ADMA read operation.
- Need a few words length memory to keep descriptor.
- For the 1~2 word data in second descriptor setting, it is the useful data, so software need to deal the data due to the application case.

27.5 Commands for MMC/SD/SDIO

See table below for the list of commands for the MMC/SD/SDIO cards.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

1. broadcast commands (bc), no response.
2. broadcast commands with response (bcr), response from all cards simultaneously.
3. addressed (point-to-point) commands (ac), no data transfer on the DAT.
4. addressed (point-to-point) data transfer commands (adtc).

The Access Bits for the EXT_CSD Access Modes are shown in table below.

Table 27-2. Commands for MMC/SD/SDIO Cards

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.

Table continues on the next page...

Table 27-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.

Table continues on the next page...

Table 27-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.

Table continues on the next page...

Table 27-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather that a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57~59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 shall be used to read and write these registers.

Table continues on the next page...

Table 27-2. Commands for MMC/SD/SDIO Cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62~63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac		R1	SET_WR_BLK_ERASE_COUNT	-
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac		R1	SET_CLR_CARD_DETECT	-
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed MMC cards . The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 26-3.
4. ACMDs shall be preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this block).

Table 27-3. EXT_CSD Access Modes

Bits	Access Name	Operation
00	Command Set	The command set is changed according to the Cmd Set field of the argument
01	Set Bits	The bits in the pointed byte are set, according to the 1 bits in the Value field.
10	Clear Bits	The bits in the pointed byte are cleared, according to the 1 bits in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

27.6 Software Restrictions

27.6.1 Initialization Active

The driver cannot set INITA bit in System Control register when any of the command line or data lines is active, so the driver must ensure both CDIHB and CIHB bits are cleared. In order to auto clear the INITA bit, the SDCLKEN bit must be '1', otherwise no clocks can go out to the card and INITA will never clear.

27.6.2 Softeware Polling Procedure

For polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not the times of the value in Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block. For example, for read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

27.6.3 Suspend Operation

In order to suspend the data transfer, the software must inform ESDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform ESDHC that the transfer is suspended.

If software needs to resume the suspended trasnfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending a 'suspend' command, ESDHC will regard the current transfer as aborted and change BLKCNT register to its original value, rather than keeping the remaining number of blocks.

27.6.4 Data Length Setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

27.6.5 (A)DMA Address Setting

To configure ADMA1/ADMA2/DMA address register, when TC bit is set, the register will always update itself with the internal address value to support dynamic address synchronization, so software must make sure TC bit is cleared prior to configuring ADMA1/ADMA2/DMA address register.

27.6.6 Data Port Access

Data Port does not support parallel access. For example, during an external DMA access, it is not allowed to write any data to the Data Port by ARM platform; or during a ARM read operation, it is also prohibited to write any data to the Data Port, by either ARM or external DMA. Otherwise the data would be corrupted inside the ESDHC buffer.

27.6.7 Change Clock Frequency

ESDHC does not automatically gates off the card clock when the Host Driver changes the clock frequency. To remove possible glitch on the card clock, clear SDCLKEN bit when changing clock divisor value and set SDCLKEN bit to '1' after SDSTB bit is '1' again.

27.6.8 Multi-block Read

For pre-defined multi-block read operation, that is, the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by ESDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode. In this case, the card may not respond to this extra abort command and ESDHC will get Response Timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

27.7 Programmable Registers

This section includes the block memory map and detailed descriptions of all registers. Each of these registers support only 32-bit accesses.

NOTE

Addresses greater than 0x44, except 0x50, 0x54, 0x58, 0x60, 0x64, 0xC0, 0xC4 and 0xFC, are reserved and read as all 0s. Write to these

ESDHCV3 memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5002_0000	DMA System Address (ESDHCV3-3_DSADDR)	32	R/W	0000_0000h	27.7.1/ 1412
5002_0004	Block Attributes (ESDHCV3-3_BLKATTR)	32	R/W	0000_0000h	27.7.2/ 1413
5002_0008	Command Argument (ESDHCV3-3_CMDARG)	32	R/W	0000_0000h	27.7.3/ 1414
5002_000C	Command Transfer Type (ESDHCV3-3_XFERTYP)	32	R/W	0000_0000h	27.7.4/ 1415
5002_0010	Command Response n (ESDHCV3-3_CMDRSP0)	32	R	0000_0000h	27.7.5/ 1419
5002_0014	Command Response n (ESDHCV3-3_CMDRSP1)	32	R	0000_0000h	27.7.5/ 1419
5002_0018	Command Response n (ESDHCV3-3_CMDRSP2)	32	R	0000_0000h	27.7.5/ 1419
5002_001C	Command Response n (ESDHCV3-3_CMDRSP3)	32	R	0000_0000h	27.7.5/ 1419
5002_0020	Data Buffer Access Port (ESDHCV3-3_DATPORT)	32	R/W	0000_0000h	27.7.6/ 1421
5002_0024	Present State (ESDHCV3-3_PRSSTAT)	32	R	0000_0000h	27.7.7/ 1421
5002_0028	Protocol Control (ESDHCV3-3_PROCTL)	32	R/W	0000_0000h	27.7.8/ 1426
5002_002C	System Control (ESDHCV3-3_SYSCTL)	32	R/W	0080_8008h	27.7.9/ 1430
5002_0030	Interrupt Status (ESDHCV3-3_IRQSTAT)	32	w1c	0000_0000h	27.7.10/ 1434

Table continues on the next page...

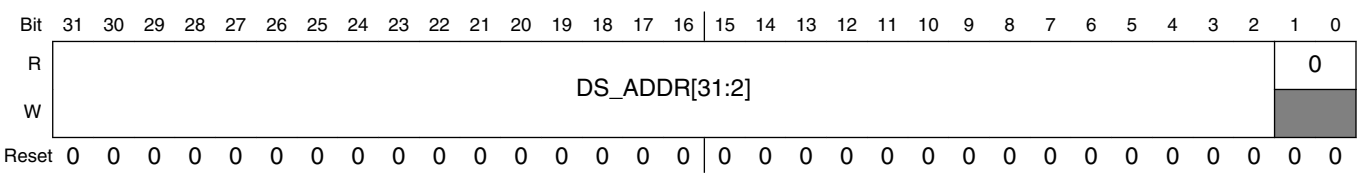
ESDHCV3 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5002_0034	Interrupt Status Enable (ESDHCV3-3_IRQSTATEN)	32	R/W	117F_013Fh	27.7.11/ 1439
5002_0038	Interrupt Signal Enable (ESDHCV3-3_IRQSIGEN)	32	R/W	0000_0000h	27.7.12/ 1442
5002_003C	Auto CMD12 Status (ESDHCV3-3_AUTOC12ERR)	32	R	0000_0000h	27.7.13/ 1444
5002_0040	Host Controller Capabilities (ESDHCV3-3_HOSTCAPBLT)	32	R	07F3_0000h	27.7.14/ 1447
5002_0044	Watermark Level (ESDHCV3-3_WML)	32	R/W	0810_0810h	27.7.15/ 1449
5002_0050	Force Event (ESDHCV3-3_FEVT)	32	W (always reads zero)	0000_0000h	27.7.16/ 1450
5002_0054	ADMA Error Status Register (ESDHCV3-3_ADMAES)	32	R	0000_0000h	27.7.17/ 1452
5002_0058	ADMA System Address (ESDHCV3-3_ADSADDR)	32	R/W	0000_0000h	27.7.18/ 1454
5002_0060	DLL (Delay Line) Control (ESDHCV3-3_DLLCTRL)	32	R/W	0000_0000h	27.7.19/ 1455
5002_0064	DLL Status (ESDHCV3-3_DLLSTS)	32	R	0000_0000h	27.7.20/ 1456
5002_00C0	Vendor Specific Register (ESDHCV3-3_VENDOR)	32	R/W	0000_0001h	27.7.21/ 1457
5002_00C4	MMC Boot Register (ESDHCV3-3_MMCBOOT)	32	R/W	0000_0000h	27.7.22/ 1458
5002_00FC	Host Controller Version (ESDHCV3-3_HOSTVER)	32	R	0000_1201h	27.7.23/ 1459

27.7.1 DMA System Address (ESDHCV3x_DSADDR)

This register contains the physical system memory address used for DMA transfers.

Addresses: ESDHCV3-3_DSADDR is 5002_0000h base + 0h offset = 5002_0000h



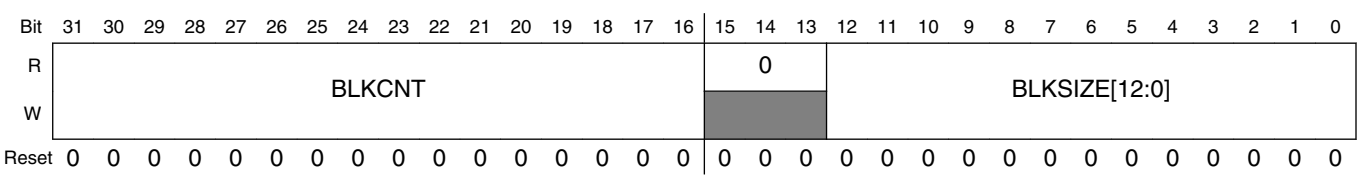
ESDHCV3x_DSADDR field descriptions

Field	Description
31–2 DS_ADDR[31:2]	DMA System Address: This register contains the 32-bit system memory address for a DMA transfer. As the address must be word (4 bytes) align, the least 2 bits are reserved, always 0. When the ESDHC stops a DMA transfer, this register points to the system address of the next contiguous data position. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operation during transfers may return an invalid value. The Host Driver shall initialize this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The Host driver shall wait, until ESDHCV3_PRSSTAT[DLA] bit is cleared, before writing to this register. The ESDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, ESDHC will automatically change SEQ burst type to NSEQ. As this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so SW cannot change this register when TC bit is set. Such restriction is also listed in Software Restrictions.
1–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

27.7.2 Block Attributes (ESDHCV3x_BLKATTR)

This register is used to configure the number of data blocks and the number of bytes in each block.

Addresses: ESDHCV3-3_BLKATTR is 5002_0000h base + 4h offset = 5002_0004h



ESDHCV3x_BLKATTR field descriptions

Field	Description
31–16 BLKCNT	Blocks Count For Current Transfer: This register is enabled when ESDHCV3_XFERTYP[BCEN] bit is set to 1 and is valid only for multiple block transfers. For single block transfer, this register will always read as 1. The Host Driver shall set this register to a value between 1 and the maximum block count. The ESDHC decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred. This register should be accessed only when no transaction is executing (that is, after transactions have stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored. When saving transfer content as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register. The reading of this register should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend.

Table continues on the next page...

ESDHCV3x_BLKATTR field descriptions (continued)

Field	Description
	This is because when Suspend command is sent out, ESDHC will regard the current transfer is aborted and change BLKCNT register back to its original value instead of keeping the dynamical indicator of remained block count. When restoring transfer content prior to issuing a Resume command, the Host Driver shall restore the previously saved block count. 0xFFFF 65535 blocks 0x0002 2 blocks 0x0001 1 blocks 0x0000 Stop Count
15–13 Reserved	This read-only field is reserved and always has the value zero. Reserved
12–0 BLKSIZE[12:0]	Transfer Block Size: This register specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations will be ignored. 0x1000 4096 Bytes 0x800 2048 Bytes 0x200 512 Bytes 0x1FF 511 Bytes 0x004 4 Bytes 0x003 3 Bytes 0x002 2 Bytes 0x001 1 Bytes 0x000 No data transfer

27.7.3 Command Argument (ESDHCV3x_CMDARG)

This register contains the SD/MMC Command Argument.

Addresses: ESDHCV3-3_CMDARG is 5002_0000h base + 8h offset = 5002_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMDARG[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV3x_CMDARG field descriptions

Field	Description
31–0 CMDARG[31:0]	Command Argument: The SD/MMC Command Argument is specified as bits 39-8 of the Command Format in the SD or MMC Specification.This register is write protected when the Command Inhibit (CMD) bit in the Present State register is set.

27.7.4 Command Transfer Type (ESDHCV3x_XFERTYP)

This register is used to control the operation of data transfers. The Host Driver sets this register before issuing a command followed by a data transfer, or before issuing a Resume command. To prevent data loss, the ESDHC prevents writing to the bits, that are involved in the data transfer of this register, when data transfer is active. These bits are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

The Host Driver shall check the Command Inhibit DAT bit (CDIHB) and the Command Inhibit CMD bit (CIHB) in the Present State register before writing to this register. When the CDIHB bit in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB bit is set, any write to this register is ignored.

On sending commands with data transfer involved, it is mandatory that the block size is non-zero. Besides, block count must also be non-zero, or indicated as single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise ESDHC will ignore the sending of this command and do nothing. For write command, with all above restrictions, it is also mandatory that the write protect switch is not active (WPSPL bit of Present State Register is '1), otherwise ESDHC will also ignore the command.

If the commands with data transfer does not receive the response in 64 clock cycles, that is, response time-out, ESDHC will regard the external device does not accept the command and abort the data transfer. In this scenario, the driver should issue the command again to re-try the transfer. It is also possible that for some reason the card responds the command but ESDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The CMDTYP field of this register is used to initiate three special commands; these are:

- Suspend: ESDHC does not monitor the content of the Suspend command response and therefore assumes that the commnand succeeds when issued. It then operates assuming the card bus has been released and that it is permissible to issue the next command using the DAT line. It is the responsibility of S/W to check the status of the Suspend command and send another command marked as Suspend to inform the ESDHC that a Suspend command was successfully issued. Refer to Suspend Resume for more details. After the end bit of command is sent, ESDHC de-asserts Read Wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the Suspend command fails, the ESDHC will maintain its

current state and the Host Driver shall restart the transfer by setting the Continue Request bit in the Protocol Control register.

- Resume: S/W re-starts the data transfer by restoring the registers saved before sending the Suspend Command and then sends the Resume Command. The ESDHC will check for a pending busy state before starting write transfers.
- Abort: If this command is set when executing a read transfer, ESDHC will stop reads to the buffer. If this command is set when executing a write transfer, ESDHC will stop driving the DAT line. After issuing the Abort command, the Host Driver should issue a software reset (Abort Transaction).

The following table shows the summary of how register settings determine the type of data transfer.

Table 27-49. Transfer Type Register Setting for Various Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

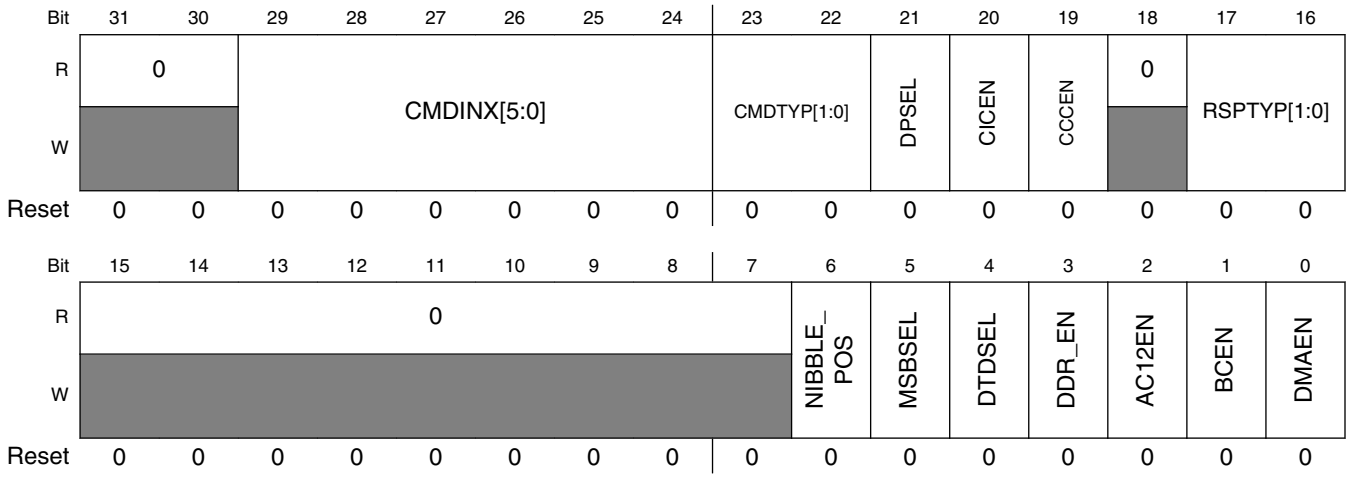
The following table shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, in regards to the Response Type bits and the name of the response type.

Table 27-50. Relationship Between Parameters and the Name of the Response Type

Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3,R4
10	1	1	R1,R5,R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification. But R5b is defined in this specification to indicate that ESDHC will check the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command will be used with R5b.
- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check shall be disabled for these response types.

Addresses: ESDHCV3-3_XFERTYP is 5002_0000h base + Ch offset = 5002_000Ch



ESDHCV3x_XFERTYP field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value zero. Reserved
29–24 CMDINX[5:0]	Command Index: These bits shall be set to the command number that is specified in bits 45-40 of the Command-Format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
23–22 CMDTYP[1:0]	Command Type: 11 Abort CMD12, CMD52 for writing I/O Abort in CCCR 10 Resume CMD52 for writing Function Select in CCCR 01 Suspend CMD52 for writing Bus Suspend in CCCR 00 Normal mode. Used for all other commands
21 DPSEL	Data Present Select: This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. It is set to 0 for the following: • Commands using only the CMD line (for example, CMD52). • Commands with no data transfer, but using the busy signal on DAT[0] line (R1b or R5b such as, CMD38) **NOTE:** In resume command, this bit shall be set, and other bits in this register shall be set the same as when the transfer was initially launched. When the Write Protect switch is on, (that is, the WPSPL bit is active as '0'), any command with a write operation will be ignored. That is to say, when this bit is set, while the DTDSEL bit is 0, writes to the register Transfer Type are ignored. 1 Data Present 0 No Data Present
20 CICEN	Command Index Check Enable: If this bit is set to 1, the ESDHC will check the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked.

Table continues on the next page...

ESDHCV3x_XFERTYP field descriptions (continued)

Field	Description
	1 Enable 0 Disable
19 CCCEN	Command CRC Check Enable: If this bit is set to 1, the ESDHC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. (Refer to RSPTYP[1:0] and table "Relationship Between Parameters and the Name of the Response Type ".) 1 Enable 0 Disable
18 Reserved	This read-only field is reserved and always has the value zero. Reserved
17–16 RSPTYP[1:0]	Response Type Select: 00 No Response 01 Response Length 136 10 Response Length 48 11 Response Length 48, check Busy after response
15–7 Reserved	This read-only field is reserved and always has the value zero. Reserved
6 NIBBLE_POS	In DDR 4 bit mode nibble position indication. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.
5 MSBSEL	Multi / Single Block Select: This bit enables multiple block DAT line data transfers. For any other commands, this bit shall be set to 0. If this bit is 0, it is not necessary to set the Block Count register. (See table above.) 1 Multiple Blocks 0 Single Block
4 DTDSEL	Data Transfer Direction Select: This bit defines the direction of DAT line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the ESDHC and is set to 0 for all other commands. 1 Read (Card to Host) 0 Write (Host to Card)
3 DDR_EN	Dual Data Rate mode selection
2 AC12EN	Auto CMD12 Enable: Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the ESDHC will issue a CMD12 automatically when the last block transfer has completed. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, the ESDHC will ignore this bit no matter if it is set or not. 1 Enable 0 Disable

Table continues on the next page...

ESDHCV3x_XFERTYP field descriptions (continued)

Field	Description
1 BCEN	Block Count Enable: This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 1 Enable 0 Disable
0 DMAEN	DMA Enable: This bit enables DMA functionality. If this bit is set to 1, a DMA operation shall begin when the Host Driver sets the DPSEL bit of this register. Whether the Simple DMA, or the Advanced DMA, is active depends on the DMA Select field of the Protocol Control register. 1 Enable 0 Disable

27.7.5 Command Response n (ESDHCV3x_CMDRSP*n*)

This register is used to store part n of the response bits from the card.

The table below describes the mapping of command responses from the SD Bus to Command Response registers for each response type. In the table, R[] refers to a bit range within the response data as transmitted on the SD Bus.

Table 27-52. Response Bit Definition for Each Response Type

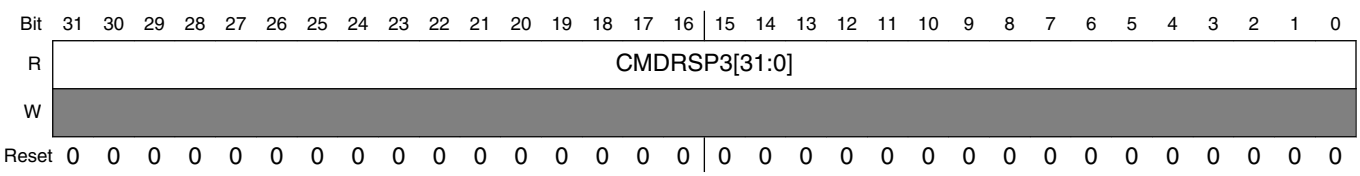
Response Type	Meaning of Response	Response Field	Response Register
R1,R1b (normal response)	Card Status	R[39:8]	ESDHCV3_CMDRSP0
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	ESDHCV3_CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{ESDHCV3_CMDRSP3[23:0] , ESDHCV3_CMDRSP2, ESDHCV3_CMDRSP1, ESDHCV3_CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	ESDHCV3_CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	ESDHCV3_CMDRSP0
R5, R5b	SDIO response	R[39:8]	ESDHCV3_CMDRSP0
R6 (Publish RCA)	New Published RCA[31:16] and card status[15:0]	R[39:9]	ESDHCV3_CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the ESDHCV3_CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the ESDHCV3_CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the ESDHCV3_CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, the ESDHC only stores part of the response data in the Command Response registers. This enables the Host Driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by the ESDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the ESDHC will check R[47:1], and if the response length is 136 the ESDHC will check R[119:1].

As ESDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, the ESDHC stores the Auto CMD12 response in the ESDHCV3_CMDRSP3 register. The CMD_wo_DAT response is stored in ESDHCV3_CMDRSP0. This allows the ESDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the ESDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

Addresses: ESDHCV3-3_CMDRSP0 is 5002_0000h base + 10h offset = 5002_0010h
ESDHCV3-3_CMDRSP1 is 5002_0000h base + 14h offset = 5002_0014h
ESDHCV3-3_CMDRSP2 is 5002_0000h base + 18h offset = 5002_0018h
ESDHCV3-3_CMDRSP3 is 5002_0000h base + 1Ch offset = 5002_001Ch



ESDHCV3x_CMDRSP*n* field descriptions

Field	Description
31–0 CMDRSP3[31:0]	Command Response n: Refer to the table "Response Bit Definition for Each Response Type" above for the mapping of command responses from the SD Bus to this register for each response type.

27.7.6 Data Buffer Access Port (ESDHCV3x_DATPORT)

This is a 32-bit data port register used to access the internal buffer.

Addresses: ESDHCV3-3_DATPORT is 5002_0000h base + 20h offset = 5002_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATCONT[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV3x_DATPORT field descriptions

Field	Description
31–0 DATCONT[31:0]	Data Content: The Buffer Data Port register is for 32-bit data access by the ARM platform or the external DMA. When the internal DMA is enabled, any write to this register is ignored, and any read from this register will always yield 0s.

27.7.7 Present State (ESDHCV3x_PRSTAT)

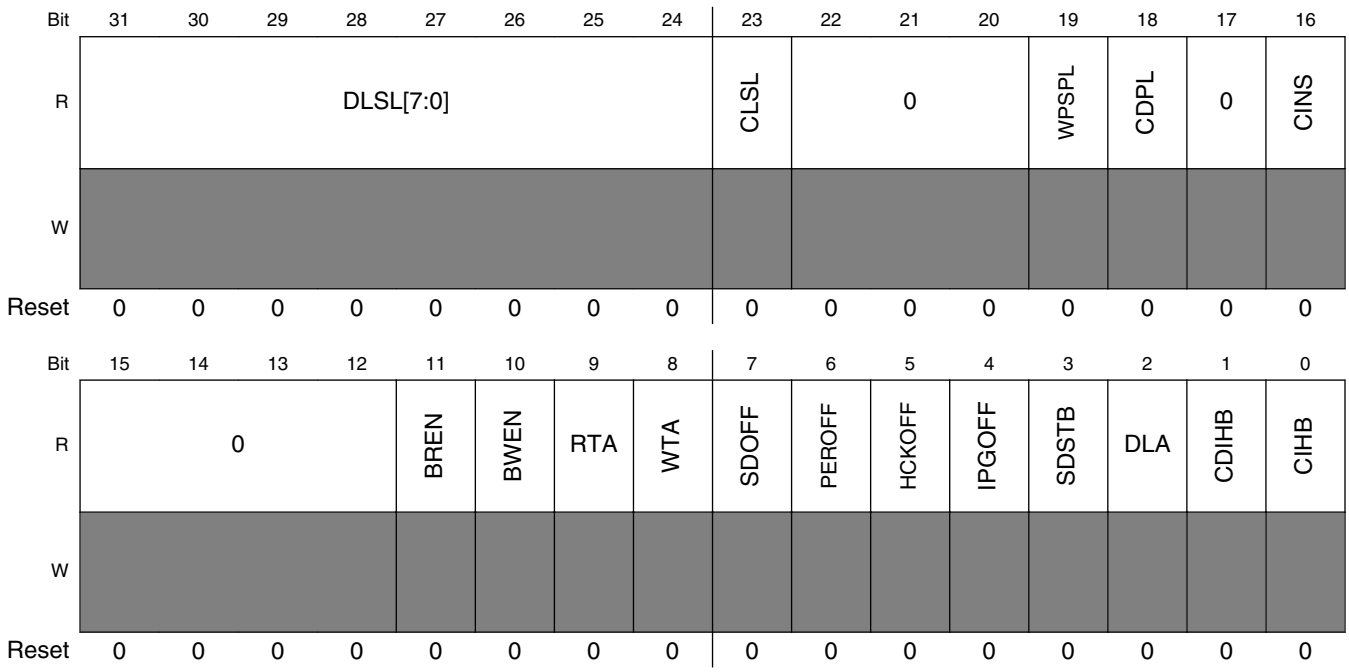
The Host Driver can get status of the ESDHC from this 32-bit read only register.

NOTE

The reset value of Present State Register depend on testbench connectivity.

- The Host Driver issues CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DAT lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands will be issued when Command Inhibit (DAT) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

Addresses: ESDHCV3-3_PRSSTAT is 5002_0000h base + 24h offset = 5002_0024h



ESDHCV3*x*_PRSSTAT field descriptions

Field	Description
31–24 DLSL[7:0]	DAT[7:0] Line Signal Level: This status is used to check the DAT line level to recover from errors, and for debugging.This is especially useful in detecting the busy signal level from DAT[0]. The reset value is effected by the external pull-up/pull-down resistors. By default, the read value of this bit field after reset is 8'b11110111, when DAT[3] is pulled down and the other lines are pulled up. DAT[7]: Data 7 line signal level DAT[6]: Data 6 line signal level DAT[5]: Data 5 line signal level DAT[4]: Data 4 line signal level DAT[3]: Data 3 line signal level DAT[2]: Data 2 line signal level DAT[1]: Data 1 line signal level DAT[0]: Data 0 line signal level
23 CLSL	CMD Line Signal Level: This status is used to check the CMD line level to recover from errors, and for debugging. The reset value is effected by the external pull-up/pull-down resistor, by default, the read value of this bit after reset is 1'b1, when the command line is pulled up.
22–20 Reserved	This read-only field is reserved and always has the value zero. Reserved
19 WPSPL	Write Protect Switch Pin Level:

Table continues on the next page...

ESDHCV3x_PRSSTAT field descriptions (continued)

Field	Description
	The Write Protect Switch is supported for memory and combo cards.This bit reflects the inverted value of the SD_WP pin of the card socket. A software reset does not affect this bit. The reset value is effected by the external write protect switch. If the SD_WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled. 1 Write enabled (SD_WP=0) 0 Write protected (SD_WP=1)
18 CDPL	Card Detect Pin Level: This bit reflects the inverse value of the SD_CD# pin for the card socket. Debouncing is not performed on this bit. This bit may be valid, but is not guaranteed, because of propagation delay. Use of this bit is limited to testing As it must be debounced by software. A software reset does not effect this bit. A write to the Force Event Register does not effect this bit. The reset value is effected by the external card detection pin. This bit shows the value on the SD_CD# pin (that is, when a card is inserted in the socket, it is 0 on the SD_CD# input, and consequently the CDPL reads 1.) 1 Card present (SD_CD#=0) 0 No card present (SD_CD#=1)
17 Reserved	This read-only field is reserved and always has the value zero. Reserved
16 CINS	Card Inserted: This bit indicates whether a card has been inserted. The ESDHC debounces this signal so that the Host Driver will not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not effect this bit. The Software Reset For All in the System Control register does not effect this bit.A software reset does not effect this bit. 1 Card Inserted 0 Power on Reset or No Card
15–12 Reserved	This read-only field is reserved and always has the value zero. Reserved
11 BREN	Buffer Read Enable: This status bit is used for non-DMA read transfers. The ESDHC may implement multiple buffers to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this bit is high, valid data greater than the watermark level exist in the buffer. A change of this bit from 1 to 0 occurs when any read from the buffer is made. A change of this bit from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 1 Read enable 0 Read disable
10 BWEN	Buffer Write Enable: This status bit is used for non-DMA write transfers. The ESDHC can implement multiple buffers to transfer data efficiently. This read only flag indicates if space is available for write data. If this bit is 1, valid data greater than the watermark level can be written to the buffer. A change of this bit from 1 to 0 occurs when any write to the buffer is made. A change of this bit from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled.

Table continues on the next page...

ESDHCV3x_PRSSTAT field descriptions (continued)

Field	Description
	1 Write enable 0 Write disable
9 RTA	Read Transfer Active: This status bit is used for detecting completion of a read transfer. This bit is set for either of the following conditions: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. A Transfer Complete interrupt is generated when this bit changes to 0. This bit is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, that is, all data are read away from ESDHC internal buffer. • When all valid data blocks have been transferred from ESDHC internal buffer to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1. 1 Transferring data 0 No valid data
8 WTA	Write Transfer Active: This status bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the ESDHC. This bit is set in either of the following cases: • After the end bit of the write command. • When writing 1 to the Continue Request bit in the Protocol Control register to restart a write transfer. This bit is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple). • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host Driver in determining when to issue commands during Write Busy state. 1 Transferring data 0 No valid data
7 SDOFF	SD Clock Gated Off Internally: This status bit indicates that the SD Clock is internally gated off, because of buffer over/under-run or read pause without read wait assertion, or the driver has cleared SDCLKEN bit to stop the SD clock. This bit is for the Host Driver to debug data transaction on the SD bus. 1 SD Clock is gated off 0 SD Clock is active
6 PEROFF	ipg_perclk Gated Off Internally:

Table continues on the next page...

ESDHCV3x_PRSSTAT field descriptions (continued)

Field	Description
	This status bit indicates that the ipg_perclk is internally gated off. This bit is for the Host Driver to debug transaction on the SD bus. When INITA bit is set, ESDHC sending 80 clock cycles to the card, the SDCLKEN bit must be '1' to enable the output card clock, otherwise the ipg_perclk will never be gate off, so ipg_perclk and ipg_clk will be always active. 1 ipg_perclk is gated off 0 ipg_perclk is active
5 HCKOFF	hclk Gated Off Internally: This status bit indicates that the hclk is internally gated off. This bit is for the Host Driver to debug during a data transfer. 1 hclk is gated off 0 hclk is active
4 IPGOFF	ipg_clk Gated Off Internally: This status bit indicates that the ipg_clk is internally gated off. This bit is for the Host Driver to debug. 1 ipg_clk is gated off 0 ipg_clk is active
3 SDSTB	SD Clock Stable This status bit indicates that the internal card clock is stable. This bit is for the Host Driver to poll clock status when changing the clock frequency. It is recommended to clear SDCLKEN bit in System Control register to remove glitch on the card clock when the frequency is changing. 1 clock is stable 0 clock is changing frequency and not stable
2 DLA	Data Line Active This status bit indicates whether one of the DAT lines on the SD Bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD Bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This bit will be set in either of the following cases: • After the end bit of the read command. • When writing a 1 to the Continue Request bit in the Protocol Control register to restart a read transfer. This bit will be cleared in either of the following cases: (1) When the end bit of the last data block is sent from the SD Bus to the ESDHC. (2) When the Read Wait state is stopped by a Suspend command and the DAT2 line is released. The ESDHC will wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the ESDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspend / resume function. This bit will remain 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD Bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This bit will be set in either of the following cases:

Table continues on the next page...

ESDHCV3x_PRSSTAT field descriptions (continued)

Field	Description
	• After the end bit of the write command. • When writing to 1 to the Continue Request bit in the Protocol Control register to continue a write transfer. This bit will be cleared in either of the following cases: • When the SD card releases Write Busy of the last data block, the ESDHC will also detect if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the ESDHC will assume the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and as a result of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This bit will be cleared when the DAT0 line is released. 1 DAT Line Active 0 DAT Line Inactive
1 CDIHB	Command Inhibit (DAT): This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this bit is 0, it indicates that the ESDHC can issue the next SD/MMC Command. Commands with a busy signal belong to Command Inhibit (DAT) (for example, R1b, R5b type). Except in the case when the command busy is finished, changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. **NOTE:** The SD Host Driver can save registers for a suspend transaction after this bit has changed from 1 to 0. 1 Cannot issue command which uses the DAT line 0 Can issue command which uses the DAT line
0 CIHB	Command Inhibit (CMD): If this status bit is 0, it indicates that the CMD line is not in use and the ESDHC can issue a SD/MMC Command using the CMD line. This bit is set also immediately after the Transfer Type register is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DAT) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If the ESDHC cannot issue the command because of a command conflict error (Refer to Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this bit will remain 1 and the Command Complete is not set. The Status of issuing an Auto CMD12 does not show on this bit. 1 Cannot issue command 0 Can issue command using only CMD line

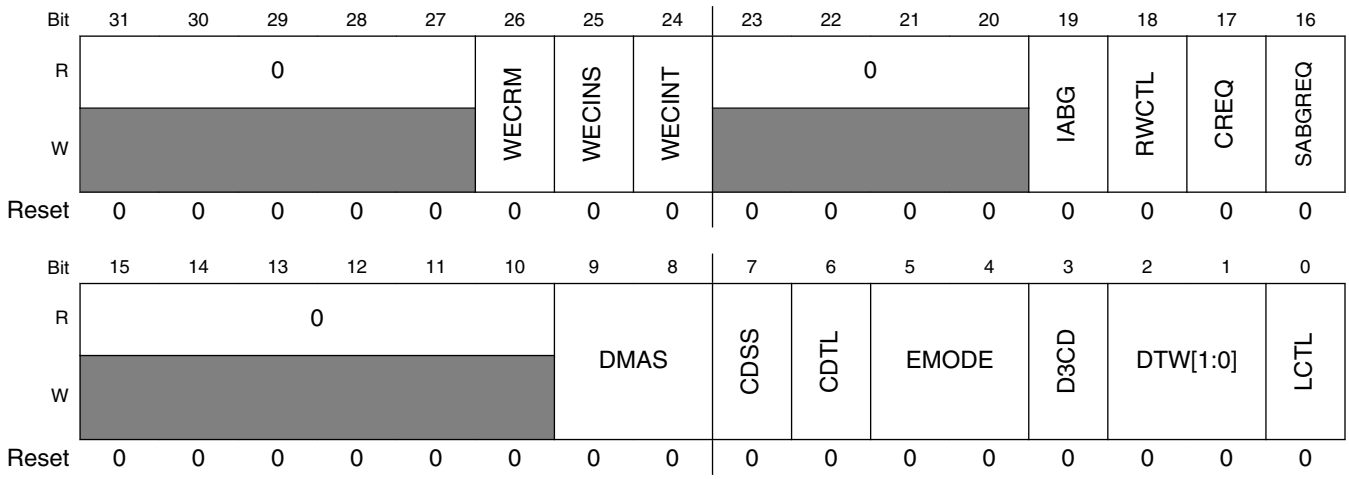
27.7.8 Protocol Control (ESDHCV3x_PROCTL)

There are three cases to restart the transfer after stop at the block gap. The appropriate case depends on whether ESDHC issues a Suspend command or the SD card accepts the Suspend command.

1. If the Host Driver does not issue a Suspend command, the Continue Request will be used to restart the transfer.
2. If the Host Driver issues a Suspend command and the SD card accepts it, a Resume command will be used to restart the transfer.
3. If the Host Driver issues a Suspend command and the SD card does not accept it, the Continue Request will be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the Host Driver will wait for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the Host Driver will clear the Stop At Block Gap Request before or simultaneously.

Addresses: ESDHCV3-3_PROCTL is 5002_0000h base + 28h offset = 5002_0028h



ESDHCV3x_PROCTL field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved
26 WECRM	Wakeup Event Enable On SD Card Removal: This bit enables a wakeup event, through a Card Removal, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Removal Status and the ESDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active in order to assert the Card Removal Status and the ESDHC interrupt. 1 Enable 0 Disable
25 WECINS	Wakeup Event Enable On SD Card Insertion: This bit enables a wakeup event, through a Card Insertion, in the Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not effect this bit. When this bit is set, the Card Insertion Status and the ESDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active in order to assert the Card Insertion Status and the ESDHC interrupt.

Table continues on the next page...

ESDHCV3x_PROCTL field descriptions (continued)

Field	Description
	1 Enable 0 Disable
24 WECINT	Wakeup Event Enable On Card Interrupt: This bit enables a wakeup event, through a Card Interrupt, in the Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. When this bit is set, the Card Interrupt Status and the ESDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active in order to assert the Card Interrupt Status and the ESDHC interrupt. 1 Enable 0 Disable
23–20 Reserved	This read-only field is reserved and always has the value zero. Reserved
19 IABG	Interrupt At Block Gap: This bit is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0 to avoid an inadvertent interrupt. When the Host Driver detects an SDIO card insertion, it sets this bit according to the CCCR of the card. 1 Enabled 0 Disabled
18 RWCTL	Read Wait Control: The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DAT[2] line. Otherwise the ESDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the Host Driver detects an SDIO card insertion, it sets this bit according to the CCCR of the card. If the card does not support read wait, this bit will never be set to 1, otherwise DAT line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the ESDHC will stop the SD Clock to pause reading operation. 1 Enable Read Wait Control, and assert Read Wait without stopping SD Clock at block gap when SABGREQ bit is set 0 Disable Read Wait Control, and stop SD Clock at block gap when SABGREQ bit is set
17 CREQ	Continue Request: This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. When a Suspend operation is not accepted by the card, it is also by setting this bit to restart the paused transfer. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit to 1 to restart the transfer. The ESDHC automatically clears this bit, therefore it is not necessary for the Host Driver to set this bit to 0. If both Stop At Block Gap Request and this bit are 1, the continue request is ignored. 1 Restart 0 No effect
16 SABGREQ	Stop At Block Gap Request: This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the Transfer Complete is set to 1, indicating a transfer completion, the Host Driver will leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request does not

Table continues on the next page...

ESDHCV3x_PROCTL field descriptions (continued)

Field	Description
	cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The ESDHC will honor the Stop At Block Gap Request for write transfers, but for read transfers it requires that the SDIO card support Read Wait. Therefore, the Host Driver will not set this bit during read transfers unless the SDIO card supports Read Wait and has set the Read Wait Control to 1, otherwise the ESDHC will stop the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the Host Driver writes data to the Data Port register, the Host Driver will set this bit after all block data is written. If this bit is set to 1, the Host Driver will not write data to the Data Port register after a block is sent. Once this bit is set, the Host Driver will not clear this bit before the Transfer Complete bit in Interrupt Status Register is set, otherwise the ESDHCs behavior is undefined. This bit effects Read Transfer Active, Write Transfer Active, DAT Line Active and Command Inhibit (DAT) in the Present State register. 1 Stop 0 Transfer
15–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9–8 DMAS	DMA Select: This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00 No DMA or Simple DMA is selected 01 ADMA1 is selected 10 ADMA2 is selected 11 reserved
7 CDSS	Card Detect Signal Selection: This bit selects the source for the card detection. 1 Card Detection Test Level is selected (for test purpose) 0 Card Detection Level is selected (for normal purpose)
6 CDTL	Card Detect Test Level: This is bit is enabled while the Card Detection Signal Selection is set to 1 and it indicates card insertion. 1 Card Detect Test Level is 1, card inserted 0 Card Detect Test Level is 0, no card inserted
5–4 EMODE	Endian Mode: The ESDHC supports all four endian modes in data transfer. Refer to Data Buffer for more details. 00 Big Endian Mode 01 Half Word Big Endian Mode 10 Little Endian Mode 11 Reserved
3 D3CD	DAT3 as Card Detection Pin: If this bit is set, DAT3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DAT3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 may set the card into the SPI mode, which the ESDHC does not support. In case of SDIO application, if SD_CD signal is not multiplexed to ESDHC block, S/W should set D3CD bit.

Table continues on the next page...

ESDHCV3x_PROCTL field descriptions (continued)

Field	Description
	1 DAT3 as Card Detection Pin 0 DAT3 does not monitor Card Insertion
2–1 DTW[1:0]	Data Transfer Width: This bit selects the data width of the SD bus for a data transfer. The Host Driver sets it to match the data width of the card. Possible Data transfer Width is 1-bit, 4-bits or 8-bits. 10 8-bit mode 01 4-bit mode 00 1-bit mode 11 Reserved
0 LCTL	LED Control: This bit, fully controlled by the Host Driver, is used to caution the user not to remove the card while the card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction. When the software issues multiple SD commands, setting the bit once before the first command is sufficient; it is not necessary to reset the bit between commands. 1 LED on 0 LED off

27.7.9 System Control (ESDHCV3x_SYSCTL)

Addresses: ESDHCV3-3_SYSCTL is 5002_0000h base + 2Ch offset = 5002_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				INITA	0	0	0	IPP_RST_N	0			DTOCV			
W						RSTD	RSTC	RSTA								
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SDCLKFS								DVS[3:0]				SDCLKEN	PEREN	HCKEN	IPGEN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

ESDHCV3x_SYSCTL field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

ESDHCV3x_SYSCTL field descriptions (continued)

Field	Description
27 INITA	Initialization Active: When this bit is set, 80 SD-Clocks are sent to the card. After the 80 clocks are sent, this bit is self cleared. This bit is very useful during the card power-up period when 74 SD-Clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the CIHB and CDIHB bits in the Present State Register are set, writing 1 to this bit is ignored (that is, when command line or data lines are active, write to this bit is not allowed). On the other hand, when this bit is set, that is, during initialization active period, it is allowed to issue commands, and the command bit stream will appear on the CMD pad after all 80 clock cycles are completer. When this command ends, the driver can make sure that the 80 clock cycles are sent out, which is very useful when the driver needs to send 80 cycles to the card and does not want to wait until this bit is self cleared.
26 RSTD	Software Reset For DAT Line: Only part of the data circuit is reset. DMA circuit is also reset. The following registers and bits are cleared by this bit: • Data Port register • Buffer is cleared and initialized.Present State register • Buffer Read Enable • Buffer Write Enable • Read Transfer Active • Write Transfer Active • DAT Line Active • Command Inhibit (DAT) Protocol Control register • Continue Request • Stop At Block Gap Request Interrupt Status register • Buffer Read Ready • Buffer Write Ready • DMA Interrupt • Block Gap Event • Transfer Complete 1 Reset 0 No Reset
25 RSTC	Software Reset For CMD Line: Only part of the command circuit is reset. The following registers and bits are cleared by this bit: • Present State register Command Inhibit (CMD) • Interrupt Status register Command Complete 1 Reset 0 No Reset
24 RSTA	Software Reset For ALL: This reset effects the entire Host Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During its initialization, the Host Driver will set this bit to 1 to reset the ESDHC. The ESDHC will reset this bit to 0 when the capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All does not affect the value of the Capabilities registers. After this bit is set, it is recommended that the Host Driver reset the external card and re-initialize it.

Table continues on the next page...

ESDHCV3x_SYSCTL field descriptions (continued)

Field	Description
	b Reset 0 No Reset
23 IPP_RST_N	If the card supports it, this Register value will be output directly to the CARD through the pad for hardware reset.
22–20 Reserved	This read-only field is reserved and always has the value zero. Reserved
19–16 DTOCV	Data Timeout Counter Value: This value determines the interval by which DAT line timeouts are detected. Refer to the Data Timeout Error bit in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock SDCLK value by this value. The Host Driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events. 1111 Reserved 1110 SDCLK x 2^{27} 0001 SDCLK x 2^{14} 0000 SDCLK x 2^{13} 1111 Reserved (in DDR mode) 1110 SDCLK x 2^{26} (in DDR mode) 0001 SDCLK x 2^{13} (in DDR mode) 0000 SDCLK x 2^{12} (in DDR mode)
15–8 SDCLKFS	SDCLK Frequency Select: This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this register holds the prescaler (this register) and divisor (next register) of the Base Clock Frequency register. Setting 00h bypasses the frequency prescaler of the SD Clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. **NOTE:** In ESDHC this register field can not be set to 00h(that is, bypass). The smallest value is 01h, that is divided by 2. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h will yield 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this bit field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD Clock after reset is 375 kHz. According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD Clock frequency is 50 MHz and will never exceed this limit. Only the following settings are allowed: 0x80 Base clock divided by 256 0x40 Base clock divided by 128 0x20 Base clock divided by 64

Table continues on the next page...

ESDHCV3x_SYSCTL field descriptions (continued)

Field	Description
	0x10 Base clock divided by 32 0x08 Base clock divided by 16 0x04 Base clock divided by 8 0x02 Base clock divided by 4 0x01 Base clock divided by 2
7–4 DVS[3:0]	Divisor: This register is used to provide a more exact divisor to generate the desired SD clock frequency. **NOTE:** The divider can even support odd divisor without deterioration of duty cycle. The setting are as following: 0x0 Divisor by 1 0x1 Divisor by 2 0xe Divisor by 15 0xf Divisor by 16
3 SDCLKEN	SD Clock Enable The Host Controller will stop SDCLK when writing this bit to 0. SDCLK Frequency can be changed when this bit is 0. Then, the Host Controller will maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK=0). If the Card Inserted in the Present State register is cleared, this bit should be cleared by the Host Driver to save power.
2 PEREN	Peripheral Clock Enable: If this bit is set, ipg_perclk will always be active and no automatic gating is applied. Thus the SDCLK is active except when auto gating-off during buffer danger (buffer about to over-run or under-run). When this bit is cleared, the ipg_perclk will be automatically off when there is no transaction on the SD bus. As this bit is only a feature enabling bit, clearing this bit does not stop SDCLK immediately. The ipg_perclk will be internally gated off, if none of the following factors are met: • The cmd part is reset, or • Data part is reset, or • A soft reset, or • The cmd is about to send, or • Clock divisor is just updated, or • Continue request is just set, or • This bit is set, or • Card insertion is detected, or • Card removal is detected, or • Card external interrupt is detected, or • 80 clocks for initialization phase is ongoing 1 ipg_perclk will not be automatically gated off 0 ipg_perclk will be internally gated off
1 HCKEN	HCLK Enable: If this bit is set, hclk will always be active and no automatic gating is applied. When this bit is cleared, hclk will be automatically off when no data transfer is on the SD bus. 1 hclk will not be automatically gated off 0 hclk will be internally gated off

Table continues on the next page...

ESDHCV3x_SYSCTL field descriptions (continued)

Field	Description
0 IPGEN	IPG Clock Enable: If this bit is set, ipg_clk will always be active and no automatic gating is applied. The ipg_clk will be internally gated off, if none of the following factors are met: • The cmd part is reset, or • Data part is reset, or • Soft reset, or • The cmd is about to send, or • Clock divisor is just updated, or • Continue request is just set, or • This bit is set, or • Card insertion is detected, or • Card removal is detected, or • Card external interrupt is detected, or • The ipg_perclk is not gated off **NOTE:** The ipg_clk will not be auto gated off if the ipg_perclk is not gated off. Clearing only this bit has no effect unless the PEREN bit is also cleared. 1 ipg_clk will not be automatically gated off 0 ipg_clk will be internally gated off

27.7.10 Interrupt Status (ESDHCV3x_IRQSTAT)

An interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. Before clearing the Card Interrupt, make sure that the card stops asserting the interrupt, that is, the Card Driver stops servicing the interrupt condition. Otherwise, the CINT bit will be asserted again.

The table below shows the relationship between the Command Timeout Error and the Command Complete.

Table 27-66. ESDHC Status for Command Timeout Error/Command Complete Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the Transfer Complete and the Data Timeout Error.

Table 27-67. ESDHC Status for Data Timeout Error/Transfer Complete Bit Combinations

Transfer Complete	Data Timeout Error	Meaning of the Status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data Transfer Complete

The table below shows the relationship between the Command CRC Error and Command Timeout Error.

Table 27-68. ESDHC Status for Command CRC Error/Command Timeout Error Bit Combinations

Command Complete	Command Timeout Error	Meaning of the Status
0	0	No error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Addresses: ESDHCV3-3_IRQSTAT is 5002_0000h base + 30h offset = 5002_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAE	0			AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				w1c				w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV3*x*_IRQSTAT field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 DMAE	DMA Error:

Table continues on the next page...

ESDHCV3x_IRQSTAT field descriptions (continued)

Field	Description
	Occurs when an Internal DMA transfer has failed. This bit is set to 1, when some error occurs in the data transfer. This error can be caused by either Simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. As any error corrupts the whole data block, the Host Driver will re-start the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 1 Error 0 No Error
27–25 Reserved	This read-only field is reserved and always has the value zero. Reserved
24 AC12E	Auto CMD12 Error: Occurs when detecting that one of the bits in the Auto CMD12 Error Status register has changed from 0 to 1. This bit is set to 1, not only when the errors in Auto CMD12 occur, but also when the Auto CMD12 is not executed due to the previous command error. 1 Error 0 No Error
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 DEBE	Data End Bit Error: Occurs either when detecting 0 at the end bit position of read data, which uses the DAT line, or at the end bit position of the CRC. 1 Error 0 No Error
21 DCE	Data CRC Error: Occurs when detecting a CRC error when transferring read data, which uses the DAT line, or when detecting the Write CRC status having a value other than 010. 1 Error 0 No Error
20 DTOE	Data Timeout Error: Occurs when detecting one of following time-out conditions. • Busy time-out for R1b,R5b type • Busy time-out after Write CRC status • Read Data time-out. 1 Time out 0 No Error
19 CIE	Command Index Error: Occurs if a Command Index error occurs in the command response. 1 Error 0 No Error
18 CEBE	Command End Bit Error: Occurs when detecting that the end bit of a command response is 0.

Table continues on the next page...

ESDHCV3x_IRQSTAT field descriptions (continued)

Field	Description
	1 End Bit Error Generated 0 No Error
17 CCE	Command CRC Error: Command CRC Error is generated in two cases. • If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this bit is set when detecting a CRC error in the command response. • The ESDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the ESDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then the ESDHC will abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error will also be set to 1 to distinguish CMD line conflict. 1 CRC Error Generated. 0 No Error
16 CTOE	Command Timeout Error: Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the ESDHC detects a CMD line conflict, in which case a Command CRC Error will also be set (as shown in the table above), this bit will be set without waiting for 64 SDCLK cycles. Because the command will be aborted by the ESDHC. 1 Time out 0 No Error
15–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 CINT	Card Interrupt: This status bit is set when an interrupt signal is detected from the external card. In 1-bit mode, the ESDHC will detect the Card Interrupt without the SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the Host System. Writing 1 to this bit will clear it, but as the interrupt factor from the SDIO card is not clear, this bit is set again. In order to clear this bit, reset the interrupt factor from the external card and then clear this bit. When this status bit is set, and the Host Driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be cleared to deactivate the interrupt signal to the Host System. After completion of the card interrupt service, (it should reset the interrupt factors in the SDIO card and the interrupt signal should not be asserted), write 1 to clear this bit, enable the Card Interrupt Signal Enable and start sampling the interrupt signal again. 1 Generate Card Interrupt 0 No Card Interrupt
7 CRM	Card Removal: This status bit is set if the Card Inserted bit (PRSSTNT[CINS]) changes from 1 to 0. When the Host Driver writes this bit to 1 to clear this status, the status of CINS should be confirmed, because the card state may possibly change when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if no card is inserted. In order to leave it cleared, clear the Card Removal Status Enable bit IRQSTATEN[CRMSEN]in Interrupt Status Enable register. 1 Card removed 0 Card state unstable or inserted

Table continues on the next page...

ESDHCV3x_IRQSTAT field descriptions (continued)

Field	Description
6 CINS	Card Insertion: This status bit is set if the Card Inserted bit in the Present State register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state may possibly be changed when the Host Driver clears this bit and the interrupt event may not be generated. When this bit is cleared, it will be set again if a card is inserted. In order to leave it cleared, clear the Card Inserted Status Enable bit in Interrupt Status Enable register. 1 Card inserted 0 Card state unstable or removed
5 BRR	Buffer Read Ready: This status bit is set if the Buffer Read Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Read Enable bit in the Present State register for additional information. 1 Ready to read buffer 0 Not ready to read buffer
4 BWR	Buffer Write Ready: This status bit is set if the Buffer Write Enable bit, in the Present State register, changes from 0 to 1. Refer to the Buffer Write Enable bit in the Present State register for additional information. 1 Ready to write buffer: 0 Not ready to write buffer
3 DINT	DMA Interrupt: Occurs only when the internal DMA finishes the data transfer successfully. When errors occur during data transfer, this bit will not be set. Instead, the DMAE bit will be set. Either Simple DMA or ADMA finishes data transferring, this bit will be set. 1 DMA Interrupt is generated 0 No DMA Interrupt
2 BGE	Block Gap Event: If the Stop At Block Gap Request bit in the Protocol Control register is set, this bit is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1. In the case of a Read Transaction: This bit is set at the falling edge of the DAT Line Active Status (When the transaction is stopped at SD Bus timing). The Read Wait must be supported in order to use this function. In the case of Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 1 Transaction stopped at block gap 0 No block gap event
1 TC	Transfer Complete: This bit is set when a read or write transfer is completed.

Table continues on the next page...

ESDHCV3x_IRQSTAT field descriptions (continued)

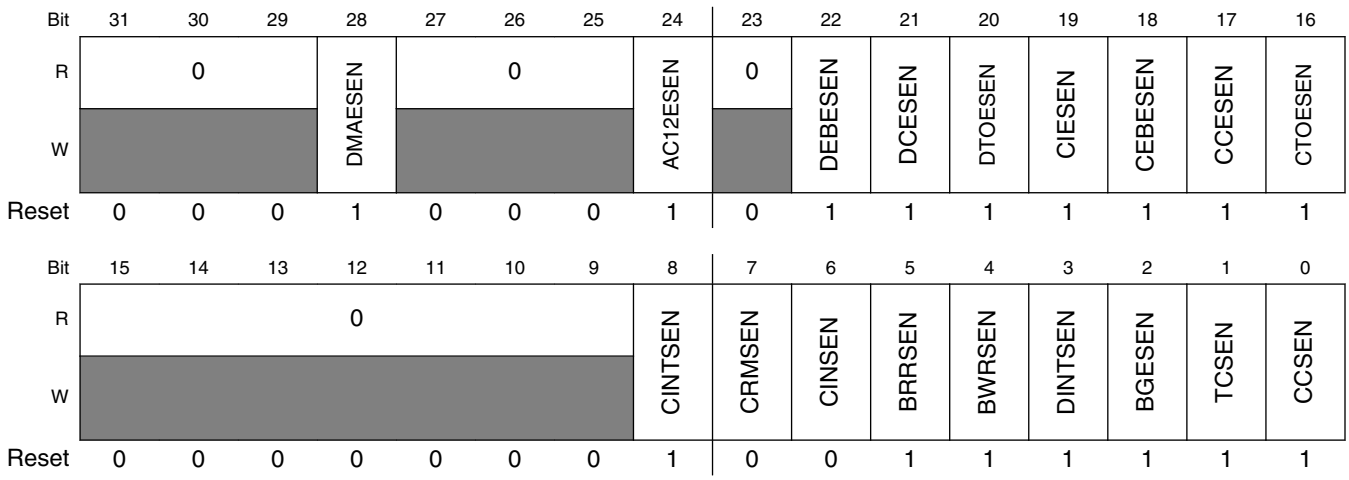
Field	Description
	In the case of a Read Transaction: This bit is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request bit in the Protocol Control register (after valid data has been read to the Host System). In the case of a Write Transaction: This bit is set at the falling edge of the DAT Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request bit in the Protocol Control register, and the data transfers are completed. (After valid data is written to the SD card and the busy signal released.) 1 Transfer complete 0 Transfer not complete
0 CC	Command Complete: This bit is set when you receive the end bit of the command response (except Auto CMD12). Refer to the Command Inhibit (CMD) in the Present State register. 1 Command complete 0 Command not complete

27.7.11 Interrupt Status Enable (ESDHCV3x_IRQSTATEN)

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event. If any bit is cleared, the corresponding Interrupt Status bit is also cleared (that is, when the bit in this register is cleared, the corresponding bit in Interrupt Status Register is always 0).

- Depending on IABG bit setting, ESDHC may be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There will be some delays on the Card Interrupt, asserted from the card, to the time the Host System is informed.
- To detect a CMD line conflict, the Host Driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

Addresses: ESDHCV3-3_IRQSTATEN is 5002_0000h base + 34h offset = 5002_0034h



ESDHCV3x_IRQSTATEN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 DMAESEN	DMA Error Status Enable: 1 Enabled 0 Masked
27–25 Reserved	This read-only field is reserved and always has the value zero. Reserved
24 AC12ESEN	Auto CMD12 Error Status Enable: 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 DEBESEN	Data End Bit Error Status Enable: 1 Enabled 0 Masked
21 DCESEN	Data CRC Error Status Enable: 1 Enabled 0 Masked
20 DTOESEN	Data Timeout Error Status Enable: 1 Enabled 0 Masked
19 CIESEN	Command Index Error Status Enable: 1 Enabled 0 Masked

Table continues on the next page...

ESDHCV3x_IRQSTATEN field descriptions (continued)

Field	Description
18 CEBESEN	Command End Bit Error Status Enable: 1 Enabled 0 Masked
17 CCESEN	Command CRC Error Status Enable: 1 Enabled 0 Masked
16 CTOESEN	Command Timeout Error Status Enable: 1 Enabled 0 Masked
15–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 CINTSEN	Card Interrupt Status Enable: If this bit is set to 0, the ESDHC will clear the interrupt request to the System. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 1 Enabled 0 Masked
7 CRMSEN	Card Removal Status Enable: 1 Enabled 0 Masked
6 CINSEN	Card Insertion Status Enable: 1 Enabled 0 Masked
5 BRRSEN	Buffer Read Ready Status Enable: 1 Enabled 0 Masked
4 BWRSEN	Buffer Write Ready Status Enable: 1 Enabled 0 Masked
3 DINTSEN	DMA Interrupt Status Enable: 1 Enabled 0 Masked
2 BGESEN	Block Gap Event Status Enable: 1 Enabled 0 Masked

Table continues on the next page...

ESDHCV3x_IRQSTATEN field descriptions (continued)

Field	Description
1 TCSEN	Transfer Complete Status Enable: 1 Enabled 0 Masked
0 CCSEN	Command Complete Status Enable: 1 Enabled 0 Masked

27.7.12 Interrupt Signal Enable (ESDHCV3x_IRQSIGEN)

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding Status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

Addresses: ESDHCV3-3_IRQSIGEN is 5002_0000h base + 38h offset = 5002_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DMAEIEN	0			AC12EIEN	0	DEBEIEN	DCEIEN	DTOEIEN	CIEIEN	CEBEIEN	CCEIEN	CTOEIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							CINTIEN	CRMIEN	CINIEN	BRRIEN	BWRIEN	DINTIEN	BGEIEN	TCIEN	CCIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV3x_IRQSIGEN field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 DMAEIEN	DMA Error Interrupt Enable: 1 Enable 0 Masked
27–25 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

ESDHCV3x_IRQSIGEN field descriptions (continued)

Field	Description
24 AC12EIEN	Auto CMD12 Error Interrupt Enable: 1 Enabled 0 Masked
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 DEBEIEN	Data End Bit Error Interrupt Enable: 1 Enabled 0 Masked
21 DCEIEN	Data CRC Error Interrupt Enable: 1 Enabled 0 Masked
20 DTOEIEN	Data Timeout Error Interrupt Enable: 1 Enabled 0 Masked
19 CIEIEN	Command Index Error Interrupt Enable: 1 Enabled 0 Masked
18 CEBEIEN	Command End Bit Error Interrupt Enable: 1 Enabled 0 Masked
17 CCEIEN	Command CRC Error Interrupt Enable: 1 Enabled 0 Masked
16 CTOEIEN	Command Timeout Error Interrupt Enable 1 Enabled 0 Masked
15–9 Reserved	This read-only field is reserved and always has the value zero. Reserved
8 CINTIEN	Card Interrupt Interrupt Enable: 1 Enabled 0 Masked
7 CRMIEN	Card Removal Interrupt Enable: 1 Enabled 0 Masked
6 CINIEN	Card Insertion Interrupt Enable:

Table continues on the next page...

ESDHCV3x_IRQSIGEN field descriptions (continued)

Field	Description
	1 Enabled 0 Masked
5 BRRIEN	Buffer Read Ready Interrupt Enable: 1 Enabled 0 Masked
4 BWRIEN	Buffer Write Ready Interrupt Enable: 1 Enabled 0 Masked
3 DINTIEN	DMA Interrupt Enable: 1 Enabled 0 Masked
2 BGEIEN	Block Gap Event Interrupt Enable: 1 Enabled 0 Masked
1 TCIEN	Transfer Complete Interrupt Enable: 1 Enabled 0 Masked
0 CCIEN	Command Complete Interrupt Enable: 1 Enabled 0 Masked

27.7.13 Auto CMD12 Status (ESDHCV3x_AUTOC12ERR)

When the Auto CMD12 Error Status bit in the Status register is set, the Host Driver will check this register to identify what kind of error the Auto CMD12 indicated. This register is valid only when the Auto CMD12 Error status bit is set.

The table below shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 27-72. Relationship Between Command CRC Error and Command Timeout Error for Auto CMD12

Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Type of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

1. When the ESDHC is going to issue an Auto CMD12.
   - Set bit 0 to 1 if the Auto CMD12 cannot be issued due to an error in the previous command
   - Set bit 0 to 0 if the Auto CMD12 is issued
2. At the end bit of an Auto CMD12 response.
   - Check errors correspond to bits 1-4.
   - Set bits 1-4 corresponding to detected errors.
   - Clear bits 1-4 corresponding to detected errors
3. Before reading the Auto CMD12 Error Status bit 7.
   - Set bit 7 to 1 if there is a command that cannot be issued
   - Clear bit 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, bit 7 will be sampled when the driver is not writing to the Command register. It is suggested to read this register only when the AC12E bit in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error bits (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

Addresses: ESDHCV3-3_AUTOC12ERR is 5002_0000h base + 3Ch offset = 5002_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								CNIBAC12E	0		AC12IE	AC12CE	AC12EBE	AC12TOE	AC12NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV3x_AUTOC12ERR field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 CNIBAC12E	Command Not Issued By Auto CMD12 Error: Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register. 1 Not Issued 0 No error
6–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4 AC12IE	Auto CMD12 Index Error: Occurs if the Command Index error occurs in response to a command. 1 Error, the CMD index in response is not CMD12 0 No error
3 AC12CE	Auto CMD12 CRC Error: Occurs when detecting a CRC error in the command response. 1 CRC Error Met in Auto CMD12 Response 0 No CRC error
2 AC12EBE	Auto CMD12 End Bit Error: Occurs when detecting that the end bit of command response is 0 which should be 1. 1 End Bit Error Generated 0 No error
1 AC12TOE	Auto CMD12 Timeout Error: Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to1, the other error status bits (2-4) have no meaning. 1 Time out 0 No error
0 AC12NE	Auto CMD12 Not Executed: If memory multiple block data transfer is not started, due to a command error, this bit is not set because it is not necessary to issue an Auto CMD12. Setting this bit to 1 means the ESDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this bit is set to 1, other error status bits (1-4) have no meaning. 1 Not executed 0 Executed

27.7.14 Host Controller Capabilities (ESDHCV3x_HOSTCAPBLT)

This register provides the Host Driver with information specific to the ESDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset. Any write to this register is ignored.

Addresses: ESDHCV3-3_HOSTCAPBLT is 5002_0000h base + 40h offset = 5002_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0					VS18	VS30	VS33	SRS	DMAS	HSS	ADMAS	-	MBL[2:0]		
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV3x_HOSTCAPBLT field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved
26 VS18	Voltage Support 1.8V: This bit depends on the Host System ability. 1 1.8V supported 0 1.8V not supported
25 VS30	Voltage Support 3.0V: This bit depends on the Host System ability. 1 3.0V supported 0 3.0V not supported
24 VS33	Voltage Support 3.3V: This bit depends on the Host System ability. 1 3.3V supported 0 3.3V not supported
23 SRS	Suspend / Resume Support:

Table continues on the next page...

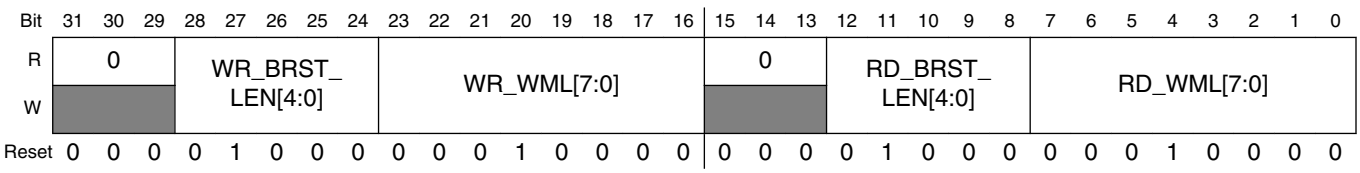
ESDHCV3x_HOSTCAPBLT field descriptions (continued)

Field	Description
	This bit indicates whether the ESDHC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism, as well as the Read Wait, are not supported, and the Host Driver will not issue either Suspend or Resume commands. 1 Supported 0 Not supported
22 DMAS	DMA Support: This bit indicates whether the ESDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly. 1 DMA Supported 0 DMA not supported
21 HSS	High Speed Support: This bit indicates whether the ESDHC supports High Speed mode and the Host System can supply a SD Clock frequency from 25 MHz to 50 MHz. 1 High Speed Supported 0 High Speed Not Supported
20 ADMAS	ADMA Support: This bit indicates whether the ESDHC supports the ADMA feature. 1 Advanced DMA Supported 0 Advanced DMA Not supported
19 -	Reserved
18–16 MBL[2:0]	Max Block Length: This value indicates the maximum block size that the Host Driver can read and write to the buffer in the ESDHC. The buffer will transfer block size without wait cycles. 000 512 bytes 001 1024 bytes 010 2048 bytes 011 4096 bytes
15–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

27.7.15 Watermark Level (ESDHCV3x_WML)

Both write and read watermark levels (FIFO threshold) are configurable. There value can range from 1 to 128 words. Both write and read burst lengths are also configurable. There value can range from 1 to 31 words.

Addresses: ESDHCV3-3_WML is 5002_0000h base + 44h offset = 5002_0044h



ESDHCV3x_WML field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28–24 WR_BRST_ LEN[4:0]	Write Burst Length: The number of words the ESDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (that is, it is not able to clear this field). **NOTE:** Due to a system restriction, the actual burst length may not exceed 16.
23–16 WR_WML[7:0]	Write Watermark Level: The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15–13 Reserved	This read-only field is reserved and always has the value zero. Reserved
12–8 RD_BRST_ LEN[4:0]	Read Burst Length: The number of words the ESDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer will be in back-to-back mode. On reset, this field will be 8. Writing 0 to this field will result in '01000' (that is, it is not able to clear this field. **NOTE:** Due to a system restriction, the actual burst length may not exceed 16.
7–0 RD_WML[7:0]	Read Watermark Level: The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

27.7.16 Force Event (ESDHCV3x_FEVT)

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Interrupt Status Register can be written if the corresponding bit of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register sets the corresponding bit of Interrupt Status Register. A read from this register always results in 0's. In order to change the corresponding status bits in the Interrupt Status Register, set IPGEN bit in System Control Register so that ipg_clk is always active.

Forcing a card interrupt will generate a short pulse on the DAT[1] line, and the driver will treat this interrupt as a normal interrupt. The interrupt service routine will skip polling the card interrupt factor as the interrupt is self cleared.

Addresses: ESDHCV3-3_FEVT is 5002_0000h base + 50h offset = 5002_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0		0	0			0	0	0	0	0	0	0	0	0
W	FEVTCINT			FEVTDMAE				FEVTAC12E		FEVTDEBE	FEVTDCE	FEVTDTOE	FEVTCIE	FEVTCEBE	FEVTCCE	FEVTCCE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0	0		0	0	0	0	0
W									FEVTCNIBAC12E			FEVTAC12IE	FEVTAC12EBE	FEVTAC12CE	FEVTAC12TOE	FEVTAC12NE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV3x_FEVT field descriptions

Field	Description
31 FEVTCINT	Force Event Card Interrupt: Writing 1 to this bit generates a short low-level pulse on the internal DAT[1] line, as if a self clearing interrupt was received from the external card. If enabled, the CINT bit will be set and the interrupt service routine will treat this interrupt as a normal interrupt from the external card.
30–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 FEVTDMAE	Force Event DMA Error: Forces the DMAE bit of Interrupt Status Register to be set
27–25 Reserved	This read-only field is reserved and always has the value zero. Reserved
24 FEVTAC12E	Force Event Auto Command 12 Error: Forces the AC12E bit of Interrupt Status Register to be set
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 FEVTDEBE	Force Event Data End Bit Error: Forces the DEBE bit of Interrupt Status Register to be set
21 FEVTDCE	Force Event Data CRC Error: Forces the DCE bit of Interrupt Status Register to be set
20 FEVTDTOE	Force Event Data Time Out Error: Force the DTOE bit of Interrupt Status Register to be set
19 FEVTCIE	Force Event Command Index Error: Forces the CCE bit of Interrupt Status Register to be set
18 FEVTCEBE	Force Event Command End Bit Error: Forces the CEBE bit of Interrupt Status Register to be set
17 FEVTCCE	Force Event Command CRC Error: Forces the CCE bit of Interrupt Status Register to be set
16 FEVTCCE	Force Event Command Time Out Error: Forces the CTOE bit of Interrupt Status Register to be set
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 FEVTCNIBAC12E	Force Event Command Not Executed By Auto Command 12 Error: Forces the CNIBAC12E bit in the Auto Command12 Error Status Register to be set
6–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4 FEVTAC12IE	Force Event Auto Command 12 Index Error: Forces the AC12IE bit in the Auto Command12 Error Status Register to be set
3 FEVTAC12EBE	Force Event Auto Command 12 End Bit Error: Forces the AC12EBE bit in the Auto Command12 Error Status Register to be set

Table continues on the next page...

ESDHCV3x_FEVT field descriptions (continued)

Field	Description
2 FEVTAC12CE	Force Event Auto Command 12 CRC Error: Forces the AC12CE bit in the Auto Command12 Error Status Register to be set
1 FEVTAC12TOE	Force Event Auto Command 12 Time Out Error: Forces the AC12TOE bit in the Auto Command12 Error Status Register to be set
0 FEVTAC12NE	Force Event Auto Command 12 Not Executed: Forces the AC12NE bit in the Auto Command12 Error Status Register to be set

27.7.17 ADMA Error Status Register (ESDHCV3x_ADMAES)

When an ADMA Error Interrupt occurrs, the ADMA Error States field (ADMAES) in this register holds the ADMA state and the ADMA System Address register (ADSADDR) holds the address of the error descriptor.

To recover from this error, the Host Driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address
- ST_CADR: This state is never set because it only increments the descriptor pointer and doesn't generate an ADMA error
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address

In case of a write operation, the Host Driver should use ACMD22 command (see below table) to get the number of the written block, because unwritten data may exist in the Host Controller.

The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The Host Driver can distinguish this error by reading the Valid bit of the error descriptor.

Table 27-77. ADMA Error State Coding

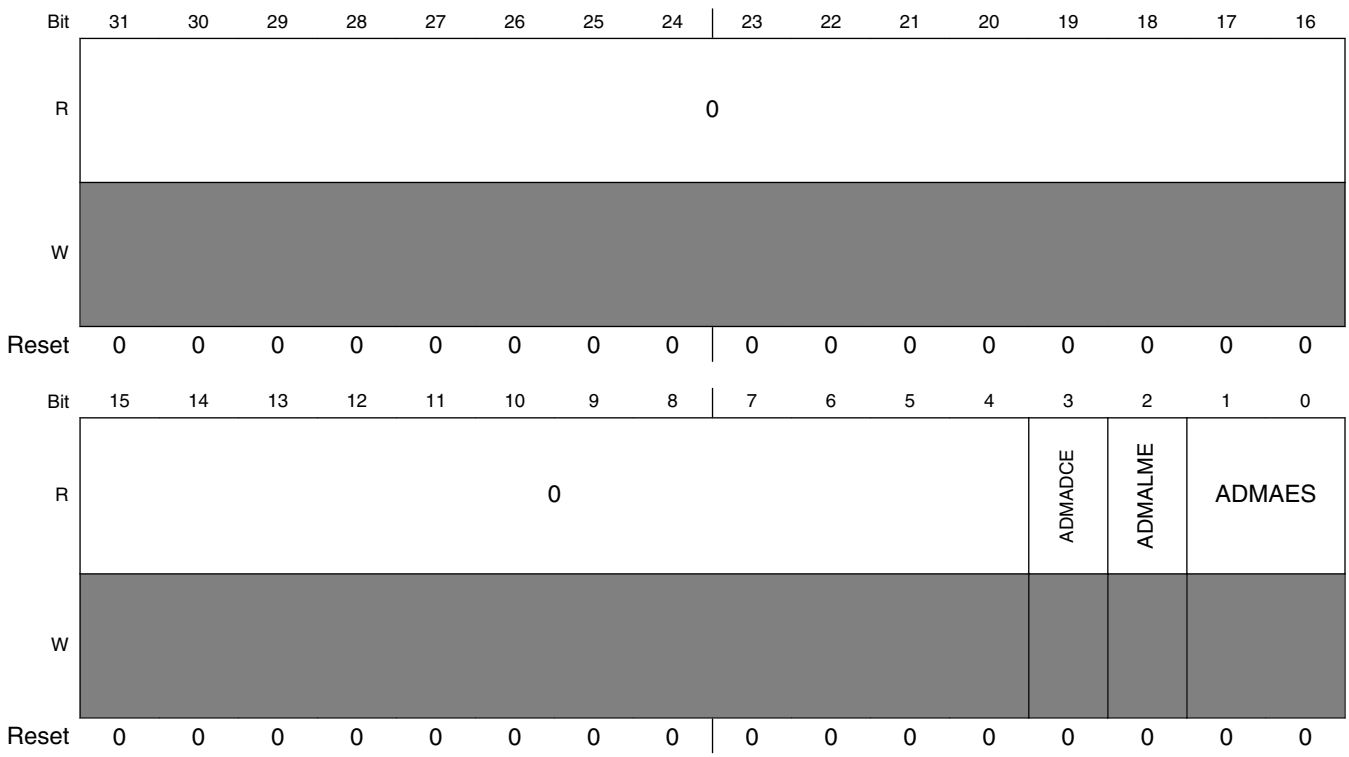
D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
00	ST_STOP (Stop DMA)	Holds the address of the next executable Descriptor command
01	ST_FDS (Fetch Descriptor)	Holds the valid Descriptor address

Table continues on the next page...

Table 27-77. ADMA Error State Coding (continued)

D01-D00	ADMA Error State (when error has occurred)	Contents of ADMA System Address Register
10	ST_CADR (Change Address)	No ADMA Error is generated
11	ST_TFR (Transfer Data)	Holds the address of the next executable Descriptor command

Addresses: ESDHCV3-3_ADMAES is 5002_0000h base + 54h offset = 5002_0054h



ESDHCV3x_ADMAES field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value zero. Reserved
3 ADMADCE	ADMA Descritor Error: This error occurs when invalid descriptor fetched by ADMA: 1 Error 0 No Error
2 ADMALME	ADMA Length Mismatch Error: This error occurs in the following 2 cases: • While the Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length • Total data length can not be divided by the block length

Table continues on the next page...

ESDHCV3x_ADMAES field descriptions (continued)

Field	Description
	1 Error 0 No Error
1–0 ADMAES	ADMA Error State (when ADMA Error is occurred.): This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. Refer to ADMA Error Status Register (ESDHCV3_ADMAES) for more details.

27.7.18 ADMA System Address (ESDHCV3x_ADSADDR)

This register contains the physical system memory address used for ADMA transfers.

Addresses: ESDHCV3-3_ADSADDR is 5002_0000h base + 58h offset = 5002_0058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADS_ADDR[31:0]																														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

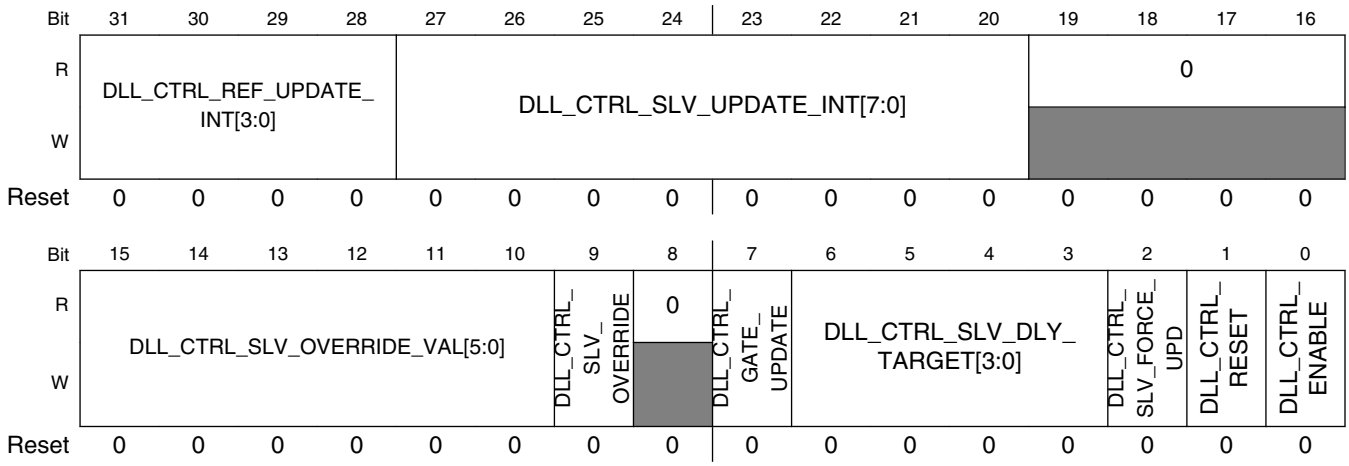
ESDHCV3x_ADSADDR field descriptions

Field	Description
31–2 ADS_ ADDR[31:0]	ADMA System Address: This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the Host Driver will set the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register will hold the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned. As this register supports dynamic address reflecting, when TC bit is set, it automatically alters the value of internal address counter, so this register value cannot be changed when the TC bit is set. Such restriction is also listed in on page 30-142.
1–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

27.7.19 DLL (Delay Line) Control (ESDHCV3x_DLLCTRL)

This register contains control bits for DLL.

Addresses: ESDHCV3-3_DLLCTRL is 5002_0000h base + 60h offset = 5002_0060h



ESDHCV3x_DLLCTRL field descriptions

Field	Description
31–28 DLL_CTRL_REF_UPDATE_INT[3:0]	DLL control loop update interval. The interval cycle is (2 + REF_UPDATE_INT) * ref_clock. By default, the DLL control loop will update every two ref_clock cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 DLL_CTRL_SLV_UPDATE_INT[7:0]	Slave delay line update interval. If default 0 is used, it means 256 cycles of ref_clock. A value of 0x0f results in 15 cycles and so on. **NOTE:** The software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. **NOTE:** The slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–10 DLL_CTRL_SLV_OVERRIDE_VAL[5:0]	When SLV_OVERRIDE=1 This field is used to select 1 of 64 physical taps manually. A value of 0 selects tap 1, and a value of 0x3f selects tap 64.
9 DLL_CTRL_SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to tbe enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0
8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 DLL_CTRL_GATE_UPDATE	Set this bit to 1 to force DLL to not update from now on. Beacue, when clock_in exists, glitches might appear during update. This bit is used by software if we met such kind of condition. Set it to 0 to allow DLL to update automatically

Table continues on the next page...

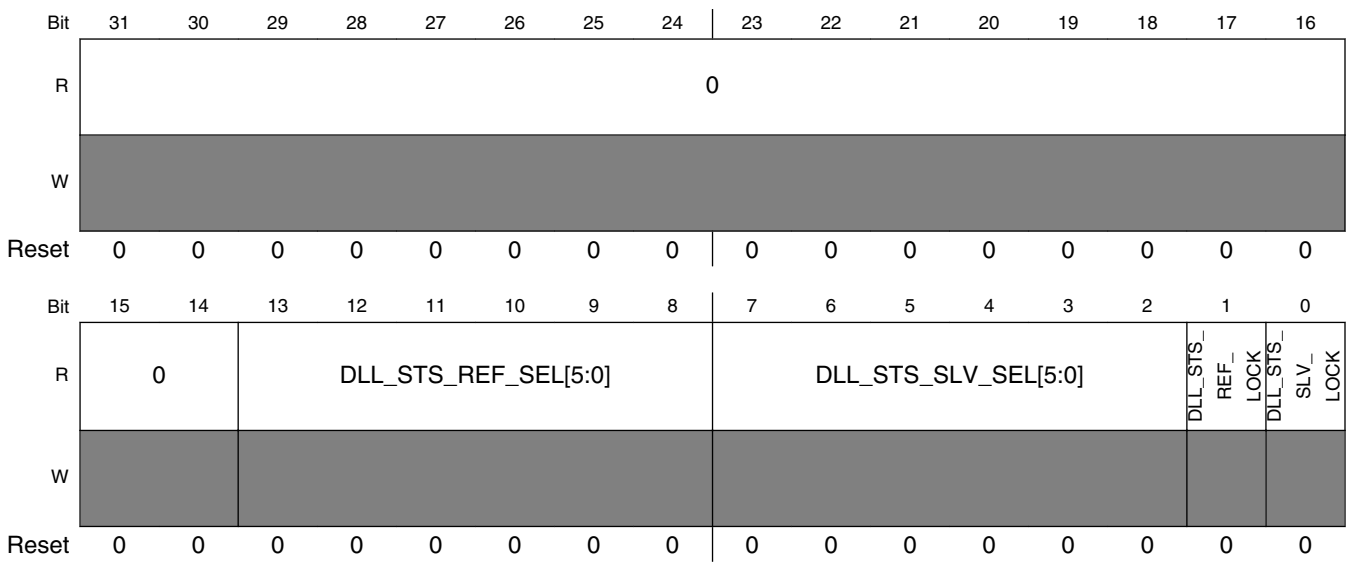
ESDHCV3x_DLLCTRL field descriptions (continued)

Field	Description
6–3 DLL_CTRL_ SLV_DLY_ TARGET[3:0]	The delay target for the ESDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. So the input read-clock can be delayed relative input data from (ref_clock/2)/16 to ref_clock/2
2 DLL_CTRL_ SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line will update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when ESDHC is idle. This function may not work when ESDHC is working on data/cmd/response.
1 DLL_CTRL_ RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an ref_clock half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again
0 DLL_CTRL_ ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. **NOTE:** Using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled

27.7.20 DLL Status (ESDHCV3x_DLLSTS)

This register contains the DLL status information. All bits are read only and will read the same as the power-reset value.

Addresses: ESDHCV3-3_DLLSTS is 5002_0000h base + 64h offset = 5002_0064h



ESDHCV3x_DLLSTS field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

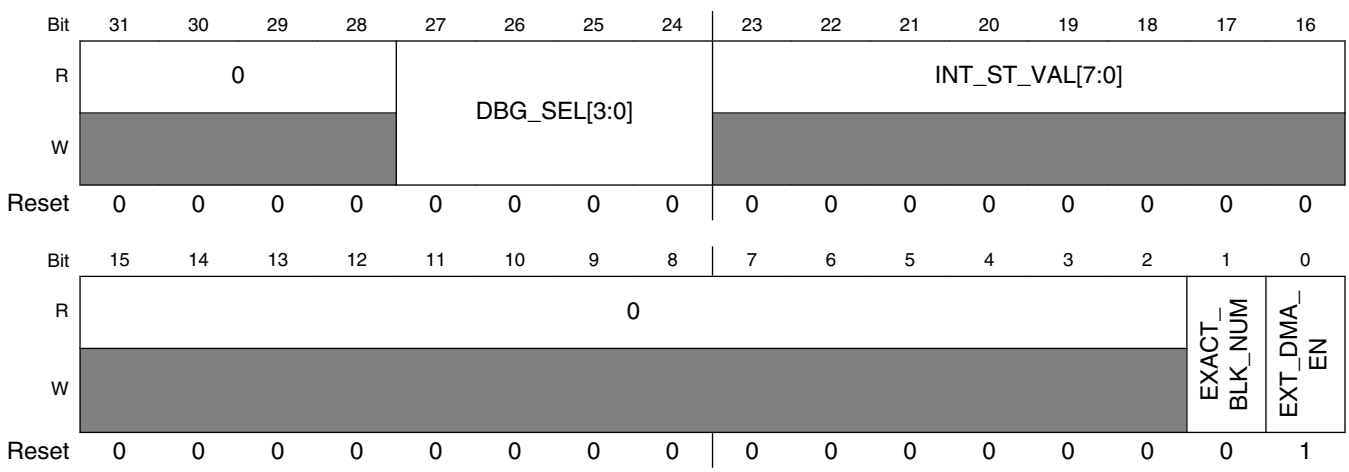
ESDHCV3x_DLLSTS field descriptions (continued)

Field	Description
13–8 DLL_STS_REF_SEL[5:0]	Reference delay line select taps. **NOTE:** this is encoded by 6 bits for 64 taps.
7–2 DLL_STS_SLV_SEL[5:0]	Slave delay line select status. This is the instant value generated from the reference chain. Because only when ref_lock is detected can the reference chain get updated, this value should be the right value to be updated next to the slave line when reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

27.7.21 Vendor Specific Register (ESDHCV3x_VENDOR)

This register contains the vendor specific control/status register.

Addresses: ESDHCV3-3_VENDOR is 5002_0000h base + C0h offset = 5002_00C0h



ESDHCV3x_VENDOR field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value zero. Reserved
27–24 DBG_SEL[3:0]	Debug Select Select the internal sub-block to show its internal state value.
23–16 INT_ST_VAL[7:0]	Internal State Value Internal state value, reflecting the corresponding state value selected by Debug Select field. This field is read-only and write to this field does not have effect.

Table continues on the next page...

ESDHCV3x_VENDOR field descriptions (continued)

Field	Description
15–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 EXACT_BLK_NUM	Exact block number block read enable for SDIO CMD53 This bit should be set before S/W issues CMD53 multi-block read with exact block number. This bit should not be set if the CMD53 multi-block read is not exact block number. 0 none exact block read 1 exact block read for SDIO CMD53
0 EXT_DMA_EN	External DMA Request Enable Enable the request to external DMA. When the internal DMA (either Simple DMA or Advanced DMA) is not in use, and this bit is set, ESDHC will send out DMA request when the internal buffer is ready. This bit is particularly useful when transferring data by ARM platform polling mode, and it is not allowed to send out the external DMA request. By default, this bit is set. 0 In any scenario, ESDHC does not send out external DMA request 1 When internal DMA is not active, the external DMA request will be sent out

27.7.22 MMC Boot Register (ESDHCV3x_MMCBOOT)

This register contains the MMC Fast Boot control register.

Addresses: ESDHCV3-3_MMCBOOT is 5002_0000h base + C4h offset = 5002_00C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BOOT_BLK_CNT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								AUTO_SABG_EN	BOOT_EN	MMC_BOOT_MODE	BOOT_ACK	DTOCV_ACK[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESDHCV3x_MMCBOOT field descriptions

Field	Description
31–16 BOOT_BLK_CNT[15:0]	The value defines the Stop At Block Gap value of automatic mode. When received card block cnt is equal to BOOT_BLK_CNT and AUTO_SABG_EN is 1, then Stop At Block Gap.
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

ESDHCV3x_MMCBOOT field descriptions (continued)

Field	Description
7 AUTO_SABG_EN	When boot, enable auto stop at block gap function. This function will be triggered, and host will stop at block gap when received card block cnt is equal to BOOT_BLK_CNT.
6 BOOT_EN	Boot mode enable 0: fast boot disable 1: fast boot enable
5 MMC_BOOT_MODE	Boot mode select 0: normal boot 1: alternative boot
4 BOOT_ACK	Boot ack mode select 0: no ack 1: ack
3–0 DTOCV_ACK[3:0]	Boot ACK time out counter value. 0000 SDCLK x 2^8 0001 SDCLK x 2^9 0010 SDCLK x 2^10 0011 SDCLK x 2^11 0100 SDCLK x 2^12 0101 SDCLK x 2^13 0110 SDCLK x 2^14 0111 SDCLK x 2^15 1110 SDCLK x 2^22 1111 Reserved 0000 SDCLK x 2^7 (when in DDR mode) 0001 SDCLK x 2^8 (when in DDR mode) 0010 SDCLK x 2^9 (when in DDR mode) 0011 SDCLK x 2^10 (when in DDR mode) 0100 SDCLK x 2^11 (when in DDR mode) 0101 SDCLK x 2^12 (when in DDR mode) 0110 SDCLK x 2^13 (when in DDR mode) 0111 SDCLK x 2^14 (when in DDR mode) 1110 SDCLK x 2^21 (when in DDR mode) 1111 Reserved (when in DDR mode)

27.7.23 Host Controller Version (ESDHCV3x_HOSTVER)

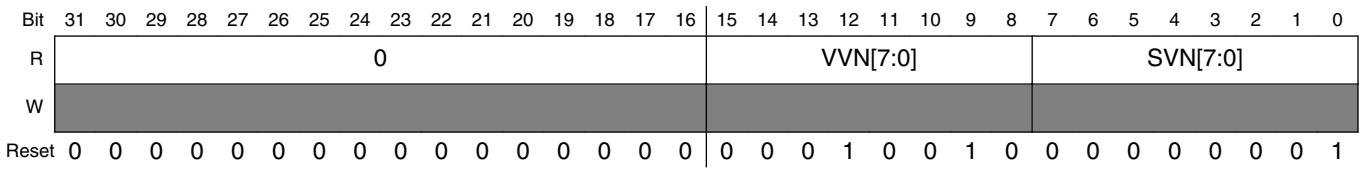
This register contains the vendor Host Controller version information. All bits are read only and will read the same as the power-reset value.

NOTE

The VVN field is 0x12 for both ESDHCV2 and ESDHCV3. Todetermine whether the controller is ESDHCV2 or ESDHCV3 do the following:

Set DLLCTRL [10]. This bit only exists in ESDHCV3. If it is set to 1 successfully, then the controller is ESDHCV3, otherwise it is ESDHCV2.

Addresses: ESDHCV3-3_HOSTVER is 5002_0000h base + FCh offset = 5002_00FCh



ESDHCV3x_HOSTVER field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–8 VVN[7:0]	Vendor Version Number: These status bits are reserved for the vendor version number. The Host Driver will not use this status. Settings not shown are reserved. 0x00 Freescale ESDHC Version 1.0 0x10 Freescale ESDHC Version 2.0 0x11 Freescale ESDHC Version 2.1 0x12 Freescale ESDHC Version 2.2
7–0 SVN[7:0]	Specification Version Number: These status bits indicate the Host Controller Specification Version. Settings not shown are reserved. — 0x01 SD Host Specification Version 2.0, supports Test Event Register and ADMA.

Chapter 28
Fast Ethernet Controller (FEC)

28.1 Overview

This chapter provides a feature-set overview, a functional block diagram, and transceiver connection information for the 10- and 100-Mbps reduced media independent interfaces (RMII) and the 7-wire serial interface. Detailed functional descriptions and application/ initialization information are included.

The Fast Ethernet controller (FEC) is designed to support both 10- and 100-Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports two different standard physical interfaces (MAC-PHY) for connection to an external Ethernet transceiver.

NOTE

Only RMII signals are available. The additional signals required for MII interface to the transceiver are not available.

The FEC is implemented with a combination of hardware and microcode. The following figure is a block diagram of the FEC.

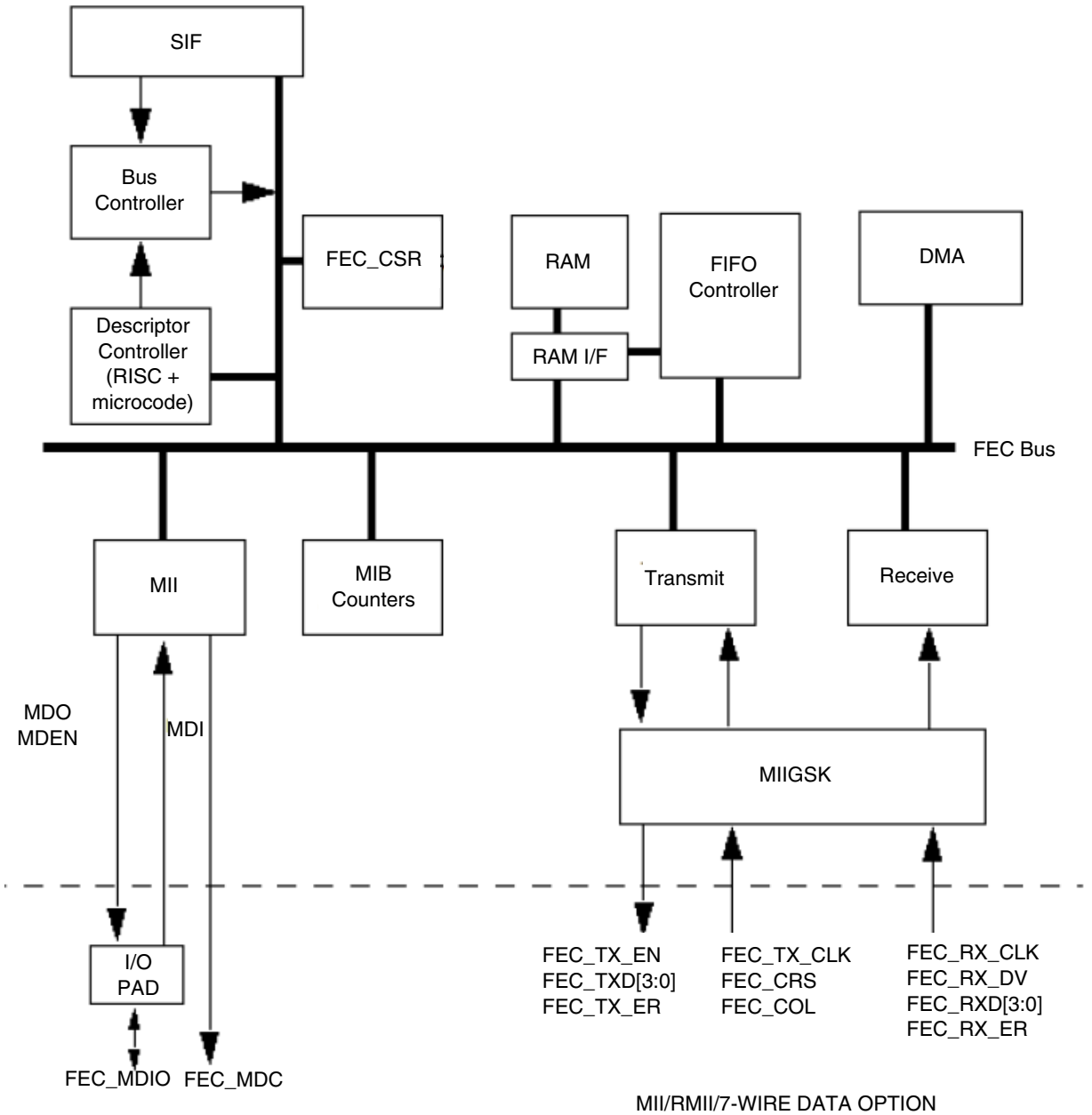


Figure 28-1. FEC Block Diagram

The following section provides information regarding the FEC submodules.

- The descriptor controller is a RISC-based controller that provides the following functions in the FEC:
  - Initialization (those internal registers not initialized by the user or hardware)
  - High-level control of the DMA channels (initiating DMA transfers)
  - Interpreting buffer descriptors
  - Address recognition for receive frames
  - Random number generation for transmit collision backoff timer

NOTE

This DMA engine is for the transfer of FEC data only and is not related to the system DMA controller.

- The RAM is the focal point of all data flow in the FEC and is divided into transmit and receive FIFOs. The FIFO boundaries are programmable using the FEC_FRSR register. User data flows to and from the DMA block, from and to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO to the transmit block, and receive data flows from the receive block to the receive FIFO.
- The user controls the FEC by writing, through the slave interface (SIF) submodule, to control registers located in each block. The control and status register (FEC_CSR) provides global control (for example, Ethernet reset and enable) and interrupt handling registers.
- The MII provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of the management data clock and management data input/output lines of the MII interface (FEC_MDC and FEC_MDIO, respectively).
- The DMA provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to run independently.
- The transmit and receive blocks provide the Ethernet MAC functionality (with some assistance from microcode).
- The message information block (MIB) maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC but provides valuable counters for network management. The counters supported are the RMON (RFC 1757) Ethernet statistics group and some of the IEEE 802.3 counters. See Message Information Block (MIB) Counters Memory Map, for more information.
- The MII gasket block (MIIGSK) provides an interface between the MII (Media Independent Interface specified by IEEE 802.3 standard) I/F and RMII (low pin count Reduced Media Independent Interface as specified by the RMII Consortium™ standard) I/F.

28.1.1 Features

The FEC incorporates the following features:

- Support for two different Ethernet physical interfaces:
  - 10-Mbps and 100-Mbps RMII
  - 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full-duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority

- Support for full-duplex operation (200Mbps throughput) with a minimum system clock rate of 50 MHz
- Support for half-duplex operation (100Mbps throughput) with a minimum system clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
  - Frames with broadcast address can be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode

28.2 Modes of Operation

The primary operational modes are described in this section.

28.2.1 Full- and Half-Duplex Operation

Full-duplex mode is intended for use on point to point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters. Selection of the duplex mode is controlled by the FEC_TCR[FDEN] bit. When configured for full-duplex mode, flow control can be enabled, which is effected by the FEC_TCR[RFC_PAUSE], FEC_TCR[TFC_PAUSE], and FEC_RCR[FCE] bits. See Full-Duplex Flow Control, for more details.

28.2.2 Interface Options

The following interface options are supported. A detailed discussion of the interface configurations is provided in Network Interface Options.

28.2.2.1 10-Mbps and 100-Mbps Media Independent Interface (MII)

The MII is defined by the IEEE 802.3 standard for 10/100-Mbps operation. The MAC-PHY interface can be configured to operate in MII mode by asserting FEC_RCR[MII_MODE].

The speed of operation is determined by the FEC_TX_CLK and FEC_RX_CLK signals which are driven by the external transceiver. The transceiver either autonegotiates the speed or control by software via the serial management interface (FEC_MDC/FEC_MDIO) signals to the transceiver. See the descriptions in MII management frame register (FEC_MMFR) and MII speed control register (FEC_MSCR) respectively, as well as MII documentation for a description of how to read and write registers in the transceiver through this interface.

28.2.2.2 10 Mbps and 100 Mbps RMII Interface

RMII is a reduced MII interface specified by the RMII Consortium^{TM} standard. The purpose of this interface is to provide a low cost alternative to the MII, with 8 pin interface instead of 16 (Not including MDC, MDIO pins).

The FEC uses a gasket (MIIGSK) to support RMII. It will operate in RMII mode by setting FEC_MIIGSK_CFGR[I/F_MODE] = 01, while setting FEC_MIIGSK_CFGR[I/F_MODE] = 00 will disable RMII mode. The speed selection in RMII Mode is supported by configuring the FEC_MIIGSK_CFGR[FRCONT]. The serial management interface is still available in RMII mode.

28.2.2.3 10-Mbps 7-Wire Interface Operation

The FEC supports a 7-wire interface as used by many 10 Mbps ethernet transceivers. The FEC_RCR[MII_MODE] bit controls this functionality. If this bit is cleared, the MII mode is disabled and the 10 Mbps, 7-wire mode is enabled.

28.2.3 Address Recognition Options

The address options supported are promiscuous, broadcast reject, individual address (hash or exact match), and multicast hash match. Address recognition options are discussed in detail in Ethernet Address Recognition.

28.2.4 Internal Loopback

Two levels of internal loopback modes are supported. The first level, which will loop data back before they enter MIIGSK, is selected by FEC_RCR[LOOP]. The second level, which will loop data back inside MIIGSK RMII domain, is selected by setting FEC_MIIGSK_CFGR[LBMODE] and FEC_MIIGSK_CFGR[I/F_MODE] = 01. Loopback mode is discussed in detail in Internal and External Loopback.

28.3 Functional Description

This section provides a detailed description of the functions of the FEC including:

- Network interface options
- Frame transmission and reception
- Full-duplex flow control
- Internal and external loopback

28.3.1 Network Interface Options

The FEC supports an RMII interface for 10/100 Mbps Ethernet and a 7-wire serial interface for 10 Mbps Ethernet.

The interface mode is selected by FEC_MIIGSK_CFGR[I/F_MODE] together with the FEC_RCR[MII_MODE] bit as shown in the table below.

Table 28-1. Interface Mode Selection

FEC_MIIGSK_CFGR[I/F_MODE]	FEC_RCR[MII_MODE]	Interface Mode Selected
00	0	7-Wire
01	1	RMII

In RMII mode, a reduced set of 10 signals (a subset of the 18 signals defined by the IEEE 802.3 standard) are used. These signals are shown in Table 28-2.

Table 28-2. RMII Mode Signal Configuration

Signal Description	FEC Signal Name	Direction
Synchronous clock reference (REF_CLK)	FEC_TX_CLK	In
Transmit Enable	FEC_TX_EN	Out
Transmit Data	FEC_TXD[1:0]	Out
Carrier Sense/Receive Data Valid (CRS_DV)	FEC_RX_DV	In

Table continues on the next page...

Table 28-2. RMII Mode Signal Configuration (continued)

Signal Description	FEC Signal Name	Direction
Receive Data	FEC_RXD[1:0]	In
Receive Error	FEC_RX_ER	In
Management Data Clock	FEC_MDC	Out
Management Data Input/Output	FEC_MDIO	I/O

The 7-wire serial interface operates in what is generally referred to as AMD mode. 7-wire mode connections to the external transceiver are shown in Table 28-3.

Table 28-3. 7-Wire Mode Signal Configuration

Signal Description	FEC Signal Name	Direction
Transmit clock	FEC_TX_CLK	In
Transmit enable	FEC_TX_EN	Out
Transmit data	FEC_TXD[0]	Out
Collision	FEC_COL	In
Receive clock	FEC_RX_CLK	In
Receive data valid	FEC_RX_DV	In
Receive data	FEC_RXD[0]	In

28.3.2 FEC Frame Transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After FEC_ECR[ETHER_EN] is set to 1 and data appears in the transmit FIFO, the FEC is able to transmit on to the network.

When the transmit FIFO fills to the watermark (defined by the FEC_TFWR register), the FEC transmit logic asserts FEC_TX_EN and starts transmitting the preamble (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller postpones transmission if the network is busy (that is, the carrier sense signal FEC_CRS is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense remains inactive for 60 bit periods. If so, the transmission begins after waiting an additional 36 bit periods (96 bit periods after carrier sense originally became inactive). See Transmission Error Handling for more details.

If a collision occurs during transmission of a frame in half-duplex mode, the Ethernet controller follows the specified backoff procedures (see Collision Handling) and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least

the first 64 bytes of the transmit frame, so that they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data has been transmitted, if the TC bit is set in the transmit frame control word then a 32-bit cyclic redundancy check (CRC) known as the frame check sequence (FCS) is appended. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Short frames are automatically padded by the transmit logic (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set to 1).

Both buffer (TXB) and frame (TXF) interrupts can be generated as determined by the settings in the FEC_EIMR.

The transmit error interrupts are HBERR, BABT, LATE_COL, COL_RETRY_LIM, and XFIFO_UN. If the transmit frame length exceeds MAX_FL bytes the BABT interrupt is asserted: however, the entire frame is transmitted (no truncation).

Transmission is paused by setting the graceful transmit stop (GTS) bit in the FEC_TCR register. When the FEC_TCR[GTS] is set to 1, the FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame either finishes or terminates with a collision. After the transmitter has stopped the GRA (graceful stop complete) interrupt is asserted. If FEC_TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

The Ethernet controller transmits bytes LSB first.

28.3.2.1 Transmit Inter-Packet Gap (IPG) Time

The minimum inter-packet gap (IPG) time for back-to-back transmission is 96 bit periods. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96-bit-period IPG counter. Frame transmission begins 96 bit periods after carrier sense is negated if it stays negated for at least 60 bit periods. If carrier sense is asserted during the last 36 bit periods, it is ignored and a collision occurs.

28.3.2.2 Collision Handling

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit periods, transmitting a jam pattern consisting of 32 ones. If the collision occurs during the preamble sequence, the jam pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit periods, the retry process is initiated. The transmitter waits a random number of slot periods, where one slot period is 512 bit periods. If a collision occurs after 512 bit periods, then no retransmission is performed and the end of frame buffer is closed with a late collision (LC) error indication.

28.3.2.3 Transmission Error Handling

The Ethernet controller reports frame transmission error conditions using the FEC RxBDs, the FEC_EIR register, and the MIB block counters.

There are four types of transmission errors:

- Transmitter underrun
- Retransmission attempts limit expired
- Late collision
- Heartbeat

The FEC's procedures for handling these errors are described in the following subsections.

28.3.2.3.1 Transmitter Underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error, then stops transmitting. All remaining buffers for that frame are then flushed and closed. The UN bit is set in the FEC_EIR and the UN interrupt is asserted (if enabled in the FEC_EIMR register). The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

28.3.2.3.2 Retransmission Attempts Limit Expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the RL bit is set in the FEC_EIR. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

The "RL" interrupt is asserted if enabled in the FEC_EIMR register.

28.3.2.3.3 Late Collision

When a collision occurs after the slot time (512 bits starting at the preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the LC bit is set in the FEC_EIR register. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

The LC interrupt is asserted if enabled in the FEC_EIMR register.

28.3.2.3.4 Heartbeat

Some transceivers have a self-test feature called "heartbeat" or "signal quality error." To signify a good self-test, the transceiver indicates a collision to the FEC within 4 microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver still seems to be functioning properly. This is called the heartbeat condition.

If the HBC bit is set in the FEC_TCR register and the heartbeat condition is not detected by the **FEC** after a frame transmission, then a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets the HB bit in the FEC_EIR register, and generates the HBERR interrupt if it is enabled.

28.3.3 FEC Frame Reception

The FEC receiver is designed to work with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking.

When the driver enables the FEC receiver by setting FEC_ECR[ETHER_EN] to 1, it immediately starts processing receive frames. When FEC_RX_DV asserts, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the frame is processed by the receiver. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit periods of RX_D0 following assertion of FEC_RX_DV are ignored. Following the first 16 bit periods the data sequence is checked for alternating 1/0s. If a 0b11 or 0b00 data sequence is detected during bit periods 17 to 21, the remainder of the frame is ignored. After bit period 21, the data sequence is monitored for a valid SFD (11). If a 0b00 is detected, the frame is rejected. When a 0b11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes can occur, but if a 0b00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame have been received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data has been received, and if address recognition has not rejected the frame, the receive FIFO is signaled that the frame is accepted and can be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Thus no collision fragments are presented to the user except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV and TR status bits, and the frame length. See Reception Error Handling for more details.

Receive Buffer (RXB) and Frame Interrupts (RXF) can be generated if enabled by the FEC_EIMR register. The only receive error interrupt is babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See Ethernet Receive Buffer Descriptor (RxBD) for more details.

When the receive frame is complete, the FEC sets the L bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E bit. The Ethernet controller next generates a maskable interrupt (RXF bit in FEC_EIR, maskable by RXF bit in FEC_EIMR), indicating that a frame has been received and is in memory. The Ethernet controller then waits for a new frame.

The Ethernet controller receives serial data LSB first.

28.3.3.1 Receive Inter-Packet Gap (IPG) Time

The receiver receives back-to-back frames with a minimum spacing of 28-bit periods. If an inter-packet gap between receive frames is less than 28 bit periods, the second frame may be discarded by the receiver.

28.3.3.2 Ethernet Address Recognition

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in Figure 28-2 illustrates the address recognition decisions made by the receive block, while Figure 28-3 illustrates the decisions made by the microcontroller.

The FEC filters the received frames based on destination address (DA) type - individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the DA field.

If the DA is a broadcast address, and broadcast reject (FEC_RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in Figure 28-2. Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in Figure 28-3.

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in FEC_GAUR and FEC_GALR. If a hash match occurs, the receiver accepts the frame. The hash algorithm is described in Hash Algorithm.

If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines that the received frame is a valid pause frame, then the frame is rejected.

NOTE

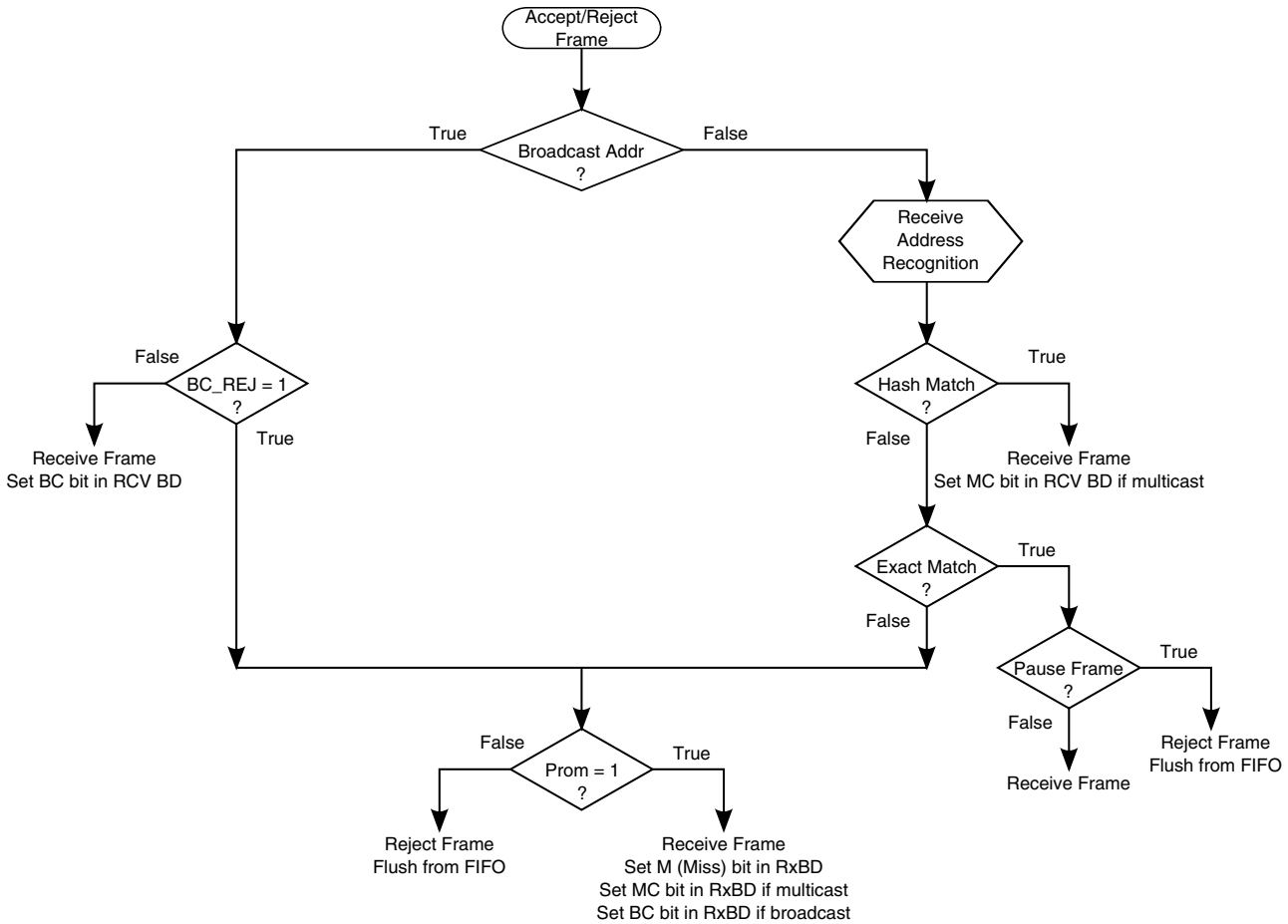
The receiver detects a pause frame with the DA field set to either the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that the user programs in the FEC_PALR and FEC_PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, FEC_IAUR and FEC_IALR (the hash algorithm is described in Hash Algorithm). In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on pause frame detection, shown in Figure 28-2.

If neither a hash match (group or individual), nor an exact match occur, then if promiscuous mode is enabled (FEC_RCR[PROM] = 1), the frame is accepted and the MISS bit in the receive buffer descriptor is set. Otherwise, the frame is rejected.

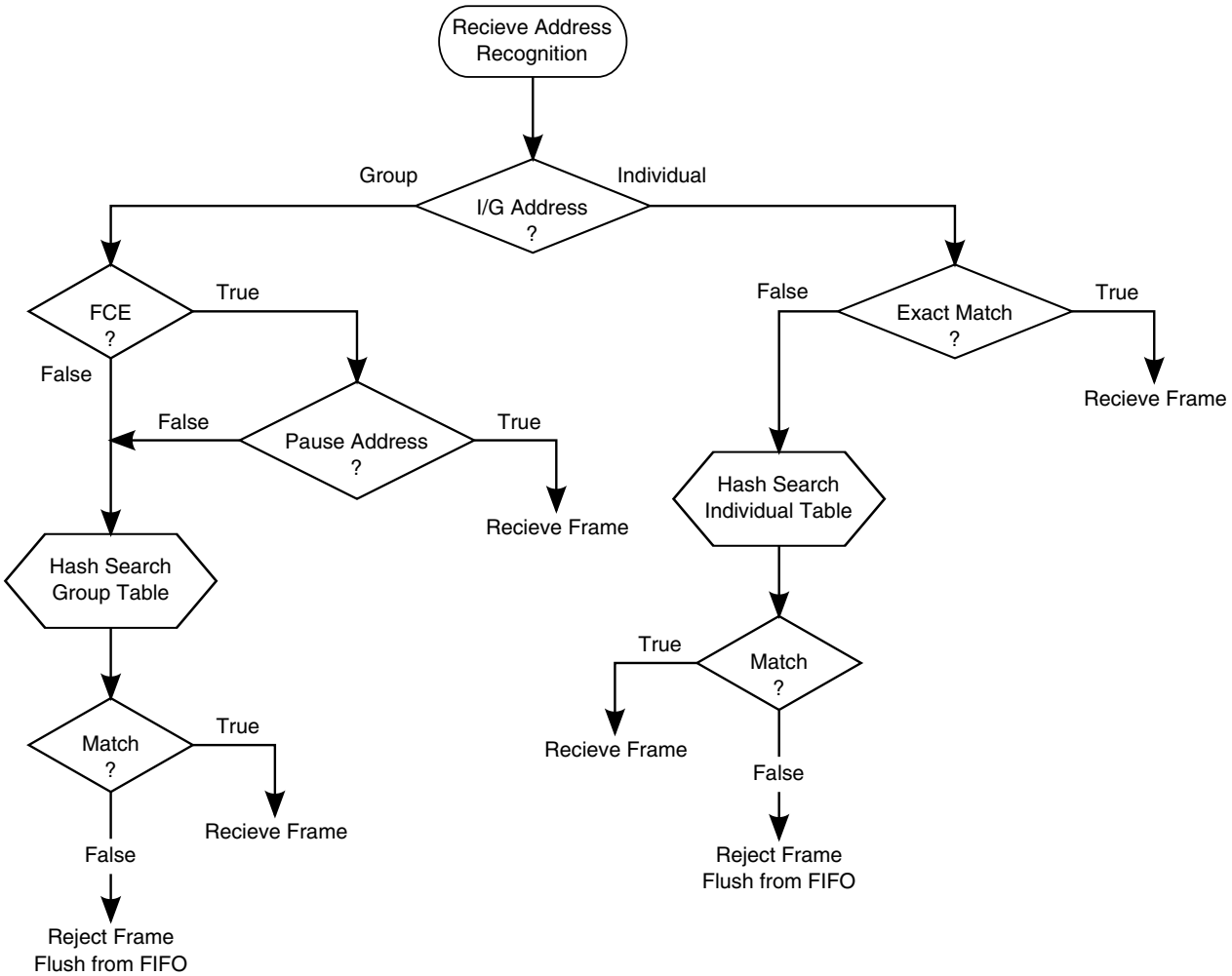
Similarly, if the DA is a broadcast address, broadcast reject (FEC_RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, then the frame is accepted and the MISS bit in the receive buffer descriptor is set. Otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.



NOTES:
BC_REJ - Field in RCR register (BroadCast REJect)
PROM-Field in RCR register (PROMiscuous mode)
Pause Frame - Valid PAUSE frame received

Figure 28-2. Ethernet Address Recognition-Receive Block Decisions



NOTES:
FCE - field in RCR register (Flow Control Enable)
I/G - Individual/Group bit in Destination Address (least signifigant bit in first byte recieved in MAC frame)

Figure 28-3. Ethernet Address Recognition-Microcode Decisions

28.3.3.2.1 Hash Algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, which are represented by 64 bits stored in FEC_GAUR, FEC_GALR (group address hash match) or FEC_IAUR, FEC_IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The MSB of the CRC result selects FEC_GAUR (MSB = 1) or FEC_GALR (MSB = 0). The least significant five bits of the hash result select the bit within the selected register. If the CRC generator selects a bit that is set in the hash table, the frame is accepted; otherwise, it is rejected. For example, if eight group addresses are stored in the hash table

and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The hash table registers must be initialized by the user. The CRC32 polynomial to use in computing the hash is shown in the following equation:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

Table 28-4 shows example destination addresses and corresponding hash values is included below for reference.

Table 28-4. Destination Address to 6-Bit Hash

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
65:FF:FF:FF:FF:FF	0x0	0
55:FF:FF:FF:FF:FF	0x1	1
15:FF:FF:FF:FF:FF	0x2	2
35:FF:FF:FF:FF:FF	0x3	3
B5:FF:FF:FF:FF:FF	0x4	4
95:FF:FF:FF:FF:FF	0x5	5
D5:FF:FF:FF:FF:FF	0x6	6
F5:FF:FF:FF:FF:FF	0x7	7
DB:FF:FF:FF:FF:FF	0x8	8
FB:FF:FF:FF:FF:FF	0x9	9
BB:FF:FF:FF:FF:FF	0xA	10
8B:FF:FF:FF:FF:FF	0xB	11
0B:FF:FF:FF:FF:FF	0xC	12
3B:FF:FF:FF:FF:FF	0xD	13
7B:FF:FF:FF:FF:FF	0xE	14
5B:FF:FF:FF:FF:FF	0xF	15
27:FF:FF:FF:FF:FF	0x10	16
07:FF:FF:FF:FF:FF	0x11	17

Table continues on the next page...

Table 28-4. Destination Address to 6-Bit Hash (continued)

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
57:FF:FF:FF:FF:FF	0x12	18
77:FF:FF:FF:FF:FF	0x13	19
F7:FF:FF:FF:FF:FF	0x14	20
C7:FF:FF:FF:FF:FF	0x15	21
97:FF:FF:FF:FF:FF	0x16	22
A7:FF:FF:FF:FF:FF	0x17	23
99:FF:FF:FF:FF:FF	0x18	24
B9:FF:FF:FF:FF:FF	0x19	25
F9:FF:FF:FF:FF:FF	0x1A	26
C9:FF:FF:FF:FF:FF	0x1B	27
59:FF:FF:FF:FF:FF	0x1C	28
79:FF:FF:FF:FF:FF	0x1D	29
29:FF:FF:FF:FF:FF	0x1E	30
19:FF:FF:FF:FF:FF	0x1F	31
D1:FF:FF:FF:FF:FF	0x20	32
F1:FF:FF:FF:FF:FF	0x21	33
B1:FF:FF:FF:FF:FF	0x22	34
91:FF:FF:FF:FF:FF	0x23	35
11:FF:FF:FF:FF:FF	0x24	36
31:FF:FF:FF:FF:FF	0x25	37
71:FF:FF:FF:FF:FF	0x26	38
51:FF:FF:FF:FF:FF	0x27	39
7F:FF:FF:FF:FF:FF	0x28	40
4F:FF:FF:FF:FF:FF	0x29	41
1F:FF:FF:FF:FF:FF	0x2A	42
3F:FF:FF:FF:FF:FF	0x2B	43
BF:FF:FF:FF:FF:FF	0x2C	44
9F:FF:FF:FF:FF:FF	0x2D	45
DF:FF:FF:FF:FF:FF	0x2E	46
EF:FF:FF:FF:FF:FF	0x2F	47
93:FF:FF:FF:FF:FF	0x30	48
B3:FF:FF:FF:FF:FF	0x31	49
F3:FF:FF:FF:FF:FF	0x32	50
D3:FF:FF:FF:FF:FF	0x33	51
53:FF:FF:FF:FF:FF	0x34	52

Table continues on the next page...

Table 28-4. Destination Address to 6-Bit Hash (continued)

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
73:FF:FF:FF:FF:FF	0x35	53
23:FF:FF:FF:FF:FF	0x36	54
13:FF:FF:FF:FF:FF	0x37	55
3D:FF:FF:FF:FF:FF	0x38	56
0D:FF:FF:FF:FF:FF	0x39	57
5D:FF:FF:FF:FF:FF	0x3A	58
7D:FF:FF:FF:FF:FF	0x3B	59
FD:FF:FF:FF:FF:FF	0x3C	60
DD:FF:FF:FF:FF:FF	0x3D	61
9D:FF:FF:FF:FF:FF	0x3E	62
BD:FF:FF:FF:FF:FF	0x3F	63

28.3.3.3 Reception Error Handling

The Ethernet controller reports frame reception error conditions using the FEC RxBDs, the FEC_EIR register, and the MIB counters.

There are five types of reception errors:

- Overrun
- Non-octet (dribbling bits)
- CRC
- Frame-length violation
- Truncation

These are described in the following subsections.

28.3.3.3.1 Overrun

If the receive block has data to put into the receive FIFO and the receive FIFO is full, the FEC sets the OV bit in the RxBD. All subsequent data in the frame is discarded, and subsequent frames can also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. This frame must be discarded by the driver.

28.3.3.3.2 Non-Octet (Dribbling Bits)

The Ethernet controller handles up to seven dribbling bits when the receive frame terminates past an non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, then the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, then no error is reported.

28.3.3.3.3 CRC

When a CRC error occurs with no dribble bits, the FEC closes the buffer and sets the CR bit in the RxBD. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

28.3.3.3.4 Frame Length Violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and the LG bit in the end of frame RxBD is set. The frame is not truncated unless the frame length exceeds 2047 bytes).

28.3.3.3.5 Truncation

When the receive frame length exceeds 2047 bytes, the frame is truncated and the TR bit is set in the RxBD.

28.3.4 Full-Duplex Flow Control

Full-duplex flow control allows the user to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, FEC data frame transmission stops for a given pause duration.

To enable pause frame detection, the FEC must operate in full-duplex mode (FEC_TCR[FDEN] asserted) and flow control enable (FEC_RCR[FCE]) must be asserted. The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in Table 28-5. In addition, the receive status associated with the frame indicates that the frame is valid.

Table 28-5. Pause Frame Field Specification

48-bit destination address	0x0180_C200_0001 or physical address
48-bit source address	Any
16-bit type	0x8808
16-bit opcode	0x0001

Table continues on the next page...

Table 28-5. Pause Frame Field Specification (continued)

48-bit destination address	0x0180_C200_0001 or physical address
16-bit pause duration	0x0000-0xFFFF

Pause frame detection is performed by the receiver and microcontroller modules. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, FEC_TCR[GTS] is asserted by the FEC internally. When transmission has paused, the FEC_EIR[GRA] interrupt is asserted and the pause timer begins to increment.

NOTE

The pause timer makes use of the transmit backoff timer hardware, which is used for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments after every slot time, until FEC_OPD[PAUSE_DUR] slot times have expired.

On FEC_OPD[PAUSE_DUR] expiration, FEC_TCR[GTS] is cleared allowing FEC data frame transmission to resume.

NOTE

The receive flow control pause (FEC_TCR[RFC_PAUSE]) status bit is asserted while the transmitter is paused due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and the user must assert flow control pause (FEC_TCR[TFC_PAUSE]). On assertion of transmit flow control pause (FEC_TCR[TFC_PAUSE]), the transmitter asserts FEC_TCR[GTS] internally. When the transmission of data frames stops, the FEC_EIR[GRA] (graceful stop complete) interrupt asserts. Following FEC_EIR[GRA] assertion, the pause frame is transmitted. On completion of pause frame transmission, flow control pause (FEC_TCR[TFC_PAUSE]) and FEC_TCR[GTS] are cleared internally.

The user must specify the desired pause duration in the FEC_OPD register.

NOTE

When the transmitter is paused due to receiver/microcontroller pause frame detection, transmit flow control pause (FEC_TCR[TFC_PAUSE]) still can be asserted and causes the transmission of a single pause frame. In this case, the FEC_EIR[GRA] interrupt is not asserted.

28.3.5 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Both internal and external loopback are configured using combinations of the LOOP and DRT bits in the FEC_RCR register, the FDEN bit in the FEC_TCR register and the LBMODE bit in FEC_MIIGSK_CFGR register. Furthermore internal loopback has two levels which loops data back in MII domain and RMII domain respectively.

For both internal and external loopback set FDEN = 1 and FEC_RCR[DRT] = 0.

For the MII domain internal loopback set FEC_RCR[LOOP] = 1 and FEC_MIIGSK_CFGR[LBMODE] = 0. FEC_TX_EN and FEC_TX_ER will not assert during internal loopback. During the MII domain internal loopback, the transmit/receive data rate is higher than in normal operation because the internal system clock is used by the transmit and receive blocks instead of the clocks from the external transceiver. This will cause an increase in the required system bus bandwidth for transmit and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underrun and receive FIFO overflow.

For the RMII domain internal loopback, set FEC_RCR[LOOP] = 0 and FEC_MIIGSK_CFGR[LBMODE] = 1. The data is sent from the MII domain into MIIGSK, converted to RMII format by a RMII transmitter inside the MIIGSK, then looped back through a RMII receiver of the MIIGSK, finally gets back to the MII domain.

For external loopback set FEC_RCR[LOOP] = 0, FEC_MIIGSK_CFGR[LBMODE] = 0, and configure the external transceiver for loopback.

28.4 Initialization/Application Information

28.4.1 Initialization Sequence

This section describes which registers are reset due to hardware reset, which are reset by the FEC RISC, and what locations the user must initialize prior to enabling the FEC.

28.4.1.1 Hardware Controlled Initialization

In the FEC, registers and control logic that generate interrupts are reset by hardware. A hardware reset negates output signals and resets general configuration bits.

Other registers are reset when the FEC_ECR[ETHER_EN] bit is cleared, as indicated in Table 28-6. FEC_ECR[ETHER_EN] is cleared by a hard reset or can be cleared by software to halt operation. By clearing FEC_ECR[ETHER_EN], the configuration control registers such as the FEC_TCR and FEC_RCR do not reset, but the entire data path resets.

Table 28-6. Effect on FEC of Clearing FEC_ECR[ETHER_EN]

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
FEC_RDAR	Cleared
FEC_TDAR	Cleared
Descriptor Controller block	Halt operation
MIIGSK	Interface to transceiver is disabled

28.4.1.2 User Initialization (Prior to Asserting FEC_ECR[ETHER_EN])

The user must initialize portions of the FEC prior to setting the FEC_ECR[ETHER_EN] bit. The exact values depend on the particular application. The order of initializations is not important except those that are explicitly mentioned.

The FEC and FEC FIFO/DMA registers which require initialization are defined in Table 28-7.

Table 28-7. Registers Requiring Initialization before Setting FEC_ECR[ETHER_EN]

Register	Location (FEC or FIFO/DMA)	Comments
FEC_EIMR	FEC	Requires initialization
FEC_EIR	FEC	Must be cleared by writing 0xFFFF_FFFF
FEC_TFWR	FEC	Optional
FEC_IALR / FEC_IAUR	FEC	-
FEC_GAUR / FEC_GALR	FEC	-

Table continues on the next page...

Table 28-7. Registers Requiring Initialization before Setting FEC_ECR[ETHER_EN] (continued)

Register	Location (FEC or FIFO/ DMA)	Comments
FEC_PALR / FEC_PAUR	FEC	-
FEC_OPD	FEC	Only needed for full-duplex flow control
FEC_RCR	FEC	-
FEC_TCR	FEC	-
FEC_MSCR	FEC	Optional
FEC_MIIGSK_CFGR	FEC	Must be configured before setting FEC_MIIGSK_ENR[EN]
FEC_MIIGSK_ENR	FEC	EN bit must be set
MIB counters	FEC	Must be cleared
FEC_FRSR	FIFO/DMA	Initialization is optional
FEC_EMRBR	FIFO/DMA	-
FEC_ERDSR	FIFO/DMA	-
FEC_ETDSR	FIFO/DMA	-
Transmit Descriptor ring	FIFO/DMA	Must be emptied
Receive Descriptor ring	FIFO/DMA	Must be emptied

28.4.1.3 Microcontroller Initialization

In the FEC, the descriptor control RISC initializes some registers after FEC_ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, the hardware is ready for operation.

The microcontroller initialization sequence follows:

1. Initialize BackOff Random Number Seed
2. Activate Receiver
3. Activate Transmitter
4. Clear Transmit FIFO
5. Clear Receive FIFO
6. Initialize Transmit Ring Pointer
7. Initialize Receive Ring Pointer
8. Initialize FIFO Count Register

28.4.1.4 User Initialization (after asserting FEC_ECR[ETHER_EN])

After asserting FEC_ECR[ETHER_EN], the user can set up the buffer/frame descriptors and write to the FEC_TDAR and FEC_RDAR. See Buffer Descriptors for more details.

28.4.2 Buffer Descriptors

This section provides a description of the operation of the driver/DMA through the buffer descriptors (BD). It is followed by a detailed description of the receive and transmit descriptor fields.

28.4.2.1 Driver/DMA Operation with Buffer Descriptors

The data for the FEC frames must reside in memory external to the FEC. The data for a frame is placed in one or more buffers. Associated with each buffer is a buffer descriptor (BD) which contains a starting address (pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read in by the FEC DMA engine.

Software "produces" buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit "produces" the buffer. Software writing to either the FEC_TDAR or FEC_RDAR tells the FEC that a buffer has been placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and "consumes" the buffers after they have been produced. After the data DMA is complete and the buffer descriptor status bits have been written by the DMA engine, the RxBD[E] or TxBD[R] bit is cleared by hardware to signal the buffer has been "consumed." Software can poll the BDs to detect when the buffers have been consumed or can rely on the buffer/frame interrupts. These buffers can then be processed by the driver and returned to the free list.

The FEC_ECR[ETHER_EN] signal operates as a reset to the BD/DMA logic. When FEC_ECR[ETHER_EN] is cleared the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before the FEC_ECR[ETHER_EN] bit is set.

The buffer descriptors operate as two separate rings. FEC_ERDSR defines the starting address for receive BDs and FEC_ETDSR defines the starting address for transmit BDs. The last buffer descriptor in each ring is defined by the Wrap (W) bit. When set, W indicates that the next descriptor in the ring is at the location pointed to by FEC_ERDSR and FEC_ETDSR for the receive and transmit rings, respectively.

NOTE

Buffer descriptor rings must start on a 128-bit boundary.

28.4.2.2 Ethernet Transmit Buffer Descriptor (TxBD)

Table 28-8 shows the transmit buffer descriptor format. Table 28-9 describes the TxBD fields.

Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated by individual interrupt bits (error conditions) and in statistic counters in the MIB block. See Message Information Block (MIB) Counters Memory Map, for more details.

Table 28-8. Transmit Buffer Descriptor (TxBD)

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	R	TO1	W	TO2	L	TC	ABC	PTP								
	Data Length															
0x0004	Tx Data Buffer Pointer A[31:16]															
	Tx Data Buffer Pointer A[15:0]															

Table 28-9. Transmit Buffer Descriptor Field Definitions

Offset	Field	Description
0x0000	31 R	Ready. Written by the FEC and the user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD can be written by the user after this bit is set.
0x0000	30 TO1	Transmit software ownership. This field is reserved for software use. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	29 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in FEC_ETDSR.
0x0000	28 TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	27 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.

Table continues on the next page...

Table 28-9. Transmit Buffer Descriptor Field Definitions (continued)

Offset	Field	Description
0x0000	26 TC	Tx CRC. Written by user (only valid if L = 1). 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
0x0000	25 ABC	Append bad CRC. Written by user (only valid if L = 1). 0 No effect 1 Transmit the CRC sequence inverted after the last data byte (regardless of TC value).
0x0000	24 PTP	PTP. Written by user (only valid if L = 1). 0 No effect 1 The frame is an IEEE1588 PTP frame. FEC will inform IPTP Assist modue to capture the timestamp for this frame in case of successful transmission. If unmasked, an interrupt will be issued by the IPTP Assist block after the timestamp is available.
0x0000	23-16	Reserved.
0x0000	15-0 Data Length	Data Length, written by user. Data length is the number of octets the FEC transmits from this BD's data buffer. It is never modified by the FEC. Bits [10:0] are used by the DMA engine, bits[15:11] are ignored.
0x0004	31-0 A	Tx data buffer pointer[1]

1. The transmit buffer pointer, which contains the address of the associated data buffer, must always be divisible by 16. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

28.4.2.2.1 Driver/DMA Operation with Transmit Buffer Descriptors

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. After the software driver has set up the buffers for a frame, it sets up the corresponding BDs. In the TxBD the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword, and the buffer pointer in the second longword.The last step in setting up the BDs for a transmit frame is to set the R bit in the first BD for the frame. The driver follows that with a write to FEC_TDAR which triggers the FEC to poll the next BD in the ring.

The Ethernet controller confirms transmission by clearing the ready bit (R bit) when DMA of the buffer is complete.

28.4.2.2.1.1 Transmit Frame in Multiple Buffers

Typically a transmit frame is divided between multiple buffers. For example, it is possible to have an application payload in one buffer, TCP header in a 2nd buffer, IP header in a 3rd buffer, and Ethernet/IEEE 802.3 header in a 4th buffer. The FEC does not prepend the Ethernet header (Destination Address, Source Address, Length/Type

field(s)), so this must be provided by the driver in one of the transmit buffers. The FEC or the driver can append the Ethernet CRC to the frame. The driver must set the TC bit in the transmit BD to determine whether the CRC is appended by the FEC or by the driver.

The driver (TxBD software producer) sets up Tx BDs in such a way that a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, first the BD's are initialized with pointer, length and control (W, L, TC, ABC) and then the TxBD[R] bits are set to 1 in *reverse order* (3rd, 2nd, 1st BD) to insure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the 2nd was made available, potentially causing a transmit FIFO underrun.

In the FEC, the DMA is notified by the driver that new transmit frame(s) are available by writing to the FEC_TDAR register. When this register is written to (data value is not significant) the FEC RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers, until a transmit BD is encountered with the R bit = 0. At this point the FEC polls this BD one more time. If the R bit = 0 the second time, then the RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to FEC_TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

28.4.2.3 Ethernet Receive Buffer Descriptor (RxBD)

Table 28-10. Receive Buffer Descriptor (RxBD)

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	E	RO1	W	RO2	L	PTP		M	B	MC	LG	NO		CR	OV	TR
	Data Length															
0x0004	Rx Data Buffer Pointer A[31:16]															
	Rx Data Buffer Pointer A[15:0]															

Table 28-11. Receive Buffer Descriptor Field Definitions

Offset	Field	Description
0x0000	31 E	Empty. Written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.

Table continues on the next page...

Table 28-11. Receive Buffer Descriptor Field Definitions (continued)

Offset	Field	Description
0x0000	30 RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	29 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in FEC_ERDSR.
0x0000	28 RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	27 L	Last in frame. Written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
0x0000	26 PTP	PTP frame received. Written by the FEC. This bit is valid only if the L-bit is set. 0 Received frame is not an IEEE1588 PTP frame. 1 Received frame is an IEEE1588 PTP frame. FEC has informed IPTP Assist block to capture the timestamp for this frame. If unmasked, an interrupt will be issued by the IPTP Assist block after the timestamp is available.
0x0000	25	Reserved.
0x0000	24 M	Miss. Written by the FEC. This bit is set by the FEC for frames that were accepted in promiscuous mode, but were flagged as a "miss" by the internal address recognition. Thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
0x0000	23 BC	Set if the DA is broadcast (FF:FF:FF:FF:FF:FF).
0x0000	22 MC	Set if the DA is multicast and not BC.
0x0000	21 LG	Rx frame length violation. Written by the FEC. A frame length greater than FEC_RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
0x0000	20 NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set the CR bit is not set.
0x0000	19	Reserved
0x0000	18 CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.

Table continues on the next page...

Table 28-11. Receive Buffer Descriptor Field Definitions (continued)

Offset	Field	Description
0x0000	17 OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and is zero. This bit is valid only if the L-bit is set.
0x0000	16 TR	Set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set the frame is discarded and the other error bits ignored as they can be incorrect.
0x0000	15-0 Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L = 0 (the value is equal to FEC_EMRBR), or the length of the frame including CRC if L = 1. It is written by the FEC once as the BD is closed.
0x0004	31-0	RX data buffer pointer[1]

1. The receive buffer pointer, which contains the address of the associated data buffer, must always be divisible by 16. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

28.4.2.3.1 Driver/DMA Operation with Receive Buffer Descriptors

Unlike the transmit case, the length of the receive frame is unknown by the driver ahead of time. Therefore the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the FEC_EMRBR register.

The driver (RxBD software producer) sets up some number of "empty" buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received, and clearing the E bit and writing to the L bit (1 indicates last buffer in frame), the frame status bits (if L = 1) and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end of frame buffers, the L=1 bit is set in the receive BD, and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame is discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not just the length of the last buffer.

For simplicity the driver can assign the default receive buffer length to be large enough to contain an entire frame, keeping in mind that a malfunction on the network or out-of-specification implementation could result in giant frames. Frames of 2 Kbytes (2048 bytes) or larger are truncated by the FEC at 2047 bytes, so software never sees a receive frame larger than 2047 bytes.

As in the transmit case, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the FEC_RDAR register. As frames are received the FEC fills receive buffers and update the associated BDs, then reads the next BD in the receive descriptor ring. If the FEC reads a receive BD and finds the E bit is cleared, it polls this BD once more. If E is still cleared, then the FEC stops reading receive BDs until the driver writes to FEC_RDAR.

NOTE

Whenever the software driver sets an E bit in one or more receive descriptors, the driver follows that with a write to FEC_RDAR.

28.5 Programmable Registers

28.5.1 Top Level Block Memory Map

The FEC implementation requires a 1 Kbyte memory space. This space is divided into 2 sections of 512 bytes each. The first is used for control/status registers, and the second contains event/statistics counters held in the MIB block.

Table 28-12 defines the top level memory map. For the base address of a particular block instantiation, see the system memory map.

Table 28-12. Block Memory Map

Base Address Offset	Function
0x0000-01FF	Control/Status Registers
0x0200-02FF	MIB Block Counters
0x0300-03FF	Extension Registers (MIIGSK)

28.5.2 Message Information Block (MIB) Counters Memory Map

Table 28-13 shows the MIB counters memory map, which defines the locations in the MIB RAM space where hardware maintained counters reside. It is the responsibility of software to poll the counters often enough to ensure that rollover is detected. For example, on a 100 Mbps channel an octets counter could roll over every 5.7 minutes.

These counters fall in the 0x0200-0x03FF address offset range, and are divided into RMON counters and IEEE counters, as follows:

- RMON counters are included which cover the Ethernet statistics counters defined in RFC 1757. In addition to the counters defined in the Ethernet statistics group, a counter is included to count truncated frames as the FEC only supports frame lengths up to 2047 bytes. The RMON counters are implemented independently for transmit and receive to insure accurate network statistics when operating in full-duplex mode.
- IEEE counters are included which support the mandatory and recommended counter packages defined in section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The IEEE basic package objects are supported by the FEC but do not require counters in the MIB block. In addition, some of the recommended package objects which are supported do not require MIB counters. Counters for transmit and receive full-duplex flow control frames are included as well.

Table 28-13. MIB Counters Memory Map

Base Address Offset	Mnemonic	Description
0x0200	RMON_T_DROP	Count of frames not counted correctly
0x0204	RMON_T_PACKETS	RMON Tx packet count
0x0208	RMON_T_BC_PKT	RMON Tx broadcast packets
0x020C	RMON_T_MC_PKT	RMON Tx multicast packets
0x0210	RMON_T_CRC_ALIGN	RMON Tx packets w CRC/Align error
0x0214	RMON_T_UNDERSIZE	RMON Tx packets < 64 bytes, good crc
0x0218	RMON_T_OVERSIZE	RMON Tx packets > MAX_FL bytes, good crc
0x021C	RMON_T_FRAG	RMON Tx packets < 64 bytes, bad crc
0x0220	RMON_T_JAB	RMON Tx packets > MAX_FL bytes, bad crc
0x0224	RMON_T_COL	RMON Tx collision count
0x0228	RMON_T_P64	RMON Tx 64 byte packets
0x022C	RMON_T_P65TO127	RMON Tx 65 to 127 byte packets
0x0230	RMON_T_P128TO255	RMON Tx 128 to 255 byte packets
0x0234	RMON_T_P256TO511	RMON Tx 256 to 511 byte packets
0x0238	RMON_T_P512TO1023	RMON Tx 512 to 1023 byte packets
0x023C	RMON_T_P1024TO2047	RMON Tx 1024 to 2047 byte packets
0x0240	RMON_T_P_GTE2048	RMON Tx packets w > 2048 bytes
0x0244	RMON_T_OCTETS	RMON Tx octets
0x0248	IEEE_T_DROP	Count of frames not counted correctly
0x024C	IEEE_T_FRAME_OK	Frames transmitted OK
0x0250	IEEE_T_1COL	Frames transmitted with single collision
0x0254	IEEE_T_MCOL	Frames transmitted with multiple collisions
0x0258	IEEE_T_DEF	Frames transmitted after deferral delay
0x025C	IEEE_T_LCOL	Frames transmitted with late collision

Table continues on the next page...

Table 28-13. MIB Counters Memory Map (continued)

Base Address Offset	Mnemonic	Description
0x0260	IEEE_T_EXCOL	Frames transmitted with excessive collisions
0x0264	IEEE_T_MACERR	Frames transmitted with Tx FIFO underrun
0x0268	IEEE_T_CSERR	Frames transmitted with carrier sense error
0x026C	IEEE_T_SQE	Frames transmitted with SQE error
0x0270	IEEE_T_FDXFC	Flow control pause frames transmitted
0x0274	IEEE_T_OCTETS_OK	Octet count for frames transmitted w/o error
0x0284	RMON_R_PACKETS	RMON Rx packet count
0x0288	RMON_R_BC_PKT	RMON Rx broadcast packets
0x028C	RMON_R_MC_PKT	RMON Rx multicast packets
0x0290	RMON_R_CRC_ALIGN	RMON Rx packets w CRC/Align error
0x0294	RMON_R_UNDERSIZE	RMON Rx packets < 64 bytes, good crc
0x0298	RMON_R_OVERSIZE	RMON Rx packets > MAX_FL bytes, good crc
0x029C	RMON_R_FRAG	RMON Rx packets < 64 bytes, bad crc
0x02A0	RMON_R_JAB	RMON Rx packets > MAX_FL bytes, bad crc
0x02A4	RMON_R_RESVD_0	-
0x02A8	RMON_R_P64	RMON Rx 64 byte packets
0x02AC	RMON_R_P65TO127	RMON Rx 65 to 127 byte packets
0x02B0	RMON_R_P128TO255	RMON Rx 128 to 255 byte packets
0x02B4	RMON_R_P256TO511	RMON Rx 256 to 511 byte packets
0x02B8	RMON_R_P512TO1023	RMON Rx 512 to 1023 byte packets
0x02BC	RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets
0x02C0	RMON_R_P_GTE2048	RMON Rx packets w > 2048 bytes
0x02C4	RMON_R_OCTETS	RMON Rx octets
0x02C8	IEEE_R_DROP	Count of frames not counted correctly
0x02CC	IEEE_R_FRAME_OK	Frames received OK
0x02D0	IEEE_R_CRC	Frames received with CRC error
0x02D4	IEEE_R_ALIGN	Frames received with alignment error
0x02D8	IEEE_R_MACERR	Receive FIFO overflow count
0x02DC	IEEE_R_FDXFC	Flow control pause frames received
0x02E0	IEEE_R_OCTETS_OK	Octet count for frames received w/o error

28.5.3 MIIGSK Registers Memory Map

Table 28-14. MIIGSK Registers Memory Map

Offset	Mnemonic	Description
0x0300	FEC_MIIGSK_CFGR	MIIGSK Configuration Register
0x0308	FEC_MIIGSK_ENR	MIIGSK Enable Register

This section includes the FEC memory map and detailed descriptions of all the registers, followed by a description of the buffers.

The FEC is programmed by a combination of control and status registers (FEC_CSRs) and buffer descriptors. The FEC_CSRs are used for mode control and to extract global status information. The descriptors are used to pass data buffers and related buffer information between the hardware and software.

The FEC implementation requires a 1 Kbyte memory space. This space is divided into 2 sections of 512 bytes each. The first is used for control/status registers, and the second contains event/statistics counters held in the MIB block.

Table defines the top level memory map. For the base address of a particular block instantiation, see the system memory map.

Table 28-16. Block Memory Map

Base Address Offset	Function
0x0000-01FF	Control/Status Registers
0x0200-02FF	MIB Block Counters
0x0300-03FF	Extension Registers (MIIGSK)

The following table shows the FEC register memory map. For the base address of a particular block instantiation, see the system memory map.

FEC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FE_C004	Ethernet interrupt event register (FEC_EIR)	32	R/W	0000_0000h	28.5.4/ 1494
63FE_C008	Ethernet interrupt mask register (FEC_EIMR)	32	R/W	0000_0000h	28.5.5/ 1496

Table continues on the next page...

FEC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FE_C010	Receive descriptor active register (FEC_RDAR)	32	R/W	0000_0000h	28.5.6/ 1498
63FE_C014	Transmit descriptor active register (FEC_TDAR)	32	R/W	0000_0000h	28.5.7/ 1499
63FE_C024	Ethernet control register (FEC_ECR)	32	R/W	F000_0000h	28.5.8/ 1500
63FE_C040	MII management frame register (FEC_MMFR)	32	R/W	Unaffected	28.5.9/ 1501
63FE_C044	MII speed control register (FEC_MSCR)	32	R/W	0000_0000h	28.5.10/ 1503
63FE_C064	MIB control register (FEC_MIBC)	32	R/W	C000_0000h	28.5.11/ 1505
63FE_C084	Receive control register (FEC_RCR)	32	R/W	05EE_0001h	28.5.12/ 1506
63FE_C0C4	Transmit control register (FEC_TCR)	32	R/W	0000_0000h	28.5.13/ 1507
63FE_C0E4	Physical address low register (FEC_PALR)	32	R/W	0000_0000h	28.5.14/ 1508
63FE_C0E8	Physical address upper register (FEC_PAUR)	32	R/W	0000_8808h	28.5.15/ 1509
63FE_C0EC	Opcode and pause duration register (FEC_OPDR)	32	R/W	0001_0000h	28.5.16/ 1510
63FE_C118	Descriptor individual address upper register (FEC_IAUR)	32	R/W	0000_0000h	28.5.17/ 1510
63FE_C11C	Descriptor individual address lower register (FEC_IALR)	32	R/W	0000_0000h	28.5.18/ 1511
63FE_C120	Descriptor group address upper register (FEC_GAUR)	32	R/W	0000_0000h	28.5.19/ 1511
63FE_C124	Descriptor group address lower register (FEC_GALR)	32	R/W	0000_0000h	28.5.20/ 1512
63FE_C144	Transmit FIFO watermark register (FEC_TFWR)	32	R/W	0000_0000h	28.5.21/ 1512
63FE_C14C	FIFO receive bound register (FEC_FRBR)	32	R	0000_0600h	28.5.22/ 1513
63FE_C150	FIFO receive FIFO start registers (FEC_FRSR)	32	R/W	0000_0500h	28.5.23/ 1513
63FE_C180	Receive buffer descriptor ring start register (FEC_ERDSR)	32	R/W	Unaffected	28.5.24/ 1514
63FE_C184	Transmit buffer descriptor ring start register (FEC_ETDSR)	32	R/W	0000_0000h	28.5.25/ 1515

Table continues on the next page...

FEC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FE_C188	Maximum receive buffer size register (FEC_EMRBR)	32	R/W	0000_0000h	28.5.26/ 1515

28.5.4 Ethernet interrupt event register (FEC_EIR)

The FEC_EIR bit assignments are shown below. When an event occurs that sets a bit in the FEC_EIR, an interrupt is generated if the corresponding bit in the interrupt mask register (FEC_EIMR) is also set. The bit in the FEC_EIR is cleared if a one is written to that bit position; writing zero has no effect. This register is cleared upon hardware reset.

Interrupts can be divided into three classes as follows:

- Interrupts that occur in normal operation: these include GRA, TXF, TXB, RXF, RXB, and MII.
- Interrupts that result from errors or problems detected in the network or transceiver: these include HBERR, BABR, BABT, LC, and RL.
- Interrupts that result from internal errors are EBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters. The correspondence between interrupts and counters is shown in the following table. Software can choose to mask off the interrupts because these errors are visible to network management through the MIB counters.

Table 28-18. Error Interrupts and Block Counters

Interrupt	Counter(s)
HBERR	IEEE_T_SQE
BABR	RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
BABT	RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
LATE_COL	IEEE_T_LCOL
COL_RETRY_LIM	IEEE_T_EXCOL
XFIFO_UN	IEEE_T_MACERR

Address: FEC_EIR is 63FE_C000h base + 4h offset = 63FE_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	-																		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_EIR field descriptions

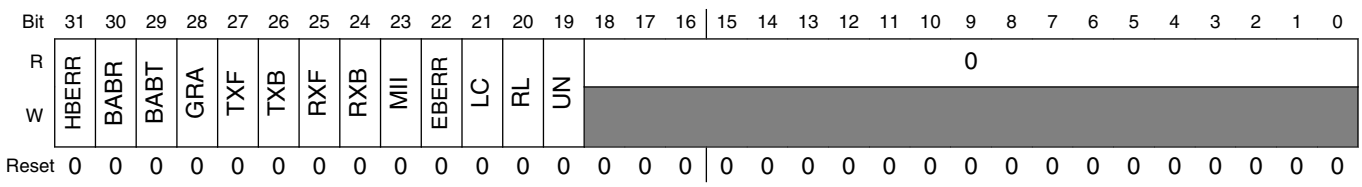
Field	Description
31 HBERR	Heartbeat error. This interrupt indicates that HBC is set in the FEC_TCR register and that the COL input was not asserted within the Heartbeat window following a transmission.
30 BABR	Babbling receive error. This bit indicates a frame was received with length in excess of FEC_RCR[MAX_FL] bytes.
29 BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded FEC_RCR[MAX_FL] bytes. This condition is usually caused by a frame that is too long being placed into the transmit data buffer(s). Truncation does not occur.
28 GRA	Graceful stop complete. This interrupt is asserted for one of three reasons. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. 1) A graceful stop, which was initiated by the setting of the FEC_TCR[GTS] bit is now complete. 2) A graceful stop, which was initiated by the setting of the FEC_TCR[TFC_PAUSE] bit is now complete. 3) A graceful stop, which was initiated by the reception of a valid full-duplex flow control "pause" frame is now complete. See the "Full-Duplex Flow Control" section of the Functional Description chapter.
27 TXF	Transmit frame interrupt. This bit indicates that a frame has been transmitted and that the last corresponding buffer descriptor has been updated.
26 TXB	Transmit buffer interrupt. This bit indicates that a transmit buffer descriptor has been updated.
25 RXF	Receive frame interrupt. This bit indicates that a frame has been received and that the last corresponding buffer descriptor has been updated.
24 RXB	Receive buffer interrupt. This bit indicates that a receive buffer descriptor has been updated that was not the last in the frame.
23 MII	MII interrupt. This bit indicates that the MII has completed the data transfer requested.
22 EBERR	Ethernet bus error. This bit indicates that a system bus error occurred when a DMA transaction was underway. When the EBERR bit is set, FEC_ECR[ETHER_EN] is cleared, halting frame processing by the FEC_ When this occurs software needs to insure that the FIFO controller and DMA are also soft reset.
21 LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded.
20 RL	Collision retry limit. This bit indicates that a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. Can only occur in half-duplex mode.
19 UN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
18–0 -	Reserved, read as 0

28.5.5 Ethernet interrupt mask register (FEC_EIMR)

The FEC_EIMR controls which of the interrupt events flagged in the FEC_EIR are allowed to generate actual interrupts. If the corresponding bits in both the FEC_EIR and FEC_EIMR are set, the interrupt is signaled to the ARM platform. The interrupt signal remains asserted until a 1 is written to the FEC_EIR bit (write 1 to clear) or a 0 is written to the FEC_EIMR bit. This register is cleared upon a hardware reset.

The FEC_EIMR bit assignments are the same as for the FEC_EIR.

Address: FEC_EIMR is 63FE_C000h base + 8h offset = 63FE_C008h



FEC_EIMR field descriptions

Field	Description
31 HBERR	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
30 BABR	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
29 BABT	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
28 GRA	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared.

Table continues on the next page...

FEC_EIMR field descriptions (continued)

Field	Description
	0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
27 TXF	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
26 TXB	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
25 RXF	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
24 RXB	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
23 MII	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
22 EBERR	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.

Table continues on the next page...

FEC_EIMR field descriptions (continued)

Field	Description
21 LC	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
20 RL	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
19 UN	Interrupt mask. Each bit corresponds to an interrupt source defined by the FEC_EIR register. The corresponding FEC_EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the FEC_EIR samples the signal generated by the interrupting source. The corresponding FEC_EIR bit reflects the state of the interrupt signal even if the corresponding FEC_EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
18–0 Reserved	This read-only field is reserved and always has the value zero. Reserved, read as 0

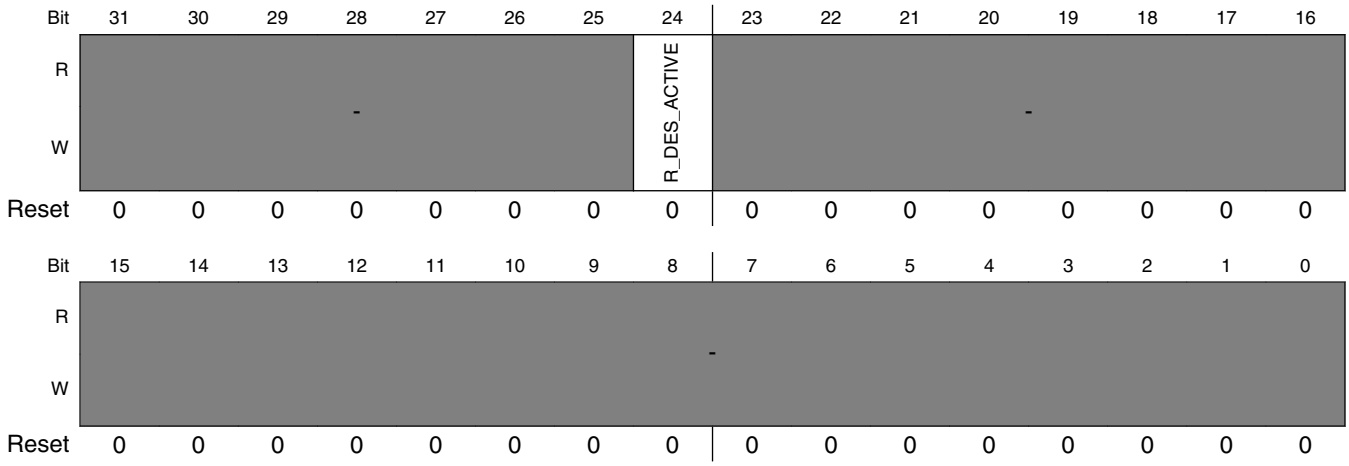
28.5.6 Receive descriptor active register (FEC_RDAR)

FEC_RDAR is a user-writeable command register which indicates that the receive descriptor ring has been updated, and that empty receive buffers have been produced by the driver with the empty bit set.

The FEC_RDAR[R_DES_ACTIVE] bit is set whenever the register is written, independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (provided FEC_ECR[ETHER_EN] is also set to 1). After the FEC polls a receive descriptor whose empty bit is not set, then the FEC clears the FEC_RDAR[R_DES_ACTIVE] bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors have been placed into the receive descriptor ring.

The FEC_RDAR is cleared at reset, and when FEC_ECR[ETHER_EN] is cleared.

Address: FEC_RDAR is 63FE_C000h base + 10h offset = 63FE_C010h



FEC_RDAR field descriptions

Field	Description
31–25 -	Reserved, read as 0
24 R_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device when the FEC polls a receive descriptor whose empty bit is not set. Also cleared when FEC_ECR[ETHER_EN] is cleared.
23–0 -	Reserved, read as 0

28.5.7 Transmit descriptor active register (FEC_TDAR)

The FEC_TDAR is a command register, written to by the user, to indicate that the transmit descriptor ring has been updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

Whenever the register is written the FEC_TDAR[X_DES_ACTIVE] bit is set to 1, independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and process transmit frames (provided FEC_ECR[ETHER_EN] is also set to 1). After the FEC polls a transmit descriptor whose ready bit is not set, then the FEC clears the FEC_TDAR[X_DES_ACTIVE] bit and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors have been placed into the transmit descriptor ring.

The FEC_TDAR is cleared at reset, when FEC_ECR[ETHER_EN] is cleared, or when FEC_ECR[RESET] is set to 1.

Address: FEC_TDAR is 63FE_C000h base + 14h offset = 63FE_C014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-							X_DES_ACTIVE	-							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_TDAR field descriptions

Field	Description
31–25 -	Reserved, read as 0
24 X_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device when the FEC polls a transmit descriptor whose ready bit is not set. Also cleared when FEC_ECR[ETHER_EN] is cleared.
23–0 -	Reserved, read as 0

28.5.8 Ethernet control register (FEC_ECR)

The ECR is used to enable/disable the FEC_ ECR is a read/write user register, though both fields in this register can also be altered by hardware.

Address: FEC_ECR is 63FE_C000h base + 24h offset = 63FE_C024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-														ETHER_EN	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_ECR field descriptions

Field	Description
31–2 -	Reserved.
1 ETHER_EN	When this bit is set, the FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception is immediately stopped and transmission is stopped after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. The ETHER_EN bit is altered by hardware under the following conditions: FEC_ECR[RESET] is set by software, in which case ETHER_EN is cleared an error condition causes the FEC_EIR[EBERR] bit to set, in which case ETHER_EN is cleared
0 RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC_ ETHER_EN is cleared and all other FEC registers take their reset values. Also, any transmission/ reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately 8 system clock cycles after RESET is written with a 1.

28.5.9 MII management frame register (FEC_MMFR)

The FEC_MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to FEC_MMFR causes a management frame to be generated unless the MII-SPEED field of the FEC_MSCR has been set to 0, in which case FEC_MSCR is set to a non-zero value and an MII frame is generated with the data previously written to FEC_MMFR. This allows FEC_MMFR and FEC_MSCR to be programmed in either order if the MII-SPEED field of FEC_MSCR is zero (for further details see MII speed control register (FEC_MSCR)).

The FEC_MMFR does not reset to a definite value.

To perform a read or write operation on the MII Management Interface, the FEC_MMFR must be written to by the user. To generate a valid read or write management frame, the ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII management interface write frame (write to a PHY register), the user must write {01 01 PHYAD REGAD 10 DATA} to the bit fields of the FEC_MMFR, as shown in MII management frame register (FEC_MMFR). Writing this pattern causes the control logic to shift out the data in the FEC_MMFR following a preamble generated by the control state machine. During this time the contents of the FEC_MMFR is altered as the contents are serially shifted and is unpredictable if read by

the user. After the write management frame operation has completed, the MII interrupt is generated. At this time the contents of the FEC_MMFR matches the original value written.

To generate an MII management interface read frame (read a PHY register) the user must write {01 10 PHYAD REGAD 10 XXXX} to the bit fields of the FEC_MMFR shown in MII management frame register (FEC_MMFR) (the contents of the 4-bit DATA field are arbitrary). Writing this pattern causes the control logic to shift out the data in the FEC_MMFR following a preamble generated by the control state machine. During this time the contents of the FEC_MMFR is altered as the contents are serially shifted, and is unpredictable if read by the user. After the read management frame operation has completed, the MII interrupt is generated. At this time the contents of the FEC_MMFR matches the original value written except for the DATA field whose contents have been replaced by the value read from the PHY register.

If the FEC_MMFR is written to while frame generation is in progress, the frame contents is altered. Software uses the MII interrupt to avoid writing to the FEC_MMFR while frame generation is in progress.

Address: FEC_MMFR is 63FE_C000h base + 40h offset = 63FE_C040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ST		OP		PA					RA					TA		DATA															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_MMFR field descriptions

Field	Description
31–30 ST	Start of frame delimiter. These bits must be programmed to 01 for a valid MII management frame.
29–28 OP	Operation code. This field must be programmed to 10 (read) or 01 (write) to generate a valid MII management frame. A value of 11 produces "read" frame operation while a value of 00 produces "write" frame operation, but these frames is not MII compliant.
27–23 PA	PHY address. This field specifies one of up to 32 attached PHY devices.
22–18 RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
17–16 TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
15–0 DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

28.5.10 MII speed control register (FEC_MSCR)

The FEC_MSCR provides control of the frequency of the MII clock (FEC_MDC signal), and allows a preamble drop on the MII management frame.

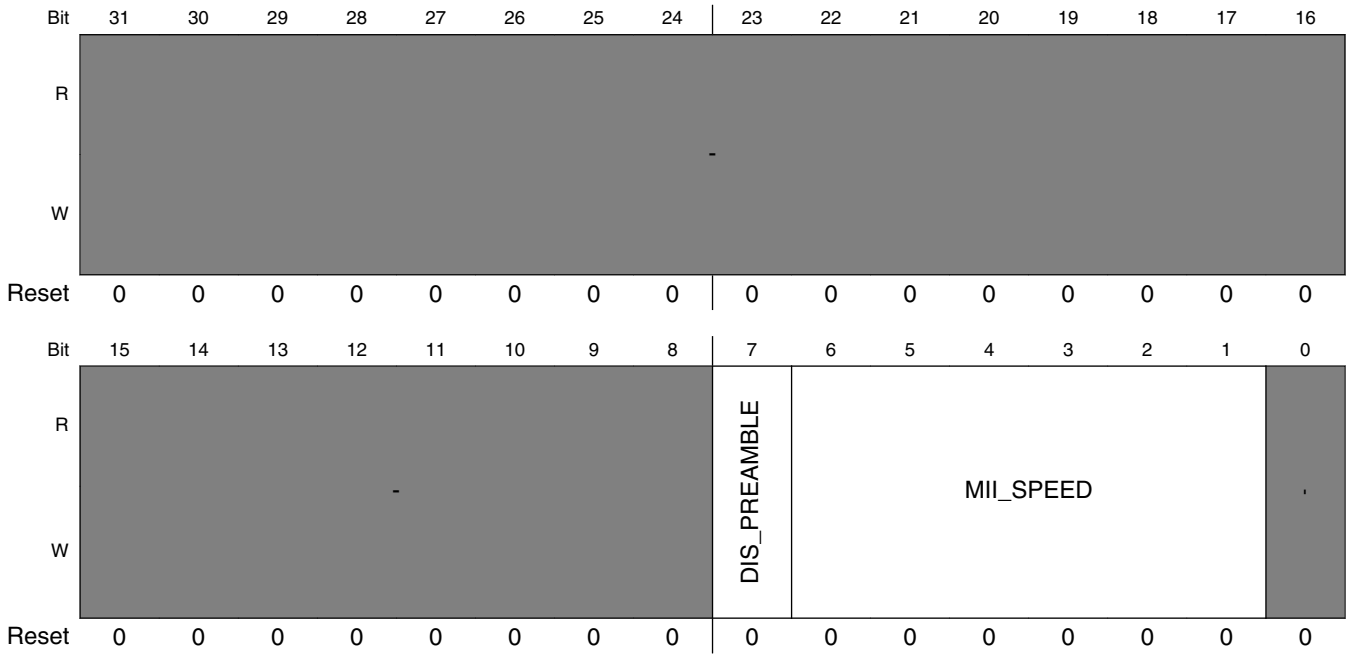
The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value in order to generate a read or write management frame. After the management frame is complete the FEC_MSCR can optionally be set to zero to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED is changed during operation (change takes effect following either a rising or falling edge of FEC_MDC).

The FEC_MDC frequency depends on both the system clock frequency and the MII_SPEED register. If the system clock is 25 MHz, programming the MII_SPEED register to 0x0000_0005 results in an FEC_MDC frequency of 25 MHz * 1/10 = 2.5 MHz. A table showing optimum values for MII_SPEED for different system clock frequencies is provided below.

Table 28-25. Programming Examples for FEC_MSCR

System Clock Frequency	MII_SPEED (field in reg)	FEC_MDC frequency
25 MHz	0x5	2.5 MHz
33 MHz	0x7	2.36 MHz
40 MHz	0x8	2.5 MHz
50 MHz	0xA	2.5 MHz
66 MHz	0xD	2.54 MHz

Address: FEC_MSCR is 63FE_C000h base + 44h offset = 63FE_C044h



FEC_MSCR field descriptions

Field	Description
31–8 -	Reserved, read as 0
7 DIS_PREAMBLE	Asserting this bit causes preamble (0xFFFF_FFFF) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
6–1 MII_SPEED	MII_SPEED controls the frequency of the MII management interface clock (FEC_MDC) relative to the system clock. A value of 0 in this field "turns off" the FEC_MDC and leave it in low voltage state. Any non-zero value results in the FEC_MDC frequency of 1/(MII_SPEED*2) of the system clock frequency.
0 -	Reserved, read as 0

28.5.11 MIB control register (FEC_MIBC)

The MIB control register is a read/write register used to provide control of and to observe the state of the Message Information Block (MIB). This register is accessed by user software if there is a need to disable the MIB operation. For example, in order to clear all MIB counters in RAM the user disables the MIB, then clears all the MIB RAM locations, then enables the MIB. The MIB_DISABLE bit is reset to 1. See MIB Counters Memory Map for the locations of the MIB counters.

Address: FEC_MIBC is 63FE_C000h base + 64h offset = 63FE_C064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MIB_DISABLE	MB_IDLE	-													
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

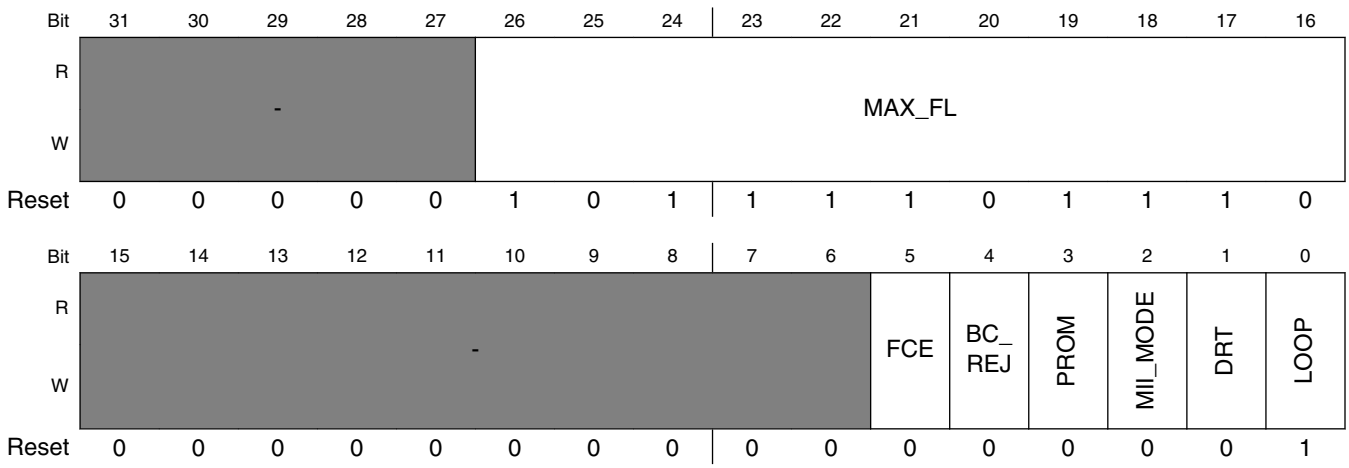
FEC_MIBC field descriptions

Field	Description
31 MIB_DISABLE	A read/write control bit. If set, the MIB logic halts and not update any MIB counters.
30 MB_IDLE	A read-only status bit. If set the MIB block is not currently updating any MIB counters.
29–0 -	Reserved.

28.5.12 Receive control register (FEC_RCR)

The FEC_RCR is programmed by the user, and controls the operational mode of the receive block. It can only be written to when FEC_ECR[ETHER_EN] = 0 (that is, during initialization).

Address: FEC_RCR is 63FE_C000h base + 84h offset = 63FE_C084h



FEC_RCR field descriptions

Field	Description
31–27 -	Reserved, read as 0
26–16 MAX_FL	Maximum frame length. Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive Frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor. The recommended default value to be programmed by the user is 1518 or 1522 (if VLAN Tags are supported).
15–6 -	Reserved, read as 0
5 FCE	Flow control enable. When FCE is set to 1, the receiver detects pause frames. Upon pause frame detection, the transmitter stops transmitting data frames for a given duration.
4 BC_REJ	Broadcast frame reject. When BC_REJ is set to 1, frames with DA (destination address) = 0xFF_FF_FF_FF_FF_FF are rejected unless the PROM bit is set to 1. If both BC_REJ and PROM are set to 1, then frames with broadcast DA is accepted and the M (MISS) bit is set in the receive buffer descriptor.
3 PROM	Promiscuous mode. All frames are accepted regardless of address matching.
2 MII_MODE	Media independent interface mode. Selects external interface mode. Setting this bit to one selects MII mode, setting this bit equal to zero selects 7-wire mode (used only for serial 10 Mbps). This bit controls the interface mode for both transmit and receive blocks.
1 DRT	Disable receive on transmit.

Table continues on the next page...

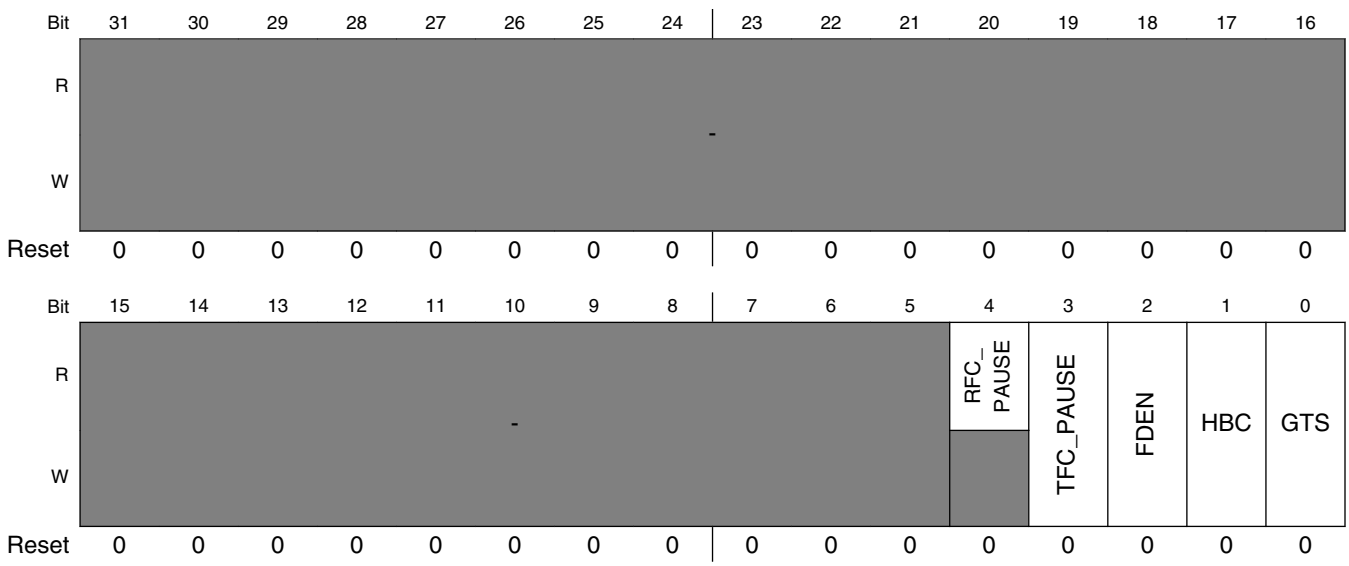
FEC_RCR field descriptions (continued)

Field	Description
	0 Receive path operates independently of transmit (use for full-duplex or to monitor transmit activity in half-duplex mode). 1 Disable reception of frames while transmitting (normally used for half-duplex mode).
0 LOOP	Internal loopback.When LOOP is set to 1, transmitted frames are looped back internal to the device and the transmit output signals are not asserted. The system clock is substituted for the FEC_TX_CLK when LOOP is set to 1. DRT must be set to zero when setting LOOP to 1.

28.5.13 Transmit control register (FEC_TCR)

This register is read/write, and is written by the user to configure the transmit block. Bits [2:1] must only be modified when FEC_ECR[ETHER_EN] = 0 (that is, during initialization). This register is cleared at system reset.

Address: FEC_TCR is 63FE_C000h base + C4h offset = 63FE_C0C4h



FEC_TCR field descriptions

Field	Description
31–5 -	Reserved read as 0
4 RFC_PAUSE	Receive frame control pause. This read-only status bit is set to 1 when a full-duplex flow control pause frame has been received and the transmitter is paused for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete. 0 Transmitter is not paused 1 Transmitter is paused after reception of full-duplex flow control pause frame
3 TFC_PAUSE	Transmit frame control pause. When this bit is set to 1, a pause frame is transmitted according to the following steps:

Table continues on the next page...

FEC_TCR field descriptions (continued)

Field	Description
	1. FEC stops transmission of data frames after the current transmission is complete. 2. The GRA interrupt in the FEC_EIR register is asserted. 3. With transmission of data frames stopped, the FEC transmits a MAC control pause frame. 4. The FEC clears the TFC_PAUSE bit and resume transmitting data frames. The FEC can still transmit a MAC control pause frame when the transmitter is paused due to user assertion of GTS or reception of a pause frame. 0 No pause frame is transmitted 1 Pause frame is transmitted
2 FDEN	Full duplex enable. When FDEN is set to 1, frames are transmitted independent of carrier sense and collision inputs. This bit must only be modified when ETHER_EN is cleared. 0 Full duplex is not enabled 1 Full duplex is enabled
1 HBC	Heartbeat control. When HBC is set to 1, the heartbeat check is performed after end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This bit must only be modified when ETHER_EN is cleared. 0 Heartbeat check is not performed after end of transmission 1 Heartbeat check is performed after end of transmission
0 GTS	Graceful transmit stop. When GTS is set to 1, the FEC stops transmission after any frame that is currently being transmitted is complete and the GRA interrupt in the FEC_EIR register is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission has completed, a "restart" can be accomplished by clearing the GTS bit. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS = 1, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There can be old frames in the transmit FIFO that is transmitted when GTS is reasserted. To avoid this, clear FEC_ECR[ETHER_EN] following the GRA interrupt. 0 Graceful transmit stop is not enabled 1 Graceful transmit stop is enabled.

28.5.14 Physical address low register (FEC_PALR)

The FEC_PALR is written by the user, and contains the lower 32 bits (bytes 0,1,2,3) of the 48-bit address used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. This register is also used for bytes 0 through 3 of the 6-byte source address field when transmitting pause frames. This register is unaffected by reset and must be initialized by the user.

Address: FEC_PALR is 63FE_C000h base + E4h offset = 63FE_C0E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_PALR field descriptions

Field	Description
31–0 PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8) and 3 (bits 7:0) of the 6-byte individual address to be used for exact match, and the source address field in pause frames.

28.5.15 Physical address upper register (FEC_PAUR)

The FEC_PAUR is written by the user, and contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte source address field when transmitting pause frames. Bits 15:0 of FEC_PAUR contain a constant type field (0x8808) used for transmission of pause frames. This register is unaffected by reset, and bits 31:16 must be initialized by the user.

Address: FEC_PAUR is 63FE_C000h base + E8h offset = 63FE_C0E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PADDR2																TYPE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

FEC_PAUR field descriptions

Field	Description
31–16 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address to be used for exact match, and the source address field in pause frames.
15–0 TYPE	Type field in pause frames. This field has a constant value of 0x8808.

28.5.16 Opcode and pause duration register (FEC_OPDR)

FEC_OPDR contains the 16-bit Opcode, and 16-bit pause duration fields used in transmission of a pause frame. The Opcode field is a constant value, 0x0001. When another node detects a pause frame, that node pauses transmission for the duration specified in the pause duration field. This register is not reset and must be initialized by the user.

Address: FEC_OPDR is 63FE_C000h base + ECh offset = 63FE_C0ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OPCODE																PAUSE_DUR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_OPDR field descriptions

Field	Description
31–16 OPCODE	Opcode field used in pause frames. These bits are a constant, 0x0001.
15–0 PAUSE_DUR	Pause duration field used in pause frames.

28.5.17 Descriptor individual address upper register (FEC_IAUR)

The FEC_IAUR is written by the user, and contains the upper 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. This register is unaffected by reset, and must be initialized by the user.

Address: FEC_IAUR is 63FE_C000h base + 118h offset = 63FE_C118h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IADDR1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_IAUR field descriptions

Field	Description
31–0 IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

28.5.18 Descriptor individual address lower register (FEC_IALR)

The FEC_IALR is written by the user, and contains the lower 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is unaffected by reset, and must be initialized by the user.

Address: FEC_IALR is 63FE_C000h base + 11Ch offset = 63FE_C11Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_IALR field descriptions

Field	Description
31–0 IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

28.5.19 Descriptor group address upper register (FEC_GAUR)

The FEC_GAUR is written by the user, and contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

Address: FEC_GAUR is 63FE_C000h base + 120h offset = 63FE_C120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_GAUR field descriptions

Field	Description
31–0 GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

28.5.20 Descriptor group address lower register (FEC_GALR)

The FEC_GALR is written by the user, and contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

Address: FEC_GALR is 63FE_C000h base + 124h offset = 63FE_C124h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GADDR2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_GALR field descriptions

Field	Description
31–0 GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

28.5.21 Transmit FIFO watermark register (FEC_TFWR)

The FEC_TFWR is programmed by the user to control the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows the user to minimize transmit latency (FEC_TFWR[1:0] = 0n) or allow for larger bus access latency (FEC_TFWR[1:0] = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. In some use cases the byte counts associated with the FEC_TFWR field need to be modified to match system requirements, such as the worst-case bus access latency by the transmit data DMA channel.

Address: FEC_TFWR is 63FE_C000h base + 144h offset = 63FE_C144h

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

FEC_TFWR field descriptions

Field	Description
31–2 -	Reserved, read as 0

Table continues on the next page...

FEC_TFWR field descriptions (continued)

Field	Description
1–0 X_WMRK	Number of bytes written to transmit FIFO before transmission of a frame begins 0x 64 bytes written 10 128 bytes written 11 192 bytes written

28.5.22 FIFO receive bound register (FEC_FRBR)

The FEC_FRBR register can be read to determine the upper address bound of the FIFO RAM. Drivers can use this value, along with the FEC_FRSR to appropriately divide the available FIFO RAM between the transmit and receive data paths.

Address: FEC_FRBR is 63FE_C000h base + 14Ch offset = 63FE_C14Ch

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

FEC_FRBR field descriptions

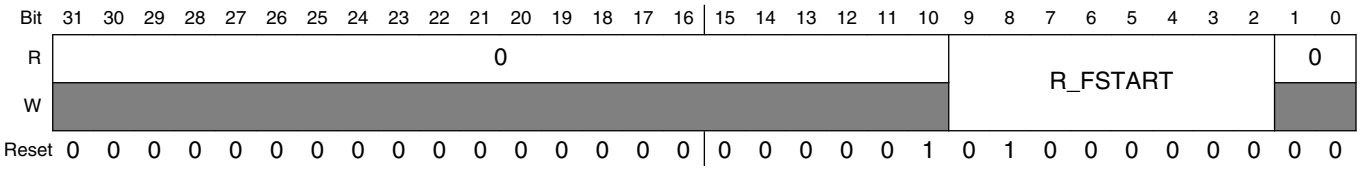
Field	Description
31–10 Reserved	This read-only field is reserved and always has the value zero. Reserved, read as 0 (except bit 10, which is read as 1).
9–2 R_BOUND	Read-only. Highest valid FIFO RAM address.
1–0 Reserved	This read-only field is reserved and always has the value zero. Reserved, read as 0.

28.5.23 FIFO receive FIFO start registers (FEC_FRSR)

FEC_FRSR is an 8-bit register programmed by the user to indicate the starting address of the receive FIFO. FEC_FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FEC_FRSR. The receive FIFO uses the addresses from FEC_FRSR to FEC_FRBR inclusive.

The default value of the receive FIFO starting address is 0x40. This is the value assigned by hardware at reset.

Address: FEC_FRSR is 63FE_C000h base + 150h offset = 63FE_C150h



FEC_FRSR field descriptions

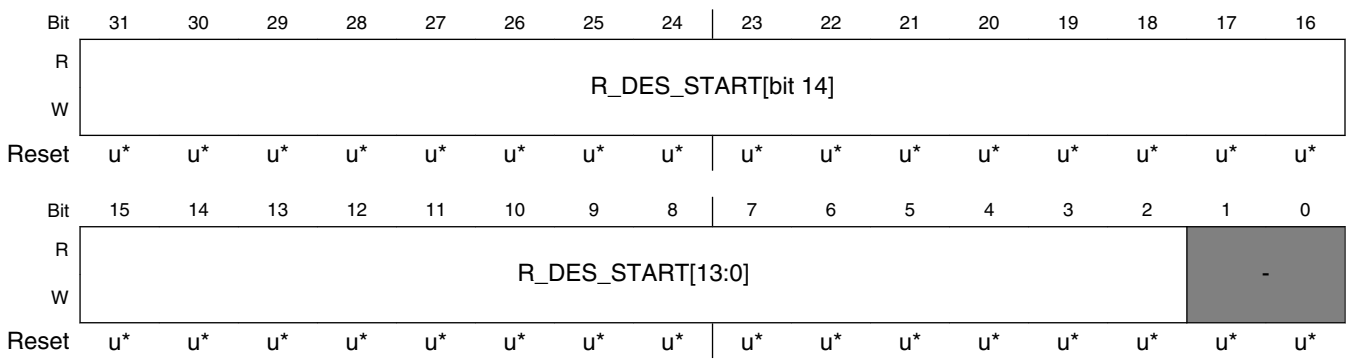
Field	Description
31–10 Reserved	This read-only field is reserved and always has the value zero. Reserved, read as 0 (except bit 10, which is read as 1).
9–2 R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs.
1–0 Reserved	This read-only field is reserved and always has the value zero. Reserved, read as 0.

28.5.24 Receive buffer descriptor ring start register (FEC_ERDSR)

The register is written by the user, and provides a pointer to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 128-bit aligned (that is, evenly divisible by 16).

This register is unaffected by reset and must be initialized by the user prior to operation.

Address: FEC_ERDSR is 63FE_C000h base + 180h offset = 63FE_C180h



* Notes:

- u = Unaffected by reset.

FEC_ERDSR field descriptions

Field	Description
31–2 R_DES_START	Pointer to start of receive buffer descriptor queue.
1–0 -	Reserved, read as 0

28.5.25 Transmit buffer descriptor ring start register (FEC_ETDSR)

The register is written by the user, and provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 128-bit aligned (that is, evenly divisible by 16).

This register is unaffected by reset and must be initialized by the user prior to operation.

Address: FEC_ETDSR is 63FE_C000h base + 184h offset = 63FE_C184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	X_DES_START[bit 14]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	X_DES_START[13:0]														-	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_ETDSR field descriptions

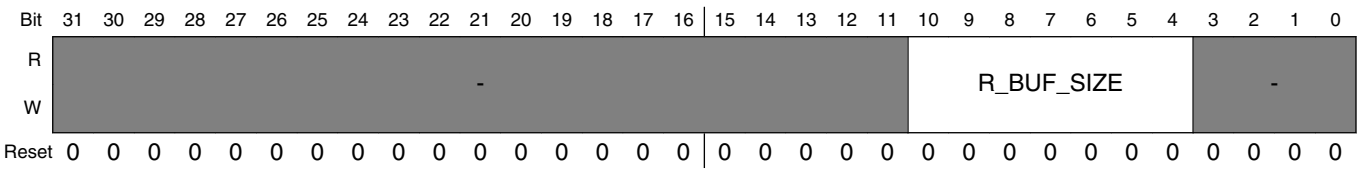
Field	Description
31–2 X_DES_START	Pointer to start of transmit buffer descriptor queue.
1–0 -	Reserved, read as 0

28.5.26 Maximum receive buffer size register (FEC_EMRBR)

The FEC_EMRBR is a user-programmable register which dictates the maximum size of all receive buffers. Note that because receive frames is truncated at 2k-1(2047) bytes, bits 31-11 are not used. The programmed value accounts for the fact that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, FEC_EMRBR must be set to FEC_RCR[MAX_FL] or larger. The FEC_EMRBR must be evenly divisible by 16. To ensure this, bits 3-0 are forced low, and hence only bits 10-4 are actually used. To minimize bus utilization (descriptor fetches) it is recommended that FEC_EMRBR be greater than or equal to 256 bytes.

The FEC_EMRBR is unaffected by reset, and must be initialized by the user.

Address: FEC_EMRBR is 63FE_C000h base + 188h offset = 63FE_C188h



FEC_EMRBR field descriptions

Field	Description
31–11 -	Reserved, is written to 0 by the host processor.
10–4 R_BUF_SIZE	Receive buffer size.
3–0 -	Reserved, is written to 0 by the host processor.

Chapter 29
General Power Controller (GPC)

29.1 Introduction

GPC is a General Power Control module.

29.1.1 Overview

GPC block includes the following sub-blocks of Advanced Power Saving techniques:

- Two DVFS sub-blocks (ARM platform and peripherals clock/power domains)
- SRPG sub-block for SRPG-ed module: ARM platform

Each of the sub-blocks has its own IP registers. The GPC also includes an arbitration block of DVFS voltage/frequency updates.

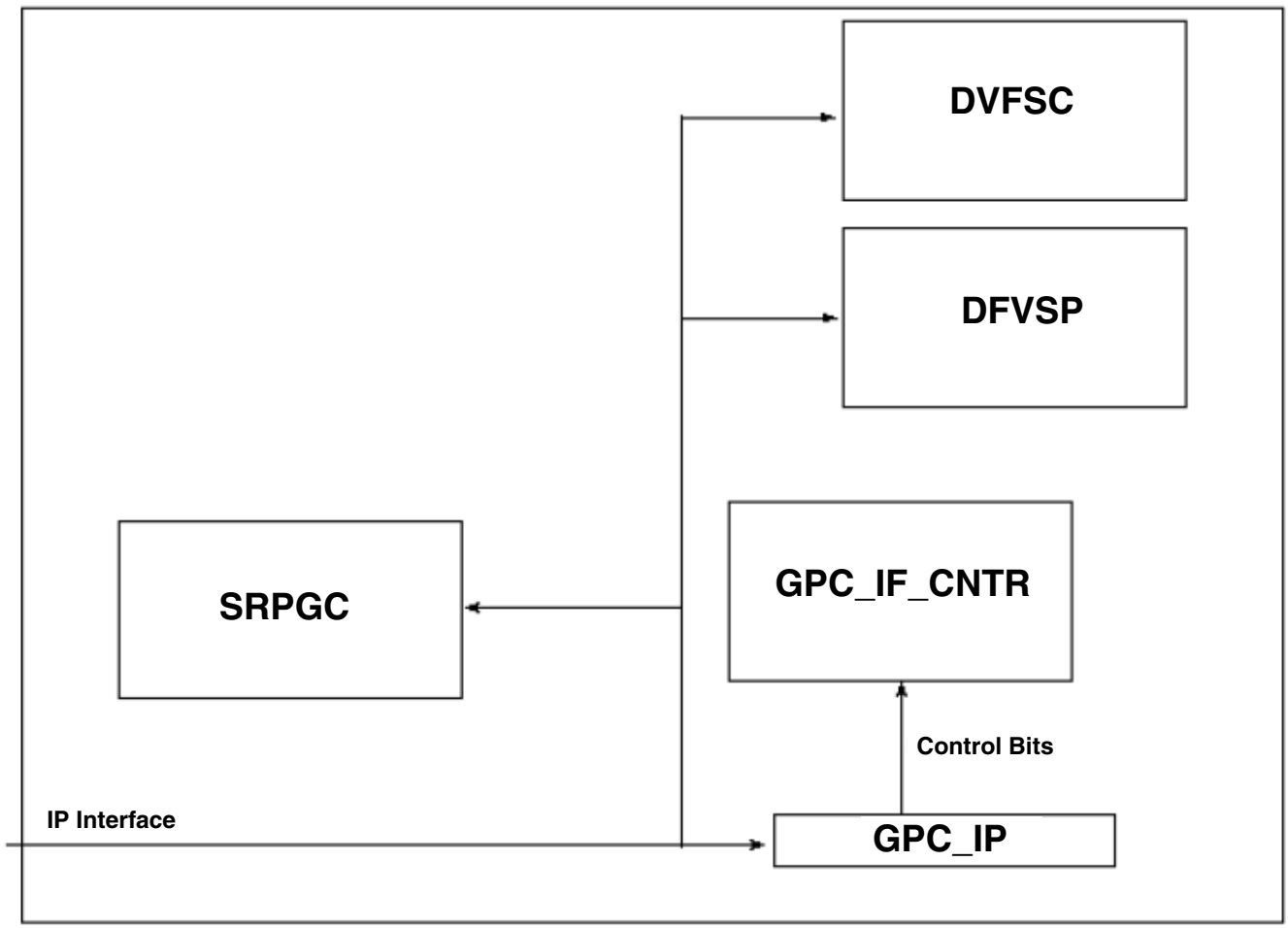


Figure 29-1. GPC Block Diagram

29.1.2 Features

- Simultaneous usage of DVFS of each domain.

- Internal counter for bypass of voltage_ready signal

- Possibility of switching between Configurable SPI (CSPI) usage and I2C Interface (I2C) .

29.2 Functional Description

GPC block includes the following sub-blocks of Advanced Power Saving techniques:

- 2 DVFS sub-blocks (ARM platform and peripherals clock/power domains)
- SRPG sub-block for SRPG-ed module: ARM platform

Each of sub-blocks has its own IP registers. As well, GPC includes an arbitration block of DVFS voltage/frequency updates.

In the following sections the global operation of the sub-blocks is described. For further details of DVFS and SRPG refer appropriate sub-block Block Guide document.

29.2.1 DVFS - Dynamic Voltage & Frequency Scaling

Dynamic Voltage & Frequency Scaling is a well-known technique to reduce power consumption in mobile devices.

In order to improve power saving efficiency, DVFS is applied both on ARM platform domain (DVFSC) and peripherals domain (DVFSP). DVFSC uses mainly ARM's IDLE signal for load monitoring, but also uses s/w writable info as secondary inputs. DVFSP uses hardware accelerators activity signals, bus activity signals and memory activity indicators for load monitoring. Both DVFS load tracking controllers are h/w implemented, providing: high resolution of load tracking (for higher efficiency), no MIPS requirements from ARM platform for continuous operation (more power saving, no additional ARM platform load).

Voltage update requests (to PMIC, via CSPI) may be served in one of the following ways:

- Hardware triggering of CSPI, by GPC
- ARM interrupts serving CSPI

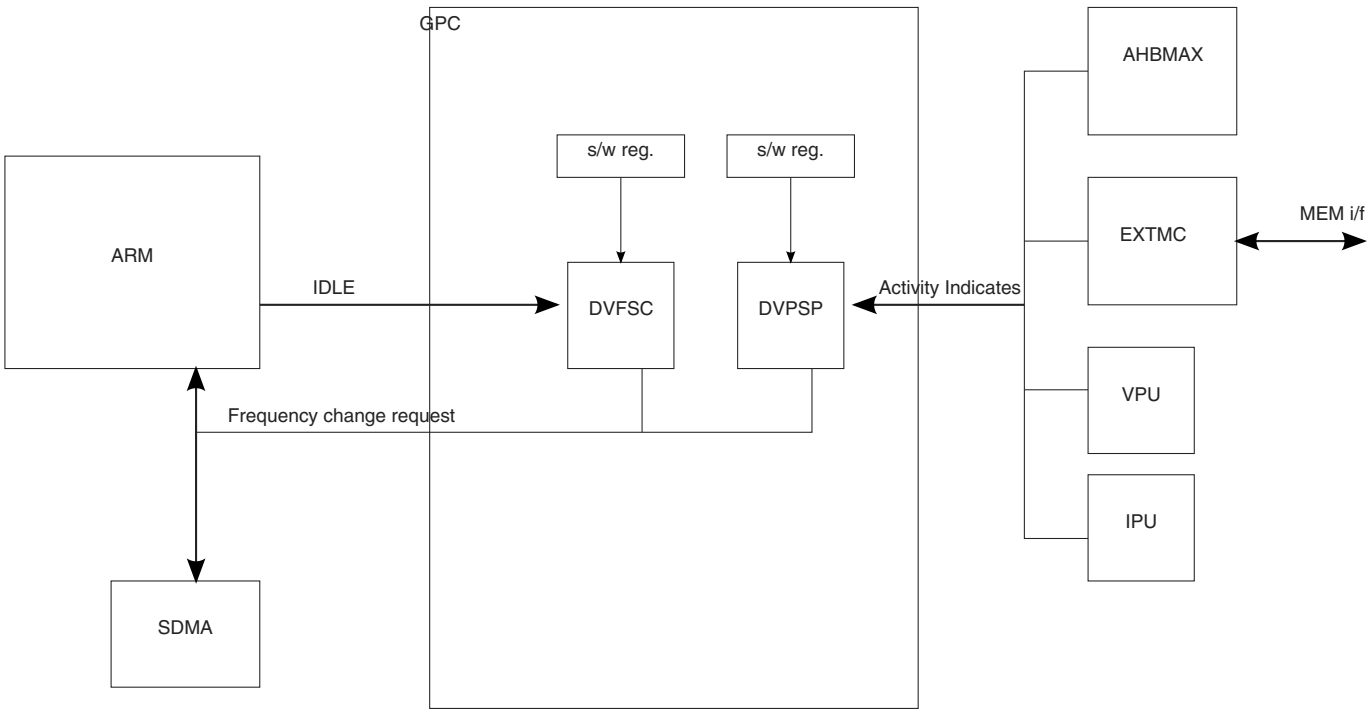


Figure 29-2. DVFS System High Level Diagram

29.2.2 DVFS Change Request Sequence Diagrams

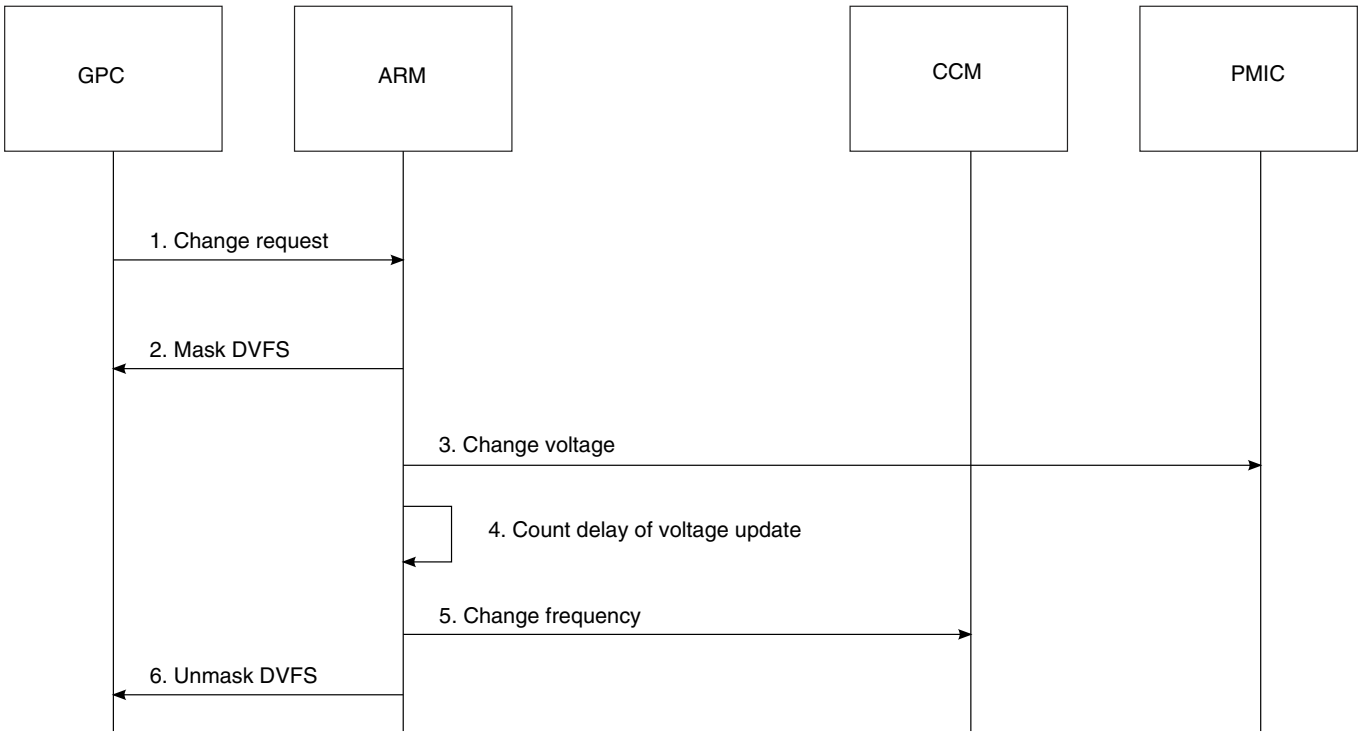


Figure 29-3. DVFS - frequency increase

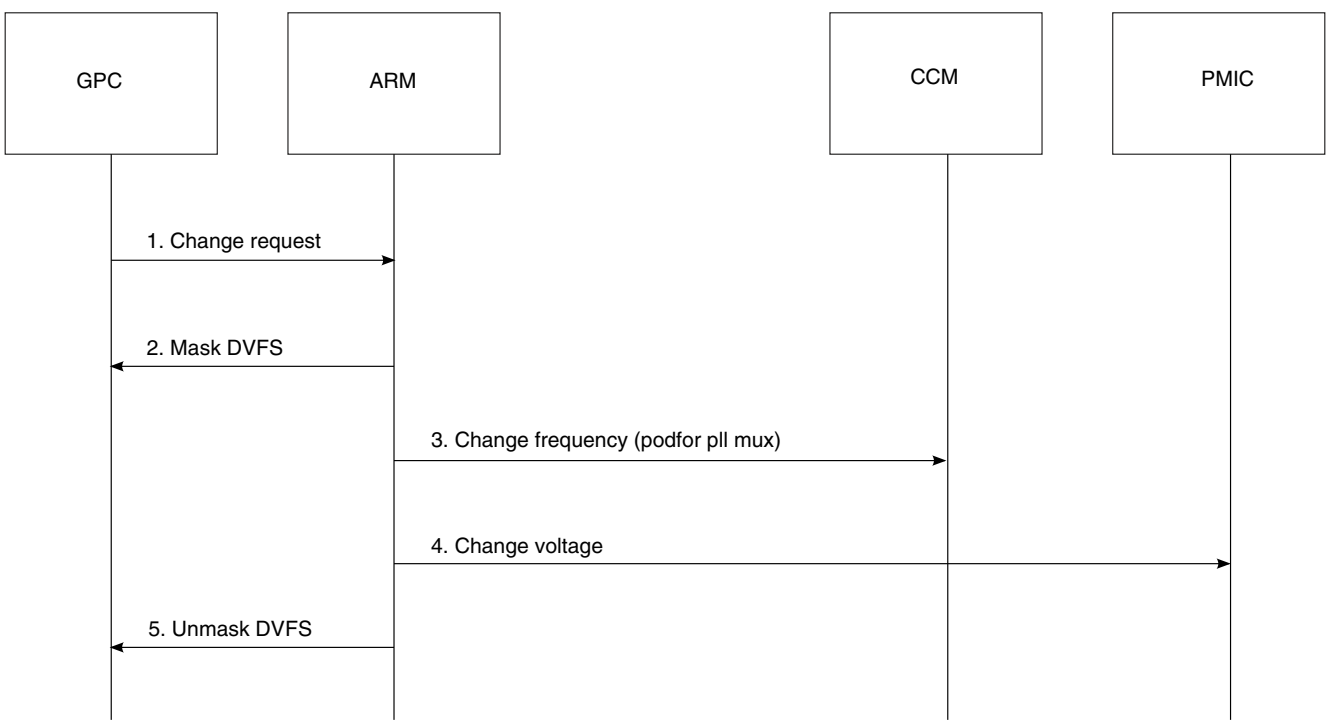


Figure 29-4. DVFS - frequency decrease

DVFSC and DVFSP will have one common interrupt line to ARM and DMA request to SDMA. SDMA (or ARM) will choose only one domain to serve.

DVFSP frequency change will be performed with post-dividers values update only.

DVFSC frequency change can be performed in 2 ways:

- PLL and post-dividers values update
- post-dividers values only update

When PLL update is required, Controller will switch ARM platform clock to another PLL, will wait until the previous PLL is locked on the required frequency, and then will switch the ARM platform clock back to the first PLL.

29.2.3 Frequency / Voltage Change Controller Description

Controller is responsible of correct frequency and voltage update flow.

29.2.3.1 GPC Controller Description

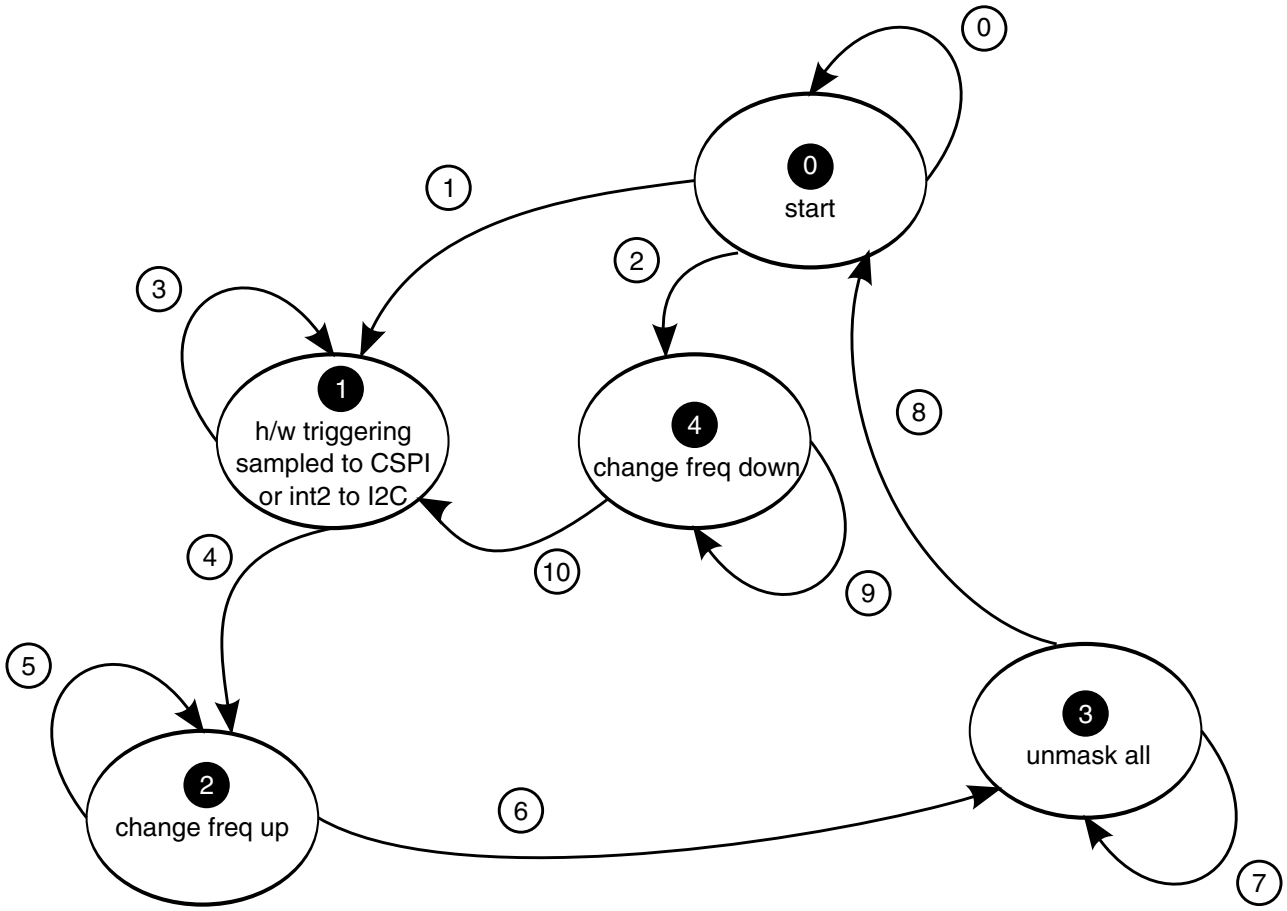


Figure 29-5. GPC DVFS State machine

Table 29-1. GPC Controller Description

Transition	Transition condition	Output
0	not (1)	-
1	STRT bit asserted, and ((freq/voltage increase needed) | (!freq/volt_inc & ! freq_update))	-
2	STRT bit set, and freq/voltage decrease needed and freq_update	-
3	not (4)	Sample h/w triggering selected reg index to CSPI or sending int2 to I2C for PMIC Start counter if VCNTU==1 (pmic ack not needed)
4	(voltage_ready & pmic ack needed) | (counter finished & !pmic ack needed)	-

Table continues on the next page...

Table 29-1. GPC Controller Description (continued)

Transition	Transition condition	Output
5	not (6)	If (dvfs update up), trigger freq. change Reset counter
6	(dvfs update up & clk change ack) | !(dvfs update up)	-
7	1 clk	unmask all, enable all toggle dvfs_cnt_res_arm or dvfs_cnt_res_per
8	not (8)	-
9	not (10)	If (dvfs update dw), trigger freq. change IRQ masked: IRQM=1
10	(dvfs update dw& clk change ack) | !(dvfs update dw) | !freq_update	-

29.2.3.2 Clock Control Module Frequency Update Controller Description

Additional State Machine is required in CCM to take care about frequency update sequence, as defined below:

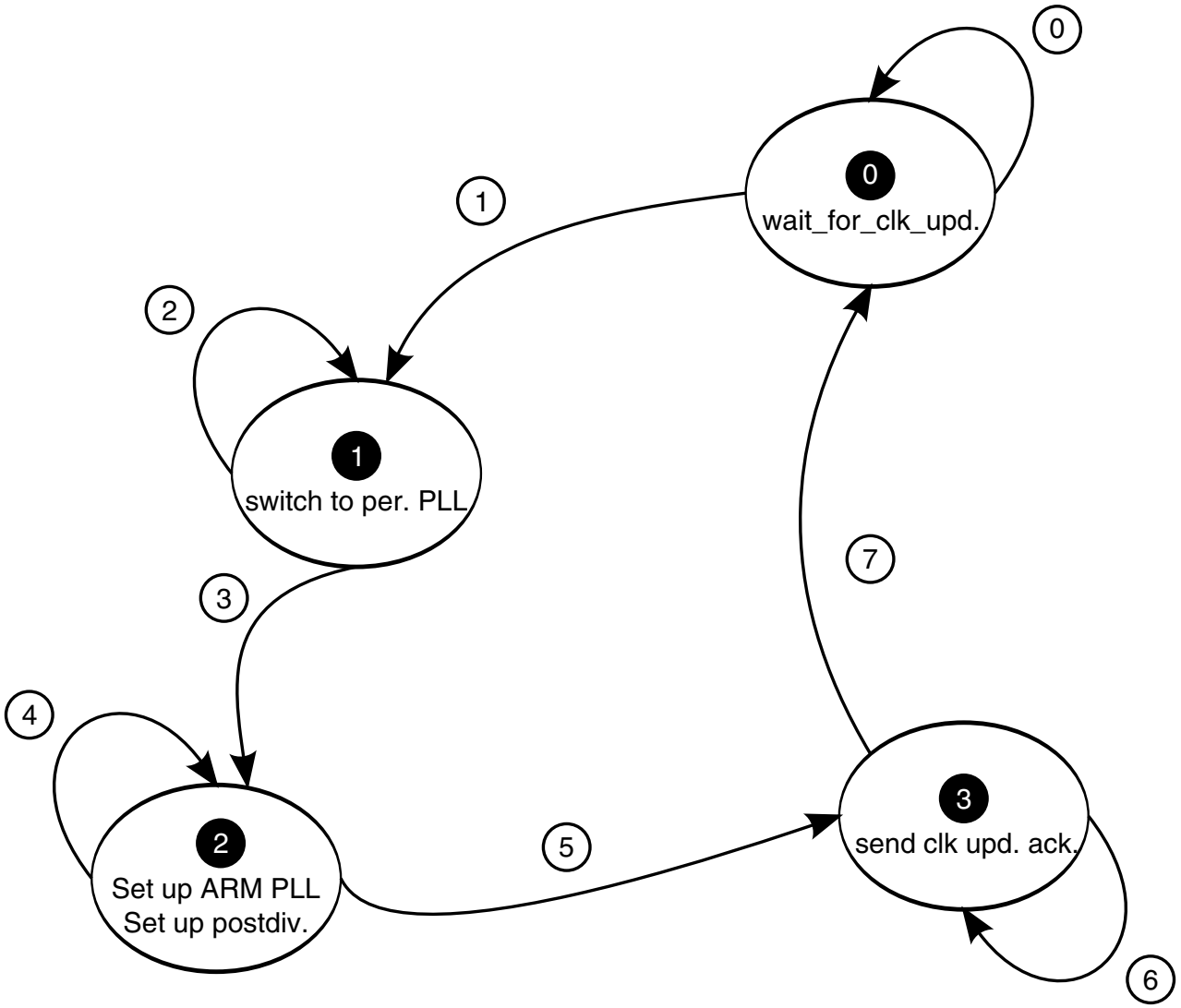


Figure 29-6. CCM Frequency Update Controller

Table 29-2. CCM Frequency Update Controller Description

Transition	Transition condition	Output
0	not (1)	clk upd ack negated
1	clk update request	-
2	not (3)	switch ARM clk to per. PLL
3	clk switched to per. PLL	-
4	not (5)	apply new PLL settings, if needed apply new postdiv settings if needed
5	ARM PLL & postdiv upd ack.	-
6	1 clk	clk update ack to GPC asserted.

Table continues on the next page...

Table 29-2. CCM Frequency Update Controller Description (continued)

Transition	Transition condition	Output
7	not (6)	-

29.2.4 State Retention Power Gating (SRPG)

State Retention Power Gating (SRPG) is an advanced power saving technique - the values of FFs are saved, while the supply for the combinational logic is gated. Such special power gating allows significant power saving during low-power (stand-by) states, when no logical operation is needed. Relatively the simple Power Gating techniques, SRPG allows to restore the state very quickly and continue the processing from the state, sampled before stand-by period.

SRPG Controller manages all the related signals for entering and exiting SRPG state of relevant modules. Not all the modules have SRPG design.

SRPG power down sequence:

CCM sends power down request when the chip enters stop or wait mode. The user should define which modules will be powered down (PGCR registers of corresponding SRPG, bit 0).

SRPG power up sequence:

The modules receive power up request in following order:

NEON and CTA8.

29.2.5 PMIC Interface Requirements for APM Support

PMIC APM interface is done through SPI protocol by enhanced CSPI module with HW triggering. Below is the description of modules requirements for APM support.

- CSPI
  - Will have 16 HW trigger registers.
  - Each register will be 32 bits:
    - 1 bits read/write (always write)
    - 6 bits address
    - 1 bit not used
    - 24 bits data
  - Each register will have an associated signal that starts the transaction to Atlas

- GPC
  - will have a 4-bit select register to select one of 16 HW trigger signals, if CSPI is used.
  - will trigger the signals during DVFS state machine according to select register value.
- PMIC
  - Only one 24-bit register should be updated for voltage change. One CSPI HW trigger will be able to make the change, or I2C will get int2.

Ability to use SDMA for h/w controllers inputs analysis and/or CSPI programming will provide high level of s/w flexibility and will be kept for backup.

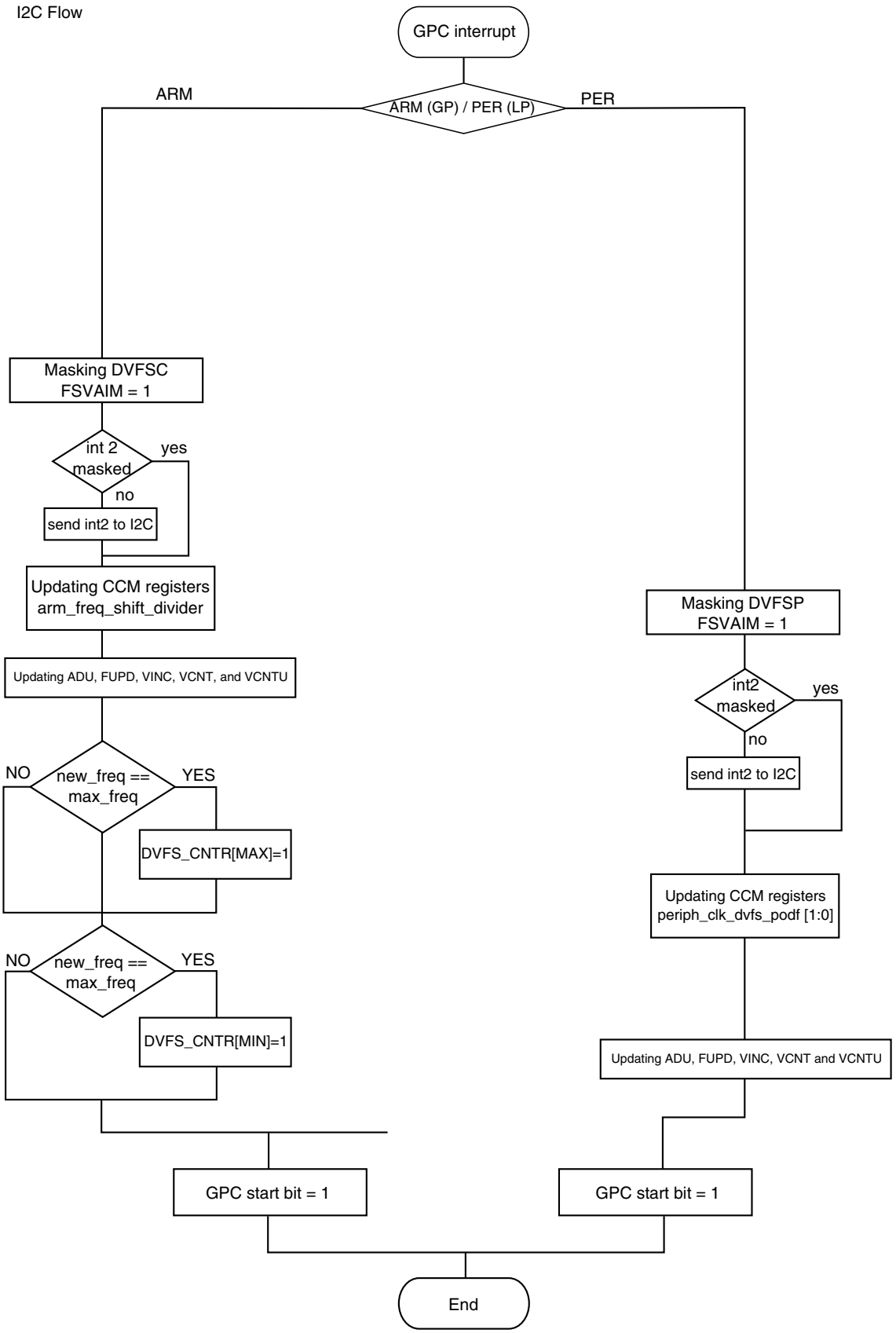


Figure 29-7. PMIC Interface Requirements IPC Flow

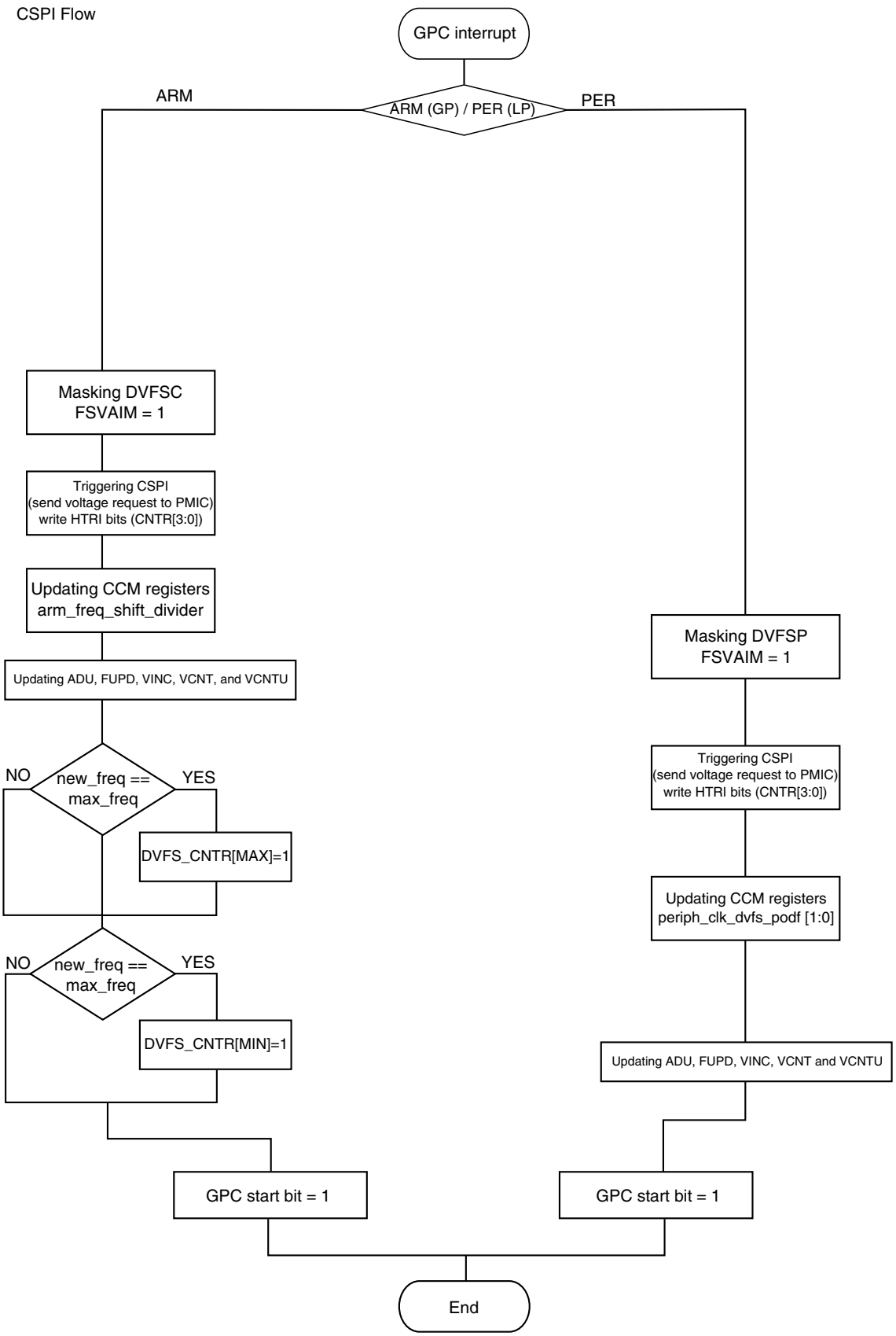


Figure 29-8. PMIC Interface Requirements CSPI Flow

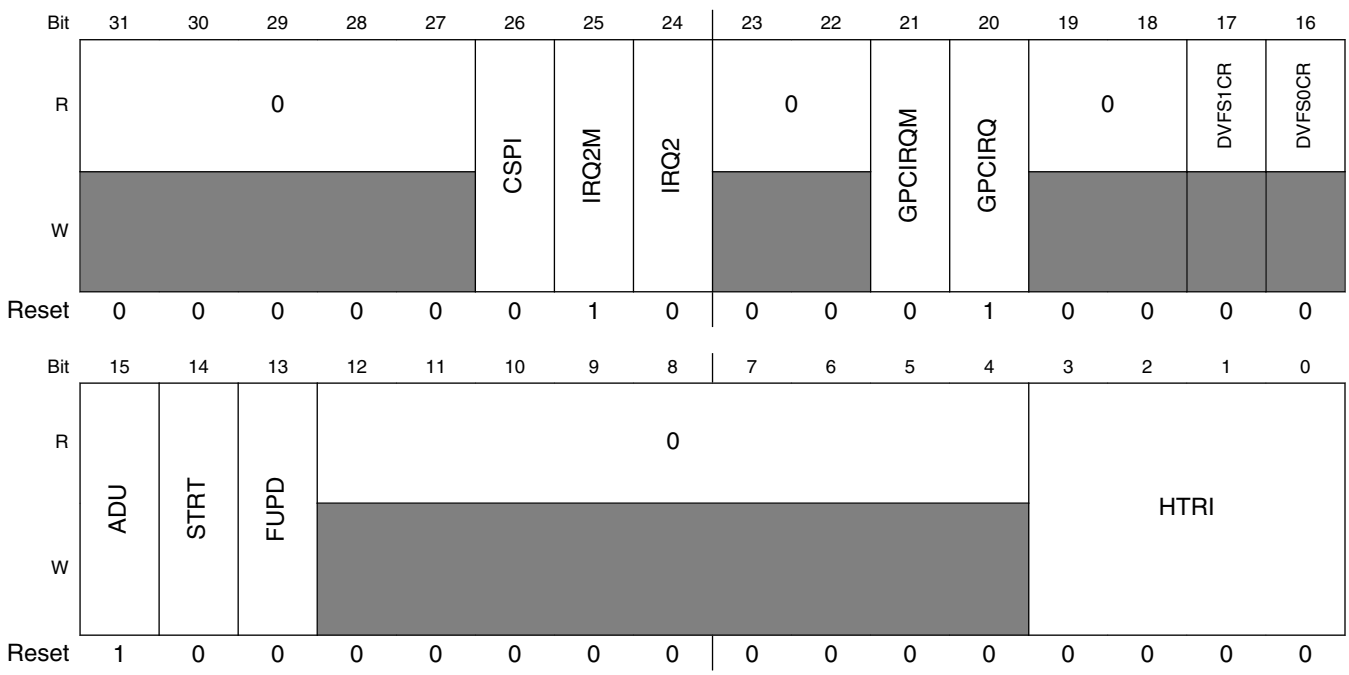
29.3 Programmable Registers

GPC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FD_8000	Interface Control Register (GPC_CNTR)	32	R/W	0210_8000h	29.3.1/ 1530
53FD_8008	Voltage Counter Register (GPC_VCR)	32	R/W	0000_0001h	29.3.2/ 1532
53FD_8010	NEON register (GPC_NEON)	32	R/W	0000_0030h	29.3.3/ 1533

29.3.1 Interface Control Register (GPC_CNTR)

Address: GPC_CNTR is 53FD_8000h base + 0h offset = 53FD_8000h



GPC_CNTR field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

GPC_CNTR field descriptions (continued)

Field	Description
26 CSPI	CSPI or I2C is used (hw trigger will be generated or irq2 will be sent for PMIC) 0 default - I2C is in use 1 CSPI is in use
25 IRQ2M	int2 (for I2C) masking 0 int2 will be sent by GPC FSM if CSPI is '0' (I2C is in use) 1 default - int2 is masked, but GPC FSM will wait for INT2 bit to be negated by writing '1', if CSPI is '0'.
24 IRQ2	Status bit, write 1 to clear.
23–22 Reserved	This read-only field is reserved and always has the value zero. Reserved
21 GPCIRQM	GPC interrupt/event masking 1 interrupt/event is masked 0 not masked
20 GPCIRQ	GPC will generate ARM IRQ or Smart Direct Memory Access (SDMA) event, as a result of DVFS change requests 1 GPC will generate ARM IRQ 0 GPC will generate SDMA event
19–18 Reserved	This read-only field is reserved and always has the value zero. Reserved
17 DVFS1CR	DVFSP Change request (bit is read-only) 1 DVFS1 is requesting for frequency/voltage update 0 DVFS1 has no request
16 DVFS0CR	DVFSC Change request (bit is read-only) 1 DVFS0 is requesting for frequency/voltage update 0 DVFS0 has no request
15 ADU	ARM domain freq/voltage update needed 1 ARM domain frequency and/or voltage update needed 0 PER domain frequency and/or voltage update needed
14 STRT	Controller start Controller operation will be started when bit set to "1". Bit will be set automatically to "0" when frequency / voltage change finished. There is still a possibility to write "0" by s/w. 1 Controller operation in progress. No IRQ is allowed during voltage update procedure (IRQ will be masked) 0 Controller operation finished. New freq/voltage change request is available
13 FUPD	Frequency update needed 1 frequency update needed 0 frequency updated is not needed

Table continues on the next page...

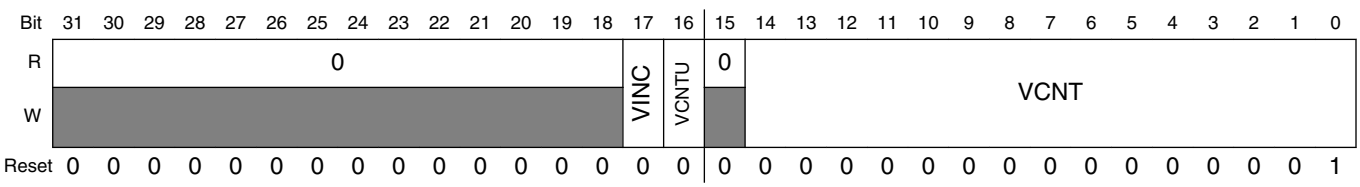
GPC_CNTR field descriptions (continued)

Field	Description
12–4 Reserved	This read-only field is reserved and always has the value zero. Reserved
3–0 HTRI	Hardware Triggering Register Index Indicates which one of the CSPI h/t register will be written to the PMIC. 0000 register #0 will be triggered, 0001 register #1 will be triggered, 0010 register #3 will be triggered...

29.3.2 Voltage Counter Register (GPC_VCR)

VCR Register - Voltage Counter Register

Address: GPC_VCR is 53FD_8000h base + 8h offset = 53FD_8008h



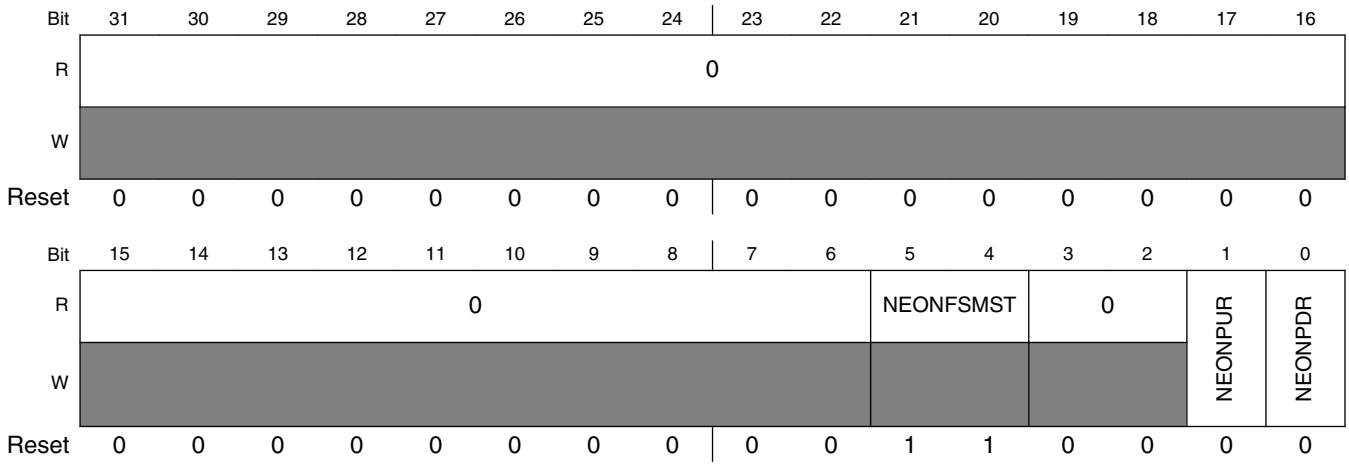
GPC_VCR field descriptions

Field	Description
31–18 Reserved	This read-only field is reserved and always has the value zero. Reserved
17 VINC	Voltage increase 1 Voltage will be increased 0 Voltage will be decreased
16 VCNTU	Voltage Count Used 1 VCNT is used 0 VCNT is not used, "voltage ready" signal of PMIC will be used.
15 Reserved	This read-only field is reserved and always has the value zero. Reserved
14–0 VCNT	Voltage update Count Counting delay of the voltage update, if PMIC's "voltage ready" signal is not used Counter is using SYS_CLK (CKIH) clock

29.3.3 NEON register (GPC_NEON)

NEON Register - Register Control Power for NEON

Address: GPC_NEON is 53FD_8000h base + 10h offset = 53FD_8010h



GPC_NEON field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5–4 NEONFSMST	Neon FSM status bits - state of Neon State Machine: 11 Power up acknowledge / Normal state 01 Power down request 00 Power down acknowledge 10 Power up request
3–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 NEONPUR	NEON Power Up Request. Assertion causes Neon PowerDown state machine to change state from PowerDown to PowerUp request. Will clear itself after a single clock.
0 NEONPDR	NEON Power Down Request. Assertion starts Neon PowerDown state machine. Will clear itself after a single clock.

Chapter 30
General Purpose Input/Output (GPIO)

30.1 Overview

The general purpose input/output (GPIO) module generates up to 32 signals for general-purpose. A block diagram of the GPIO is shown in Figure 30-1.

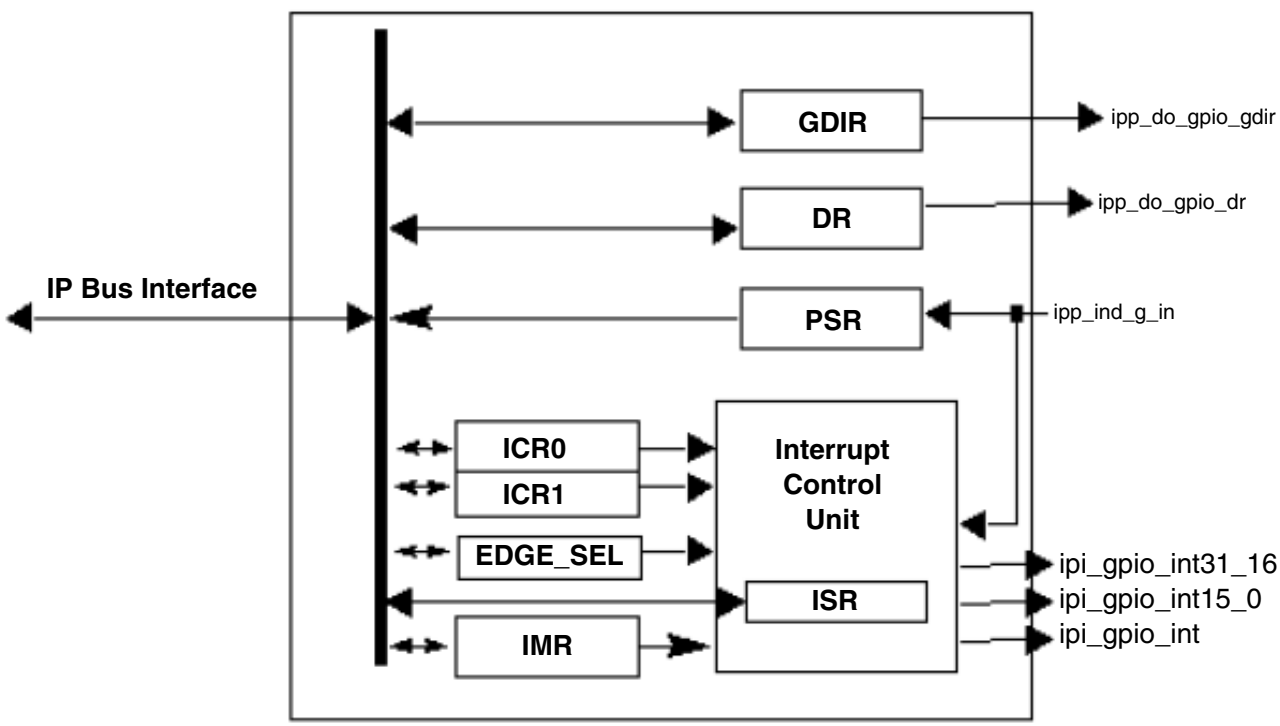


Figure 30-1. GPIO Block Diagram

The GPIO general-purpose input/output peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an output, you can write to an internal register to control the state driven on the output pin. When configured as an input, you can detect the state of the input by reading the state of an internal register.

In addition, the GPIO peripheral can produce CORE interrupts.

The GPIO module is one of the modules controlling the IOMUX of the SoC (System on a Chip). Figure 30-2. shows the SoC muxing scheme.

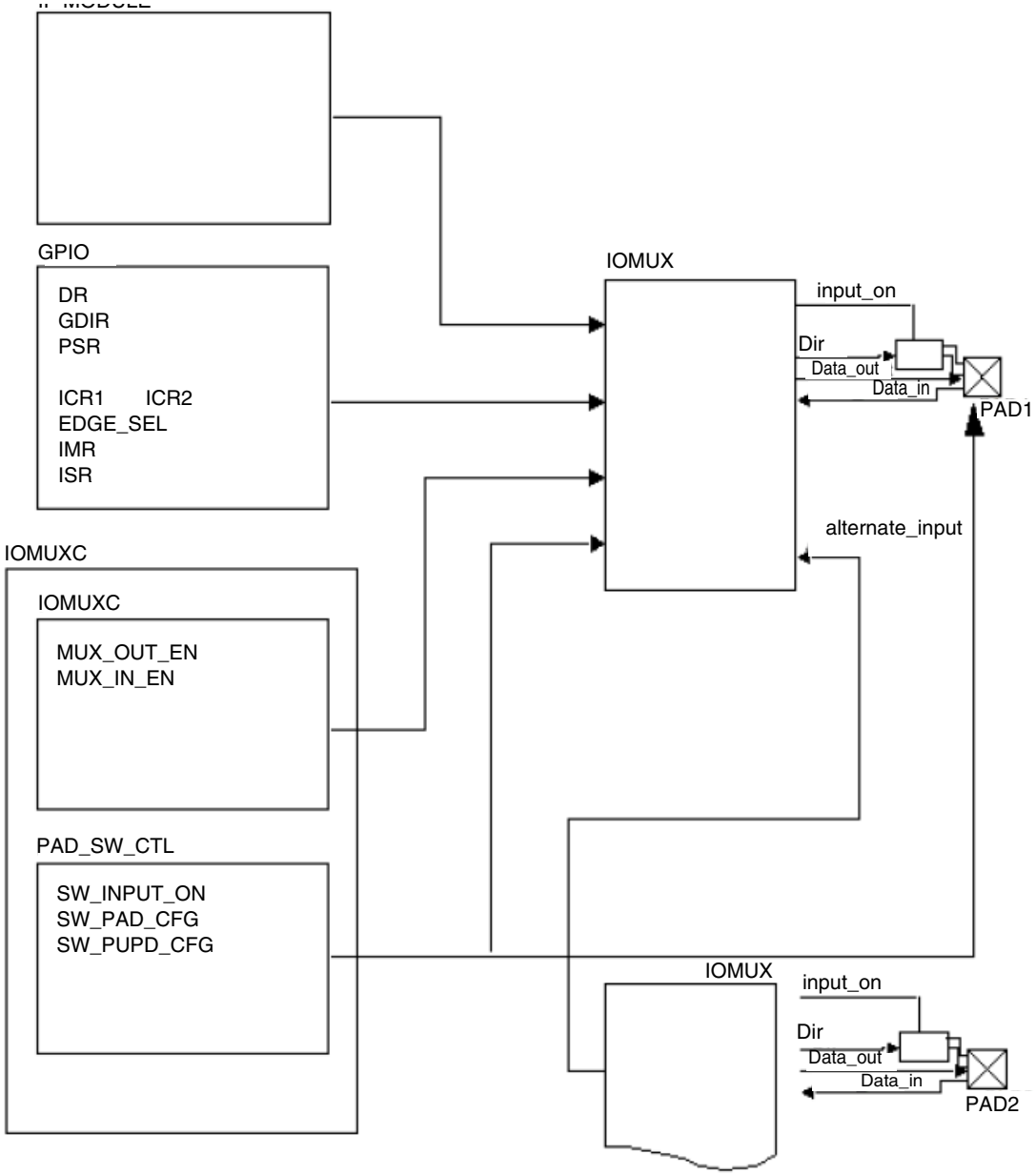


Figure 30-2. SoC IOMUX Scheme

The GPIO functionality is provided through eight registers, an edge-detect circuit, and interrupt generation logic.

The eight registers are:

- Data register (DR)
- GPIO direction register (GDIR)

- Pad sample register (PSR)
- Interrupt control registers (ICR1, ICR2)
- Edge select register (EDGE_SEL)
- Interrupt mask register (IMR)
- Interrupt status register (ISR)

These registers are described in detail in the Programmable Registers section.

Each GPIO input has a dedicated edge-detect circuit which can be configured through software to detect rising edges, falling edges, logic low-levels or logic high-levels on the input signals. The outputs of the edge detect circuits are optionally masked by setting the corresponding bit in the interrupt mask register (IMR). These qualified outputs are OR'ed together to generate three one-bit interrupt lines:

- ini_gpio_int31_16: One-bit interrupt OR of 16 high interrupts.
- ipi_gpio_int15_0: One-bit interrupt OR of 16 low interrupts.
- ipi_gpio_int: One-bit interrupt OR of 32 interrupts.

In addition, the 32-bit signal ipi_gpio_int32 gives the user access to all 32 individual interrupts.

The GPIO edge detection is described further in Interrupt Control Unit.

The GPIO's overall functionality is described further in GPIO Functional Description.

30.1.1 Features

The GPIO includes the following features:

- General purpose input/output logic capabilities:
  - Drives specific data to output using the data register (DR)
  - Controls the direction of the signal using the GPIO direction register (GDR)
  - Enables the core to sample the status of the corresponding inputs by reading the pad sample register (PSR).
- GPIO interrupt capabilities:
  - Supports up to 32 interrupts
  - Identifies interrupt edges
  - Generates three active-high interrupts to the SoC interrupt controller

30.2 GPIO Functional Description

30.2.1 GPIO Function

A GPIO signal can operate as a general-purpose input/output when the IOMUX is set to GPIO mode. Each GPIO signal my be independently configured as either an input or an output using the GPIO direction register (GDIR). When configured as an output (GDIR bit = 1), the value in the data bit in the GPIO data register (DR) is driven on the corresponding GPIO line. When a signal is configured as an input (GDIR bit = 0), the state of the input can be read from the corresponding PSR bit.

30.2.2 GPIO Programming

30.2.2.1 GPIO Read Mode

The programming sequence for reading input signals should be as follows:

1. Configure IOMUX to select GPIO mode (Via IOMUXC).
2. Configure GPIO direction register (GDR) to input.
3. Read value from data register/pad status register.

A pseudocode description to read [input3:input0] values is as follows:

```
write sw_mux_ctl_<input0>_<input1>_<input2>_<input3> , 32'h00000000      // SET INPUTS TO
GPIO MODE.
write GDIR[31:4,input3_bit, input2_bit, input1_bit, input0_bit,] 32'hxxxxxxx0 // SET GDIR TO
INPUT.
read DR                                                          // READ INPUT VALUE FROM DR.
read PSR                                                        // READ INPUT VALUE FROM PSR.
```

NOTE

While the GPIO direction is set to input (GDIR = 0), a read access to DR does not return DR data. Instead, it returns the PSR data, which is the corresponding input signal value.

30.2.2.2 GPIO Write Mode

The programming sequence for driving output signals should be as follows:

1. Configure IOMUX to select GPIO mode (Via IOMUXC).
2. Configure GPIO direction register (GDR) to output.
3. Write value to data register (DR).

A pseudocode description to drive 4'b0101 on [output3:output0] is as follows:

```
write sw_mux_ctl_<output0>_<output1>_<output2>_<output3> , 32'h00000000   // SET OUTPUTS TO
GPIO MODE.
```

```
write GDIR[31:4,output3_bit,output2_bit, output1_bit, output0_bit,] 32'hxxxxxxxF // SET GDIR
TO OUTPUT.
write DR, 32'hxxxxxxx5                                                  // WRITE OUTPUT VALUE
TO DR.
read_cmp PSR, 32'hxxxxxxx5                                              // READ OUTPUT VALUE
FROM PSR ONLY.
```

NOTE

While GPIO direction is set to output, the real internal value can only be verified through the PSR, and not through the DR.

30.2.3 Interrupt Control Unit

In addition to the general-purpose input/output function, the edge-detect logic in the GPIO peripheral reflects whether a transition has occurred on a given GPIO signal that is configured as an input (GDIR bit = 0). The interrupt control registers (ICR1 and ICR2) may be used to independently configure the interrupt condition of each input signal (low-to-high transition, high-to-low transition, low, or high). For information about ICR1 and ICR2 settings, refer to GPIO interrupt configuration register1 (GPIO_GPIO_ICR1) and GPIO interrupt configuration register2 (GPIO_GPIO_ICR2).

The interrupt control unit is built of 32 interrupt control subunits, where each subunit handles a single interrupt line.

30.3 Programmable Registers

There are eight 32-bit GPIO registers. All registers are accessible from the IP interface. Only 32-bit access is supported.

GPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F8_4000	GPIO data register (GPIO-1_GPIO_DR)	32	R/W	0000_0000h	30.3.1/ 1542
53F8_4004	GPIO direction register (GPIO-1_GPIO_GDIR)	32	R/W	0000_0000h	30.3.2/ 1543
53F8_4008	GPIO pad status register (GPIO-1_GPIO_PSR)	32	R	0000_0000h	30.3.3/ 1544
53F8_400C	GPIO interrupt configuration register1 (GPIO-1_GPIO_ICR1)	32	R/W	0000_0000h	30.3.4/ 1544

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F8_4010	GPIO interrupt configuration register2 (GPIO-1_GPIO_ICR2)	32	R/W	0000_0000h	30.3.5/ 1545
53F8_4014	GPIO interrupt mask register (GPIO-1_GPIO_IMR)	32	R/W	0000_0000h	30.3.6/ 1546
53F8_4018	GPIO interrupt status register (GPIO-1_GPIO_ISR)	32	R/W	0000_0000h	30.3.7/ 1547
53F8_401C	GPIO edge select register (GPIO-1_GPIO_EDGE_SEL)	32	R/W	0000_0000h	30.3.8/ 1548
53F8_8000	GPIO data register (GPIO-2_GPIO_DR)	32	R/W	0000_0000h	30.3.1/ 1542
53F8_8004	GPIO direction register (GPIO-2_GPIO_GDIR)	32	R/W	0000_0000h	30.3.2/ 1543
53F8_8008	GPIO pad status register (GPIO-2_GPIO_PSR)	32	R	0000_0000h	30.3.3/ 1544
53F8_800C	GPIO interrupt configuration register1 (GPIO-2_GPIO_ICR1)	32	R/W	0000_0000h	30.3.4/ 1544
53F8_8010	GPIO interrupt configuration register2 (GPIO-2_GPIO_ICR2)	32	R/W	0000_0000h	30.3.5/ 1545
53F8_8014	GPIO interrupt mask register (GPIO-2_GPIO_IMR)	32	R/W	0000_0000h	30.3.6/ 1546
53F8_8018	GPIO interrupt status register (GPIO-2_GPIO_ISR)	32	R/W	0000_0000h	30.3.7/ 1547
53F8_801C	GPIO edge select register (GPIO-2_GPIO_EDGE_SEL)	32	R/W	0000_0000h	30.3.8/ 1548
53F8_C000	GPIO data register (GPIO-3_GPIO_DR)	32	R/W	0000_0000h	30.3.1/ 1542
53F8_C004	GPIO direction register (GPIO-3_GPIO_GDIR)	32	R/W	0000_0000h	30.3.2/ 1543
53F8_C008	GPIO pad status register (GPIO-3_GPIO_PSR)	32	R	0000_0000h	30.3.3/ 1544
53F8_C00C	GPIO interrupt configuration register1 (GPIO-3_GPIO_ICR1)	32	R/W	0000_0000h	30.3.4/ 1544
53F8_C010	GPIO interrupt configuration register2 (GPIO-3_GPIO_ICR2)	32	R/W	0000_0000h	30.3.5/ 1545
53F8_C014	GPIO interrupt mask register (GPIO-3_GPIO_IMR)	32	R/W	0000_0000h	30.3.6/ 1546
53F8_C018	GPIO interrupt status register (GPIO-3_GPIO_ISR)	32	R/W	0000_0000h	30.3.7/ 1547
53F8_C01C	GPIO edge select register (GPIO-3_GPIO_EDGE_SEL)	32	R/W	0000_0000h	30.3.8/ 1548

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F9_0000	GPIO data register (GPIO-4_GPIO_DR)	32	R/W	0000_0000h	30.3.1/ 1542
53F9_0004	GPIO direction register (GPIO-4_GPIO_GDIR)	32	R/W	0000_0000h	30.3.2/ 1543
53F9_0008	GPIO pad status register (GPIO-4_GPIO_PSR)	32	R	0000_0000h	30.3.3/ 1544
53F9_000C	GPIO interrupt configuration register1 (GPIO-4_GPIO_ICR1)	32	R/W	0000_0000h	30.3.4/ 1544
53F9_0010	GPIO interrupt configuration register2 (GPIO-4_GPIO_ICR2)	32	R/W	0000_0000h	30.3.5/ 1545
53F9_0014	GPIO interrupt mask register (GPIO-4_GPIO_IMR)	32	R/W	0000_0000h	30.3.6/ 1546
53F9_0018	GPIO interrupt status register (GPIO-4_GPIO_ISR)	32	R/W	0000_0000h	30.3.7/ 1547
53F9_001C	GPIO edge select register (GPIO-4_GPIO_EDGE_SEL)	32	R/W	0000_0000h	30.3.8/ 1548
53FD_C000	GPIO data register (GPIO-5_GPIO_DR)	32	R/W	0000_0000h	30.3.1/ 1542
53FD_C004	GPIO direction register (GPIO-5_GPIO_GDIR)	32	R/W	0000_0000h	30.3.2/ 1543
53FD_C008	GPIO pad status register (GPIO-5_GPIO_PSR)	32	R	0000_0000h	30.3.3/ 1544
53FD_C00C	GPIO interrupt configuration register1 (GPIO-5_GPIO_ICR1)	32	R/W	0000_0000h	30.3.4/ 1544
53FD_C010	GPIO interrupt configuration register2 (GPIO-5_GPIO_ICR2)	32	R/W	0000_0000h	30.3.5/ 1545
53FD_C014	GPIO interrupt mask register (GPIO-5_GPIO_IMR)	32	R/W	0000_0000h	30.3.6/ 1546
53FD_C018	GPIO interrupt status register (GPIO-5_GPIO_ISR)	32	R/W	0000_0000h	30.3.7/ 1547
53FD_C01C	GPIO edge select register (GPIO-5_GPIO_EDGE_SEL)	32	R/W	0000_0000h	30.3.8/ 1548
53FE_0000	GPIO data register (GPIO-6_GPIO_DR)	32	R/W	0000_0000h	30.3.1/ 1542
53FE_0004	GPIO direction register (GPIO-6_GPIO_GDIR)	32	R/W	0000_0000h	30.3.2/ 1543
53FE_0008	GPIO pad status register (GPIO-6_GPIO_PSR)	32	R	0000_0000h	30.3.3/ 1544
53FE_000C	GPIO interrupt configuration register1 (GPIO-6_GPIO_ICR1)	32	R/W	0000_0000h	30.3.4/ 1544

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FE_0010	GPIO interrupt configuration register2 (GPIO-6_GPIO_ICR2)	32	R/W	0000_0000h	30.3.5/ 1545
53FE_0014	GPIO interrupt mask register (GPIO-6_GPIO_IMR)	32	R/W	0000_0000h	30.3.6/ 1546
53FE_0018	GPIO interrupt status register (GPIO-6_GPIO_ISR)	32	R/W	0000_0000h	30.3.7/ 1547
53FE_001C	GPIO edge select register (GPIO-6_GPIO_EDGE_SEL)	32	R/W	0000_0000h	30.3.8/ 1548

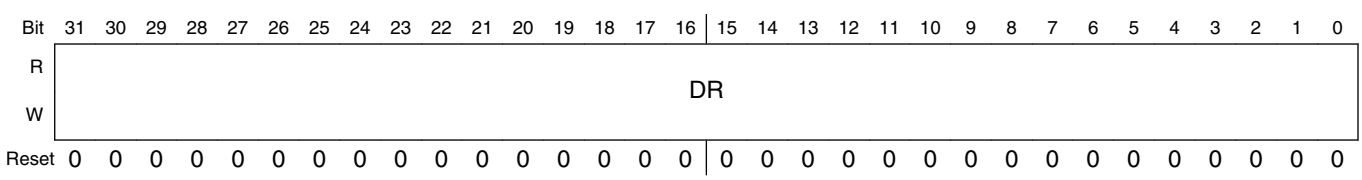
30.3.1 GPIO data register (GPIOx_GPIO_DR)

The 32-bit GPIO DR register stores data that is ready to be driven to the output lines. If the IOMUX is in GPIO mode and a given GPIO direction bit is set, then the corresponding DR bit is driven to the output. If a given GPIO direction bit is cleared, then a read of DR reflects the value of the corresponding signal.Two wait states are required in read access for synchronization.

The results of a read of a DR bit depends on the IOMUX input mode settings and the corresponding GDIR bit as follows:

- If GDIR[*n*] is set and IOMUX input mode is GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and IOMUX input mode is GPIO, then reading DR[*n*] returns the corresponding input signal's value.
- If GDIR[*n*] is set and IOMUX input mode is not GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and IOMUX input mode is not GPIO, then reading DR[*n*] always returns zero.

Addresses: GPIO-1_GPIO_DR is 53F8_4000h base + 0h offset = 53F8_4000h
GPIO-2_GPIO_DR is 53F8_8000h base + 0h offset = 53F8_8000h
GPIO-3_GPIO_DR is 53F8_C000h base + 0h offset = 53F8_C000h
GPIO-4_GPIO_DR is 53F9_0000h base + 0h offset = 53F9_0000h
GPIO-5_GPIO_DR is 53FD_C000h base + 0h offset = 53FD_C000h
GPIO-6_GPIO_DR is 53FE_0000h base + 0h offset = 53FE_0000h



GPIOx_GPIO_DR field descriptions

Field	Description
31–0 DR	Data bits. This register defines the value of the GPIO output when the signal is configured as an output (GDIR[n]=1). Writes to this register are stored in a register. Reading DR returns the value stored in the register if the signal is configured as an output (GDIR[n]=1), or the input signal's value if configured as an input (GDIR[n]=0). **NOTE:** The I/O multiplexer must be configured to GPIO mode for the DR value to connect with the signal. Reading the data register with the input path disabled always returns a zero value.

30.3.2 GPIO direction register (GPIOx_GPIO_GDIR)

GDIR functions as direction control when the IOMUX is in GPIO mode. Each bit specifies the direction of a one-bit signal. The mapping of each DIR bit to a corresponding SoC signal is determined by the SoC's pin assignment and the IOMUX table-for more details consult the IOMUX chapter.

Addresses: GPIO-1_GPIO_GDIR is 53F8_4000h base + 4h offset = 53F8_4004h
GPIO-2_GPIO_GDIR is 53F8_8000h base + 4h offset = 53F8_8004h
GPIO-3_GPIO_GDIR is 53F8_C000h base + 4h offset = 53F8_C004h
GPIO-4_GPIO_GDIR is 53F9_0000h base + 4h offset = 53F9_0004h
GPIO-5_GPIO_GDIR is 53FD_C000h base + 4h offset = 53FD_C004h
GPIO-6_GPIO_GDIR is 53FE_0000h base + 4h offset = 53FE_0004h

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

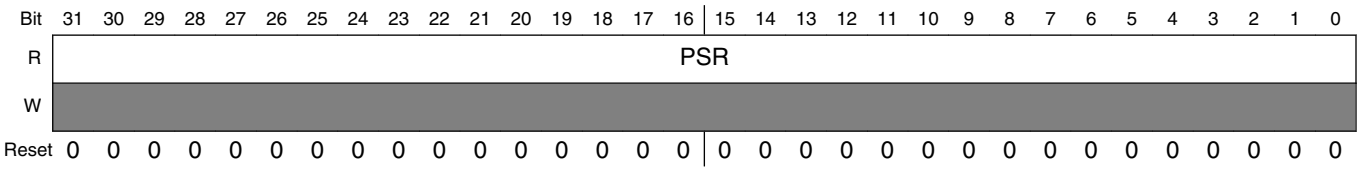
GPIOx_GPIO_GDIR field descriptions

Field	Description
31–0 GDIR	GPIO direction bits. Bit n of this register defines the direction of the GPIO[n] signal. **NOTE:** GDIR affects only the direction of the I/O signal when the corresponding bit in the I/O MUX is configured for GPIO. 0 GPIO is configured as input. 1 GPIO is configured as output.

30.3.3 GPIO pad status register (GPIOx_GPIO_PSR)

PSR is a read-only register. Each bit stores the value of the corresponding input signal (as configured in the IOMUX). This register is clocked with the ipg_clk_s clock, meaning that the input signal is sampled only when accessing this location. Two wait states are required any time this register is accessed for synchronization.

Addresses: GPIO-1_GPIO_PSR is 53F8_4000h base + 8h offset = 53F8_4008h
GPIO-2_GPIO_PSR is 53F8_8000h base + 8h offset = 53F8_8008h
GPIO-3_GPIO_PSR is 53F8_C000h base + 8h offset = 53F8_C008h
GPIO-4_GPIO_PSR is 53F9_0000h base + 8h offset = 53F9_0008h
GPIO-5_GPIO_PSR is 53FD_C000h base + 8h offset = 53FD_C008h
GPIO-6_GPIO_PSR is 53FE_0000h base + 8h offset = 53FE_0008h



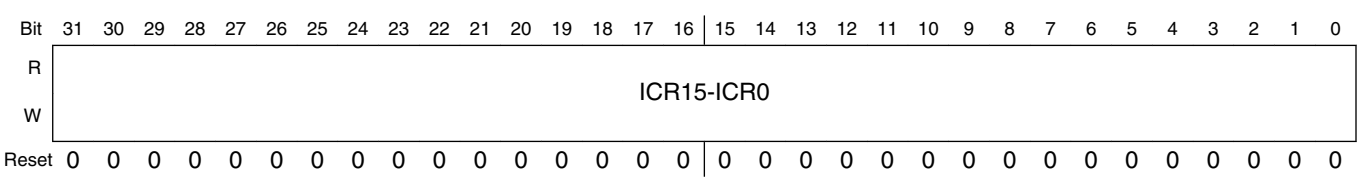
GPIOx_GPIO_PSR field descriptions

Field	Description
31–0 PSR	GPIO pad status bits (status bits). Reading PSR returns the state of the corresponding input signal. Settings: **NOTE:** The I/O multiplexer must be configured to GPIO mode for PSR to reflect the state of the corresponding signal.

30.3.4 GPIO interrupt configuration register1 (GPIOx_GPIO_ICR1)

ICR1 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

Addresses: GPIO-1_GPIO_ICR1 is 53F8_4000h base + Ch offset = 53F8_400Ch
GPIO-2_GPIO_ICR1 is 53F8_8000h base + Ch offset = 53F8_800Ch
GPIO-3_GPIO_ICR1 is 53F8_C000h base + Ch offset = 53F8_C00Ch
GPIO-4_GPIO_ICR1 is 53F9_0000h base + Ch offset = 53F9_000Ch
GPIO-5_GPIO_ICR1 is 53FD_C000h base + Ch offset = 53FD_C00Ch
GPIO-6_GPIO_ICR1 is 53FE_0000h base + Ch offset = 53FE_000Ch



GPIOx_GPIO_ICR1 field descriptions

Field	Description
31–0 ICR15-ICR0	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for lines 15 to 0. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 Interrupt n is low-level sensitive. 01 Interrupt n is high-level sensitive. 10 Interrupt n is rising-edge sensitive. 11 Interrupt n is falling-edge sensitive.

30.3.5 GPIO interrupt configuration register2 (GPIOx_GPIO_ICR2)

ICR2 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

Addresses: GPIO-1_GPIO_ICR2 is 53F8_4000h base + 10h offset = 53F8_4010h
GPIO-2_GPIO_ICR2 is 53F8_8000h base + 10h offset = 53F8_8010h
GPIO-3_GPIO_ICR2 is 53F8_C000h base + 10h offset = 53F8_C010h
GPIO-4_GPIO_ICR2 is 53F9_0000h base + 10h offset = 53F9_0010h
GPIO-5_GPIO_ICR2 is 53FD_C000h base + 10h offset = 53FD_C010h
GPIO-6_GPIO_ICR2 is 53FE_0000h base + 10h offset = 53FE_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR31-ICR16																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_GPIO_ICR2 field descriptions

Field	Description
31–0 ICR31-ICR16	Interrupt configuration 2 fields. These fields control the active condition of the interrupt function for lines 31 to 16. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00 Interrupt n is low-level sensitive. 01 Interrupt n is high-level sensitive. 10 Interrupt n is rising-edge sensitive. 11 Interrupt n is falling-edge sensitive.

30.3.6 GPIO interrupt mask register (GPIOx_GPIO_IMR)

IMR contains masking bits for each interrupt line.

Addresses: GPIO-1_GPIO_IMR is 53F8_4000h base + 14h offset = 53F8_4014h
GPIO-2_GPIO_IMR is 53F8_8000h base + 14h offset = 53F8_8014h
GPIO-3_GPIO_IMR is 53F8_C000h base + 14h offset = 53F8_C014h
GPIO-4_GPIO_IMR is 53F9_0000h base + 14h offset = 53F9_0014h
GPIO-5_GPIO_IMR is 53FD_C000h base + 14h offset = 53FD_C014h
GPIO-6_GPIO_IMR is 53FE_0000h base + 14h offset = 53FE_0014h

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

GPIOx_GPIO_IMR field descriptions

Field	Description
31–0 IMR	Interrupt Mask bits. This register is used to enable or disable the interrupt function on each of the 32 GPIO signals. Settings: Bit IMR[n] (n=0...31) controls interrupt n as follows: 0 Interrupt n is disabled. 1 Interrupt n is enabled.

30.3.7 GPIO interrupt status register (GPIOx_GPIO_ISR)

The ISR functions as an interrupt status indicator. Each bit indicates whether an interrupt condition has been met for the corresponding input signal. When an interrupt condition is met (as determined by the corresponding interrupt condition register field), the corresponding bit in this register is set. Two wait states are required in read access for synchronization. One wait state is required for reset.

Addresses: GPIO-1_GPIO_ISR is 53F8_4000h base + 18h offset = 53F8_4018h
GPIO-2_GPIO_ISR is 53F8_8000h base + 18h offset = 53F8_8018h
GPIO-3_GPIO_ISR is 53F8_C000h base + 18h offset = 53F8_C018h
GPIO-4_GPIO_ISR is 53F9_0000h base + 18h offset = 53F9_0018h
GPIO-5_GPIO_ISR is 53FD_C000h base + 18h offset = 53FD_C018h
GPIO-6_GPIO_ISR is 53FE_0000h base + 18h offset = 53FE_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_GPIO_ISR field descriptions

Field	Description
31–0 ISR	Interrupt status bits - Bit n of this register is asserted (active high) when the active condition (as determined by the corresponding ICR bit) is detected on the GPIO input and is waiting for service. The value of this register is independent of the value in IMR. When the active condition has been detected, the corresponding bit remains set until cleared by software. Status flags are cleared by writing a 1 to the corresponding bit position.

30.3.8 GPIO edge select register (GPIO*x*_GPIO_EDGE_SEL)

EDGE_SEL may be used to override the ICR registers' configuration. If the EDGE_SEL bit is set, then a rising edge or falling edge in the corresponding signal generates an interrupt. This register provides backward compatibility. On reset all bits are cleared (ICR is not overridden).

Addresses: GPIO-1_GPIO_EDGE_SEL is 53F8_4000h base + 1Ch offset = 53F8_401Ch
GPIO-2_GPIO_EDGE_SEL is 53F8_8000h base + 1Ch offset = 53F8_801Ch
GPIO-3_GPIO_EDGE_SEL is 53F8_C000h base + 1Ch offset = 53F8_C01Ch
GPIO-4_GPIO_EDGE_SEL is 53F9_0000h base + 1Ch offset = 53F9_001Ch
GPIO-5_GPIO_EDGE_SEL is 53FD_C000h base + 1Ch offset = 53FD_C01Ch
GPIO-6_GPIO_EDGE_SEL is 53FE_0000h base + 1Ch offset = 53FE_001Ch

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

GPIO*x*_GPIO_EDGE_SEL field descriptions

Field	Description
31–0 EDGE_SEL	Edge select. When EDGE_SEL[*n*] is set, the GPIO disregards the ICR[*n*] setting, and detects any edge on the corresponding input signal.

Chapter 31
General-Purpose Media Interface (GPMI)

31.1 Introduction

This chapter describes the general-purpose media interface (GPMI) on the i.MX50.

31.2 Overview

The GPMI controller is a flexible interface to up to eight NAND flash.

- ONFI 2.2, samsung Toggle NAND protocol compatible
- The NAND mode has configurable address and command behavior, providing support for future devices not yet specified.

The GPMI resides on the APBH. The GPMI also provides an interface to the BCH module to allow direct parity processing.

Registers are clocked on the HCLK domain. The I/O and pin timing are clocked on a dedicated GPMICLK domain. GPMICLK can be set to maximize I/O performance.

Figure 31-1 shows a block diagram of the GPMI controller.

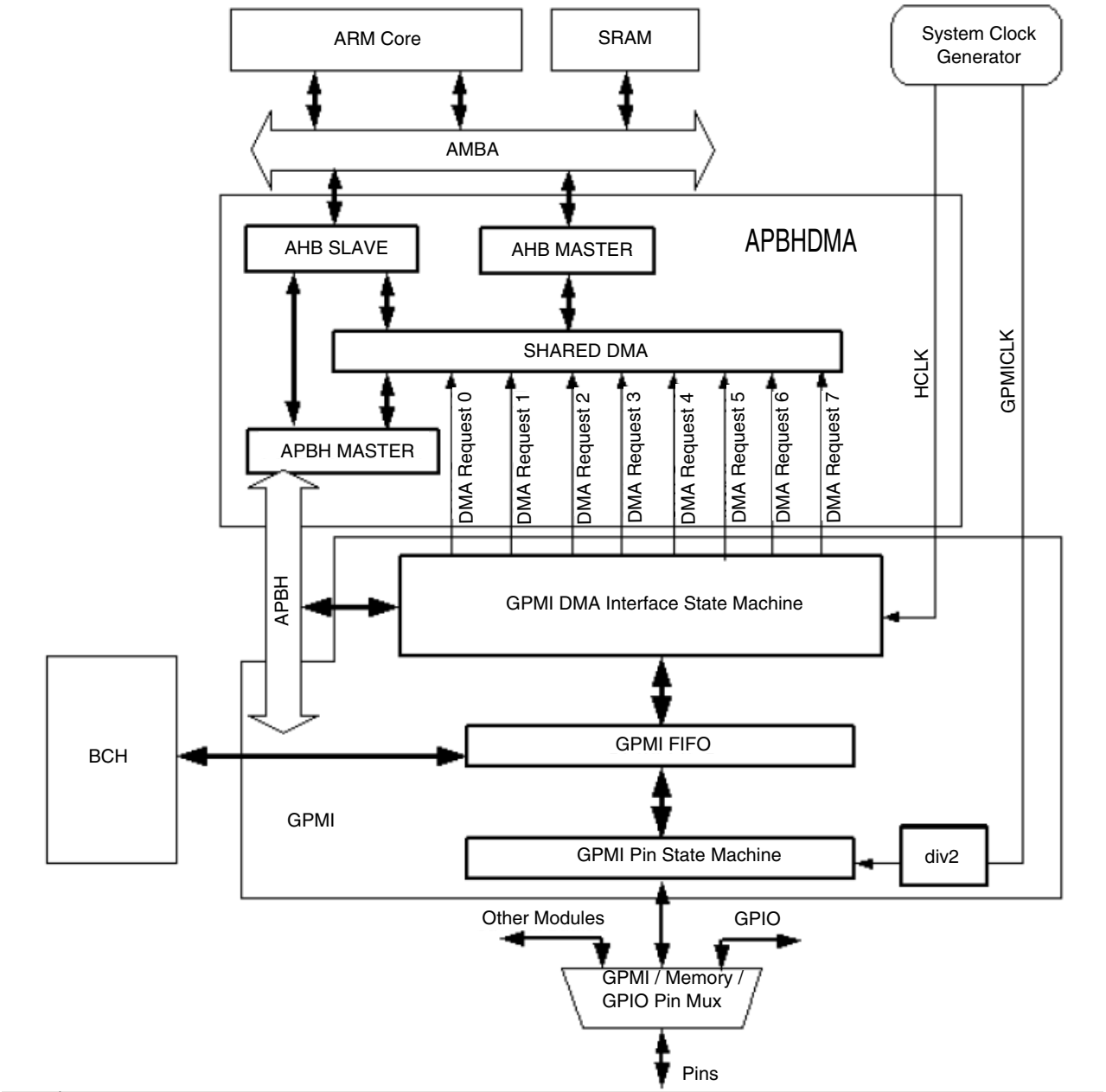


Figure 31-1. General-Purpose Media Interface Controller Block Diagram

31.3 GPMI NAND Mode

The general-purpose media interface has several features to efficiently support NAND:

- Individual chip select and ready/busy pins for up to eight NANDs.
- Individual state machine and DMA channel for each chip select.

- Special command modes work with DMA controller to perform all normal NAND functions without ARM platform intervention.
- Configurable timing based on a dedicated clock allows optimal balance of high NAND performance and low system power.

GPMI and DMA have been designed to handle complex multi-page operations without ARM platform intervention. The DMA uses a linked descriptor function with branching capability to automatically handle all of the operations needed to read/write multiple pages:

- Data/Register Read/Write-The GPMI can be programmed to read or write multiple cycles to the NAND address, command or data registers.
- Wait for NAND Ready-The GPMI's Wait-for-Ready mode can monitor the ready/busy signal of a single NAND flash and signal the DMA when the device has become ready. It also has a time-out counter and can indicate to the DMA that a time-out error has occurred. The DMAs can conditionally branch to a different descriptor in the case of an error.
- Check Status-The Read-and-Compare mode allows the GPMI to check NAND status against a reference. If an error is found, the GPMI can instruct the DMA to branch to an alternate descriptor, which attempts to fix the problem or asserts a ARM platform IRQ.

31.3.1 Multiple NAND Support

The GPMI supports up to eight NAND chip selects, each with independent ready/busy signals. Since they share a data bus and control lines, the GPMI can only actively communicate with a single NAND at a time. However, all NANDs can concurrently perform internal read, write, or erase operations. With fast NAND flash and software support for concurrent NAND operations, this architecture allows the total throughput to approach the data bus speed, which can be as high as 50 MB/s (8-bit bus running at 50 MHz single clock edge) in asynchronous mode and 200MB/s (8-bit bus running at 100MHz both edges clock) in Source Synchronous mode.

There are two options for controlling the eight NAND chip selects via the DMA interface. The first option is the one to one mapping, where the each DMA channel is tied to it own NAND. For example DMA channel 'n' accesses only NAND attached to chip select 'n'. The second option is the decoupled mode where a DMA channel can access any or all NANDs connected to the GPMI. A DMA channel will signify the NAND it wants to access by writing its chip select value in the GPMI_CTRL0_CS field and setting the GPMI_CTRL1_DECOUPLE_CS to 1. This option is useful if software chooses to use only one DMA channel to access all the attached NAND devices.

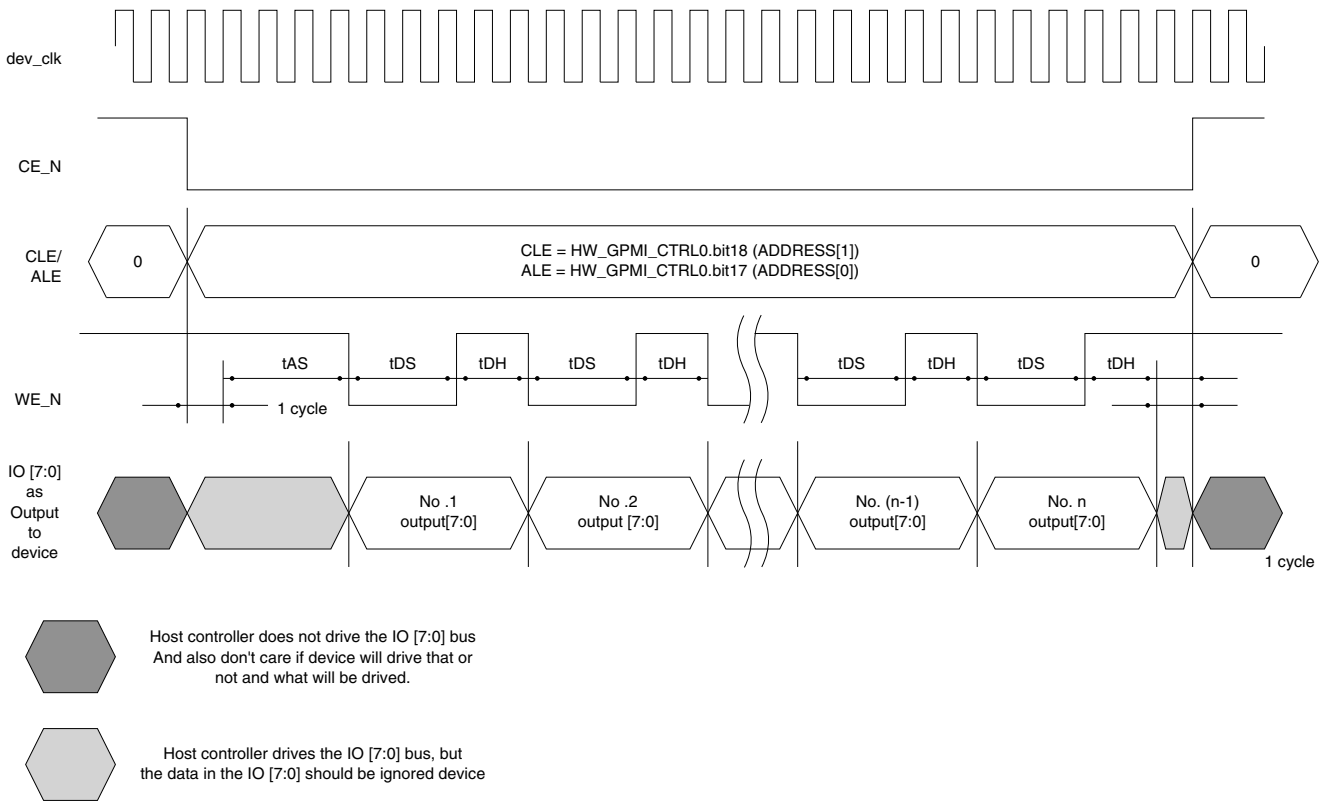
31.3.2 GPMI NAND Timing and Clocking

The dedicated clock, GPMICLK, is used as a timing reference for NAND flash I/O. Since various NANDs have different timing requirements, GPMICLK may need to be adjusted for each application. While the actual pin timings are limited by the NAND chips used, the GPMI can support data bus speeds of up to 200 MHz x 8 bits. The actual read/write strobe timing parameters are adjusted as indicated in the register descriptions in the Programmable Registers section.

31.3.3 Basic NAND Timing

31.3.3.1 NAND Asynchronous Timing

Figure 31-4 and illustrates the operation of the output (from host to device) timing parameters in NAND ONFI asynchronous mode.



- tAS is configurable by programming HW_GPMI_TIMING0 Address_Setup: In this example, Address_Setup - 4, tAS is equal to 4 dev_clk cycles.
- tDS is configurable by programming HW_GPMI_TIMING0 Data_Setup; In this example, Data_Setup - 3, tDS is equal to 3 dev_clk cycles
- tDH is configuarble by programming HW_GPMI_TIMING0 Data_Hold: In this example, Data_Hold = 2, tDH is equal to 2 dev_clk cycles
- tAS/tDS/tDH will extend, if the output data us not ready in device fifo.

Figure 31-2. Asynchronous mode basic write timing diagram

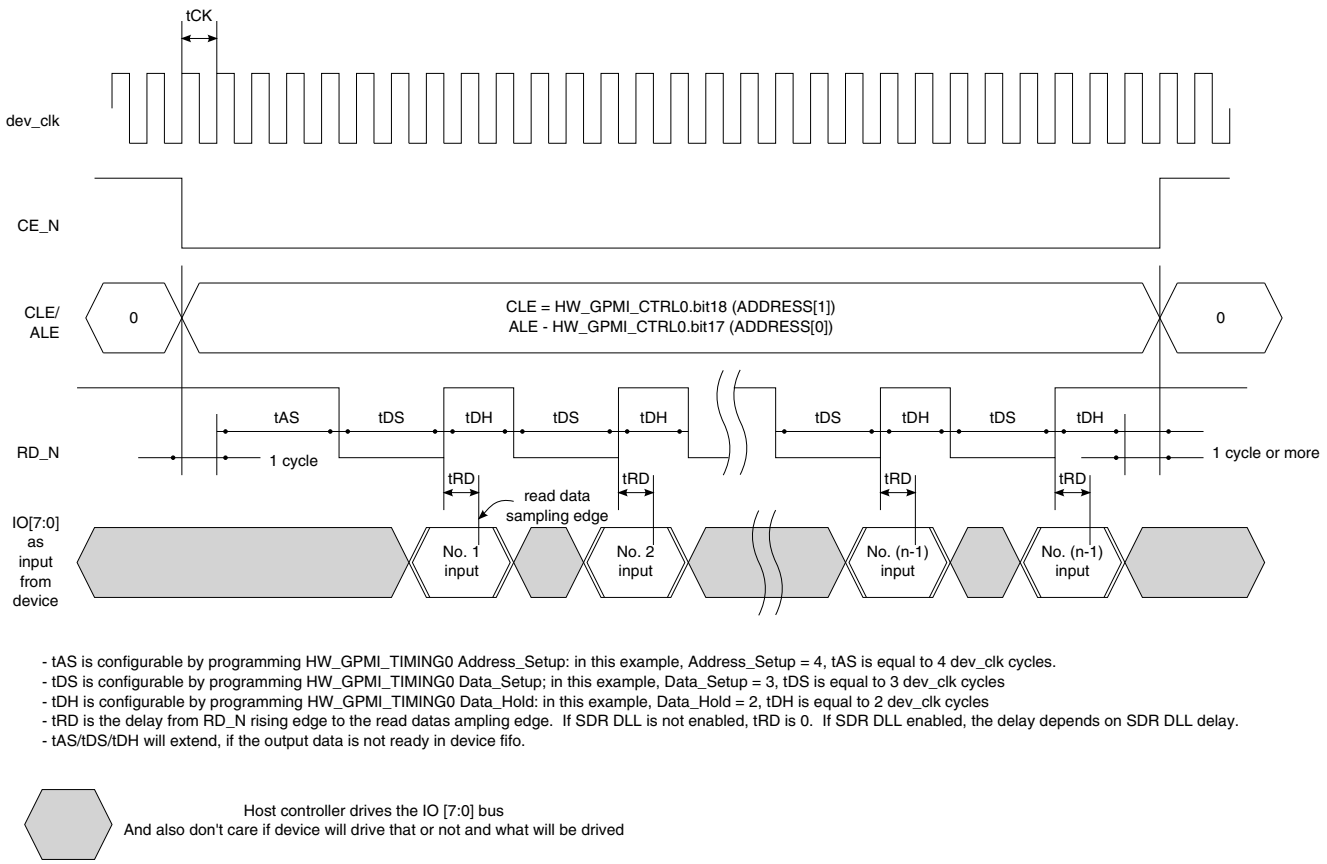


Figure 31-3. ONFI asynchronous mode basic read timing diagram

31.3.3.2 NAND Asynchronous EDO Mode Timing

In high-speed NANDS, the read data may not be valid until after the read strobe (RDN) deasserts. This is the case when the minimum tDS is programmed to achieve higher bandwidth. The GPMI implements a feedback read strobe to sample the read data. The feedback read strobe can be delayed to support fast nand EDO (Extended Data Out) timing where the read strobe may deassert before the read data is valid, and read data is valid for some time after read strobe. Nand EDO timings is applied typically for read cycle frequency above 33 MHz. See Figure 31-4.

The GPMI provides control over the amount of delay applied to the feedback read strobe. This delay depends on the maximum read access time (tREA) of the nand and the read pulse width (tRP) used to access the nand. tRP is specified by GPMI_TIMING0_DATA_SETUP register. When (tREA + 4ns) is less than tRP, no

delay is required to sample to nand read data. (The 4ns provides adequate data setup time for the GPMI.) In this case set GPMI_CTRL1_HALF_PERIOD = 0; GPMI_CTRL1_RDN_DELAY = 0; GPMI_CTRL1_DLL_ENABLE = 0.

When (tREA + 4ns) is greater than or equal to tRP, a delay of the feedback read strobe is required to sample to nand read data. This delay is equal to the difference between these two timings:

DELAY = tREA + 4ns - tRP.

Since the GPMI delay chain is limited to 16ns maximum, if DELAY > 16ns then increase tRP by increasing the value of GPMI_TIMING0_DATA_SETUP until DELAY is less than or equal to 16ns.

The GPMI programming for this DELAY depends on the GPMICLK period. The GPMI DLL will not function properly if the GPMICLK period is greater than 32ns: disable the DLL if this is the case. If the GPMICLK period is greater than 16ns (and not greater than 32ns), set the GPMI_CTRL1_HALF_PERIOD=1; This will cause the DLL reference period (RP) to be one-half of the GPMICLK period. If the GPMICLK period is 16ns or less then set the GPMI_CTRL1_HALF_PERIOD=0; This will cause the DLL reference period (RP) to be equal to the GPMICLK period. DELAY is a multiple (0 to 1.875) of RP.

The GPMI_CTRL1_RDN_DELAY is encoded as a 1-bit integer and 3-bit fraction delay factor. See table below. DELAY is a multiple of the delay factor and the reference period:

HW_GPMI_CTRL1_RDN_DELAY	0	1	2	3	4	5	6	7
Delay Factor	0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875
HW_GPMI_CTRL1_RDN_DELAY	8	9	10	11	12	13	14	15
Delay Factor	1.000	1.125	1.250	1.375	1.500	1.625	1.750	1.875

DELAY = DelayFactor x RP or DELAY = GPMI_CTRL1_RDN_DELAY x 0.125 x RP.

Use this equation to calculate the value for GPMI_CTRL1_RDN_DELAY. Then set GPMI_CTRL1_DLL_ENABLE=1.

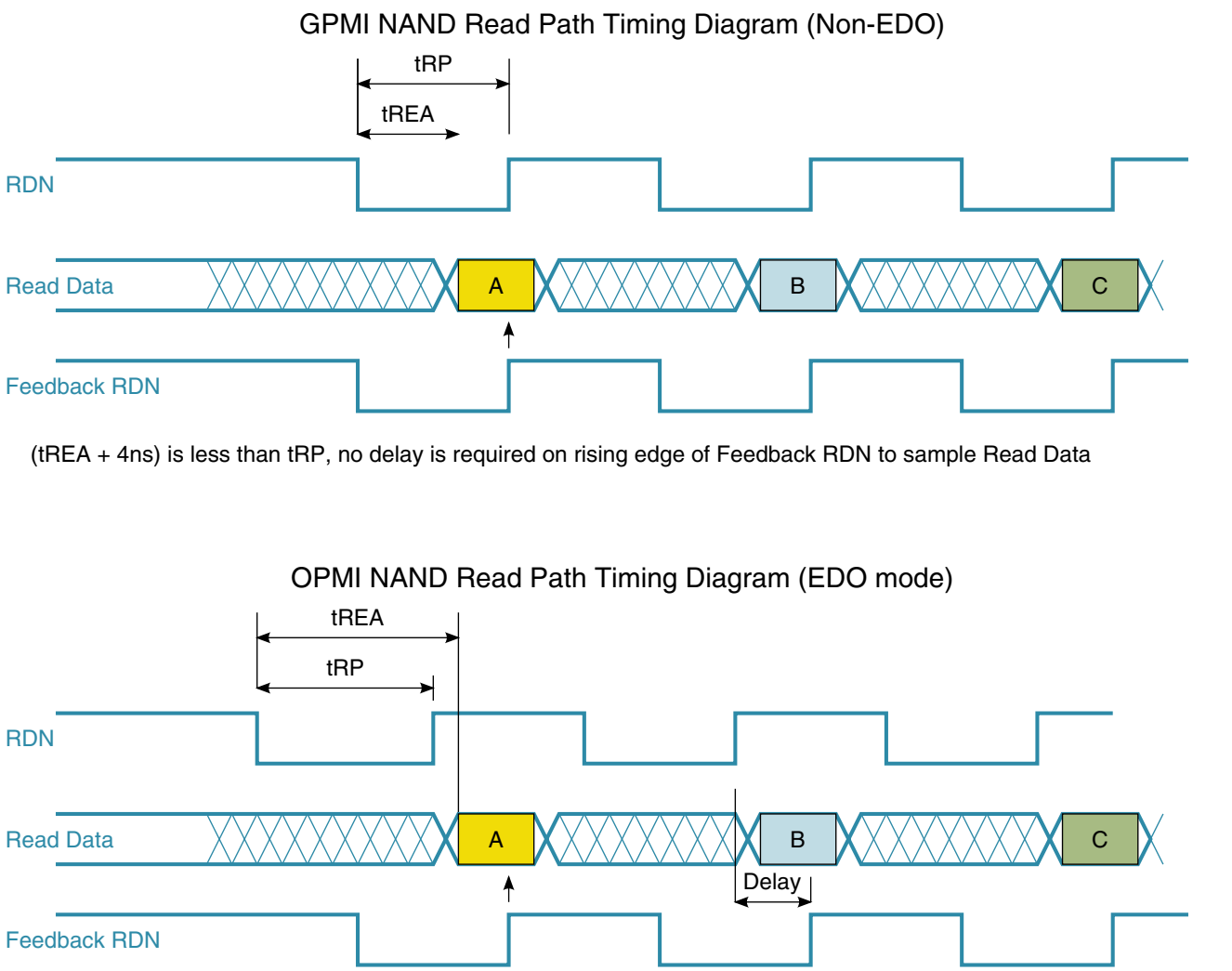


Figure 31-4. NAND Read Path Timing

For example, a nand with tREAmax = 20 ns, tRPmin = 12 ns, and tRCmin = 25 ns (read cycle time) may be programmed as follows:

- GPMICLK clock frequency: Consider 480/6 = 80 MHz which is 12.5 ns clock period. This is too close to the minimum nand spec if we program the data setup and hold to 1 GPMICLK cycle. Consider 480/7=68.57 MHz which is 14.58 ns clock period. With data setup and hold set to 1, we have a tRP of 14.58ns and a tRC of 29.16 ns (good margins).
- Since (tREA +4ns) is greater than tRP, required DELAY = tREA + 4ns - tRP = 20 + 4 - 14.58ns = 9.42 ns.

- HALF_PERIOD =0, since GPMICLK period is less than 16ns. So RP=GPMICLK period = 14.58ns.
- DELAY = GPMI_CTRL1_RDN_DELAY x 0.125 x RP. 9.42 ns = GPMI_CTRL1_RDN_DELAY x 0.125 x 14.58ns. GPMI_CTRL1_RDN_DELAY = 5 (round off 5.169)

NOTE

It is recommended that the drive strength of GPMI_RDn and GPMI_WRn output pins be set to 8 mA. This will reduce the transition time under heavy loads. Low transition times will be important when NAND interface read and write cycle times are below 30 ns. The other GPMI pins may remain at 4 mA, since their frequency is only up to half that of GPMI_RDn and GPMI_WRn.

31.3.3.3 NAND ONFI Source Synchronous Mode Timing

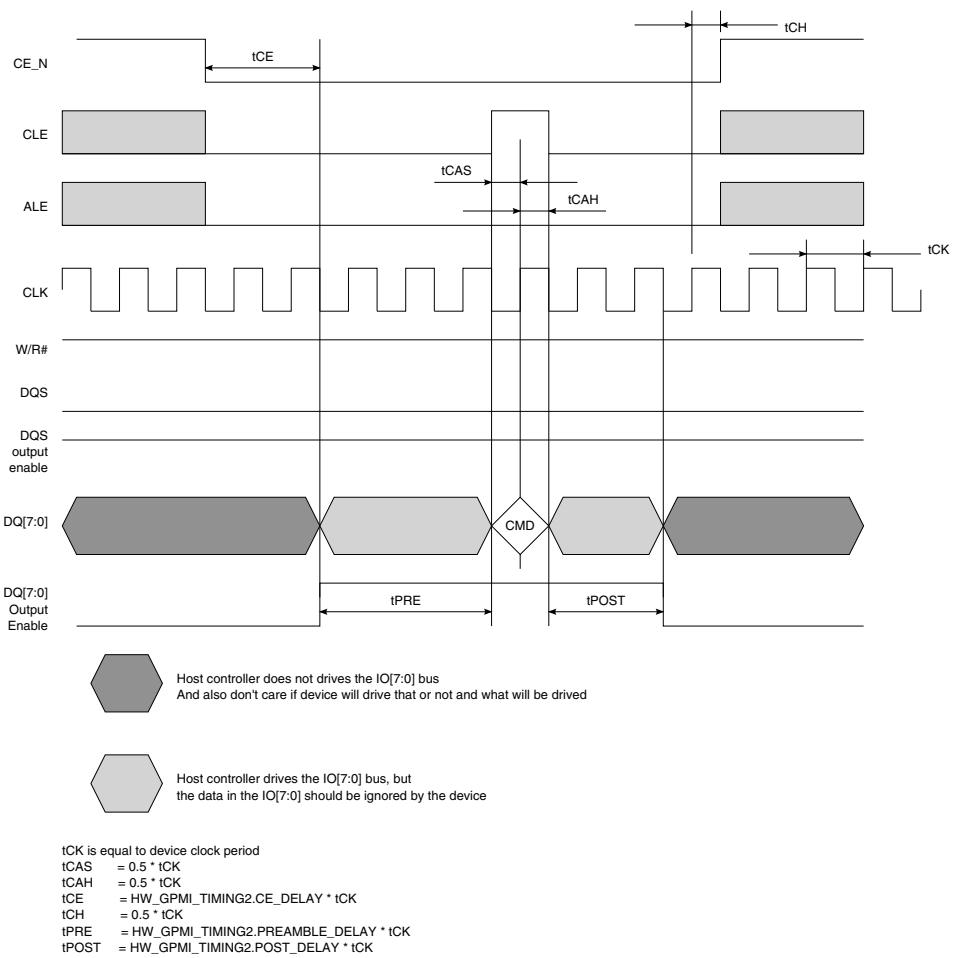


Figure 31-5. ONFI source synchronous mode basic command write timing diagram

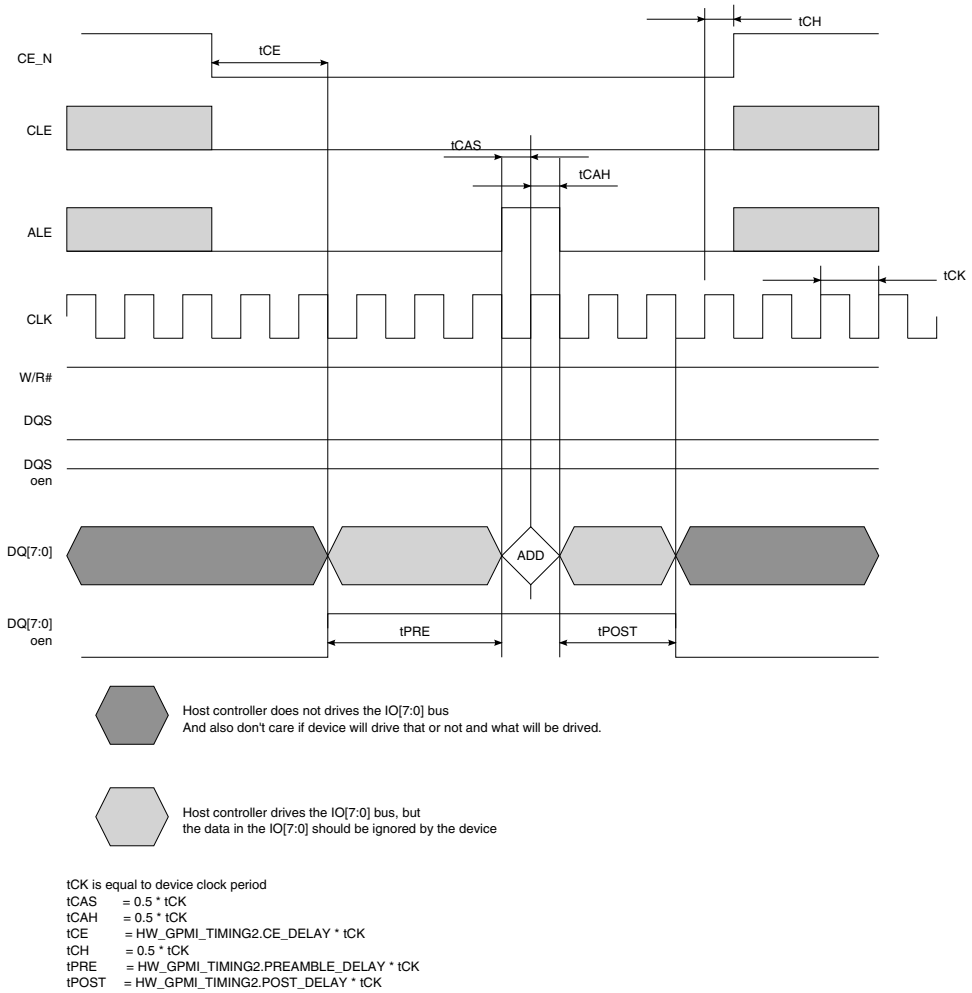


Figure 31-6. ONFI source synchronous mode basic address write timing diagram

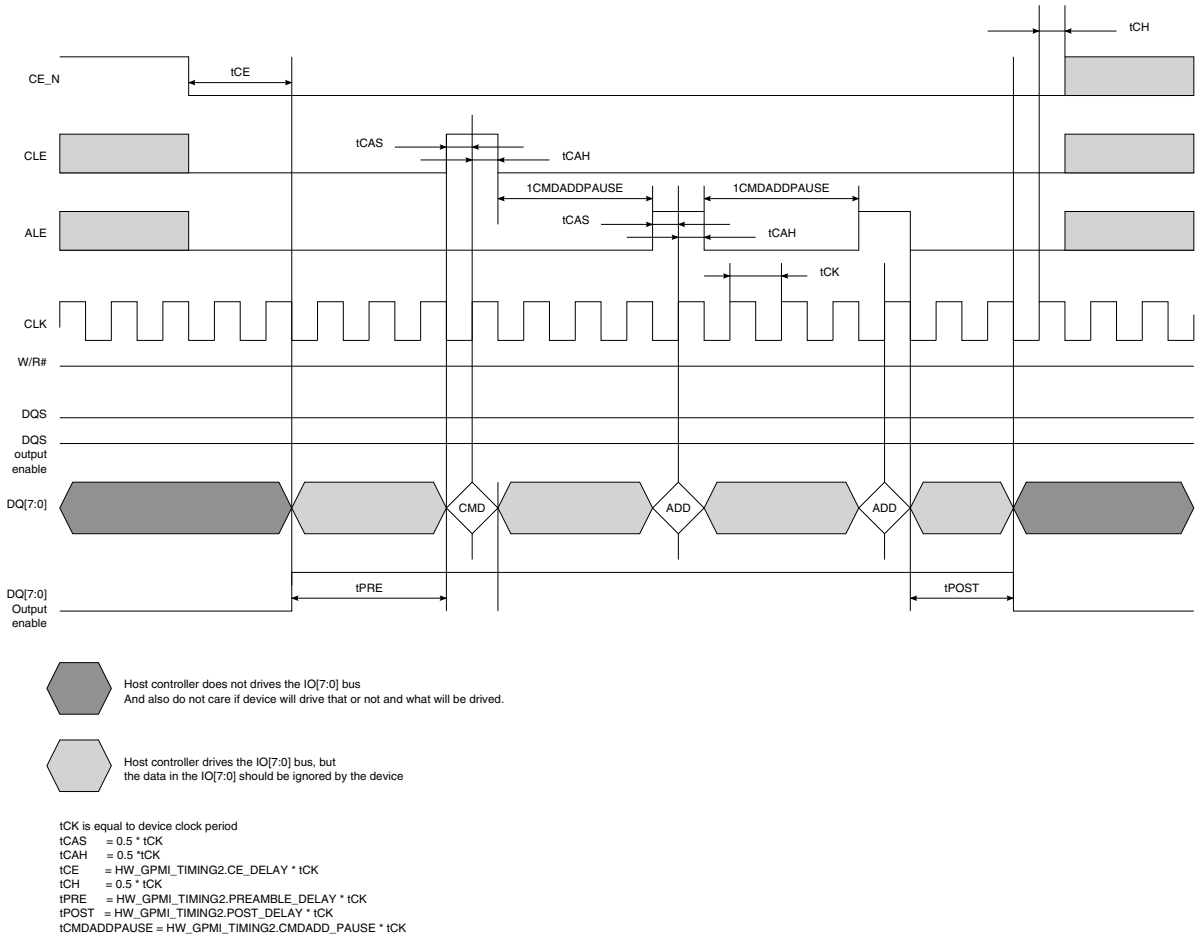
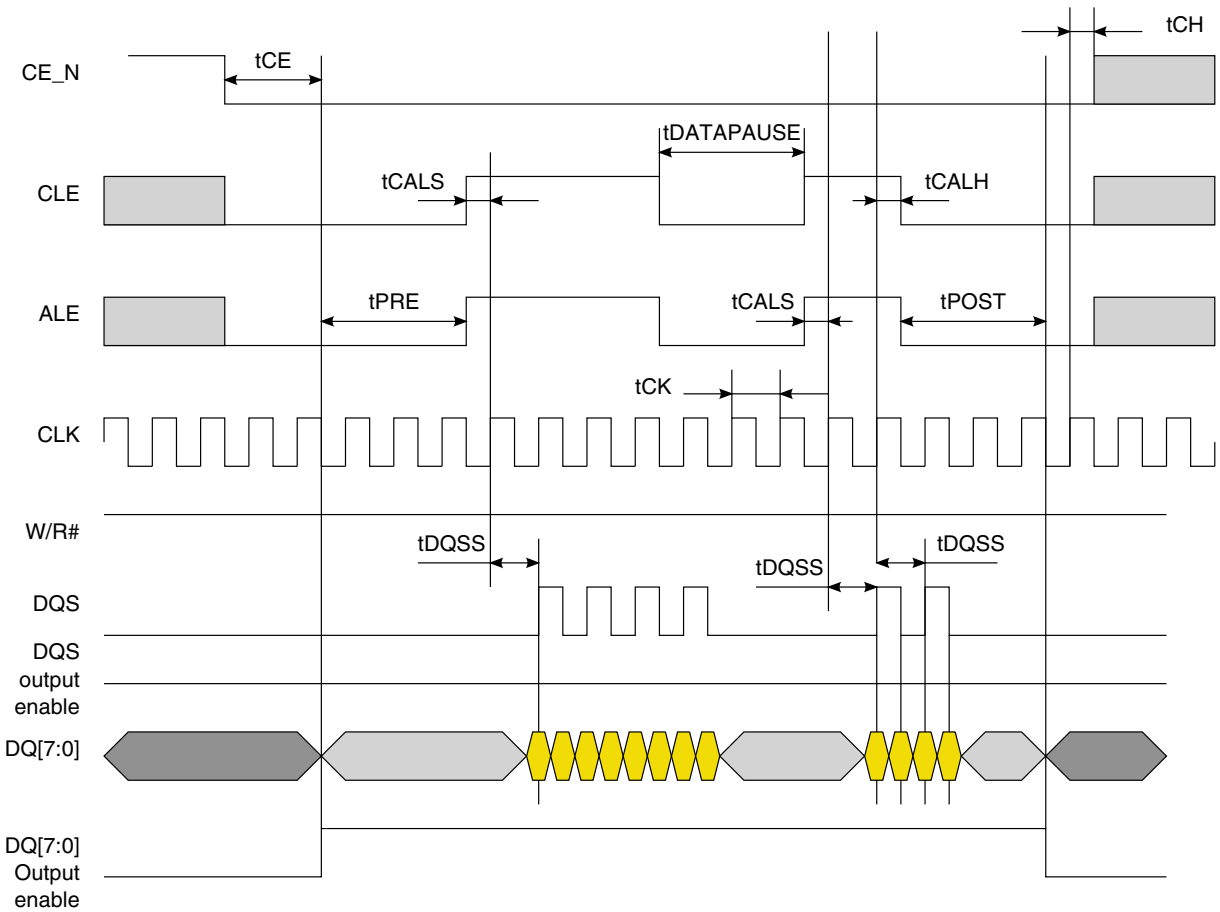
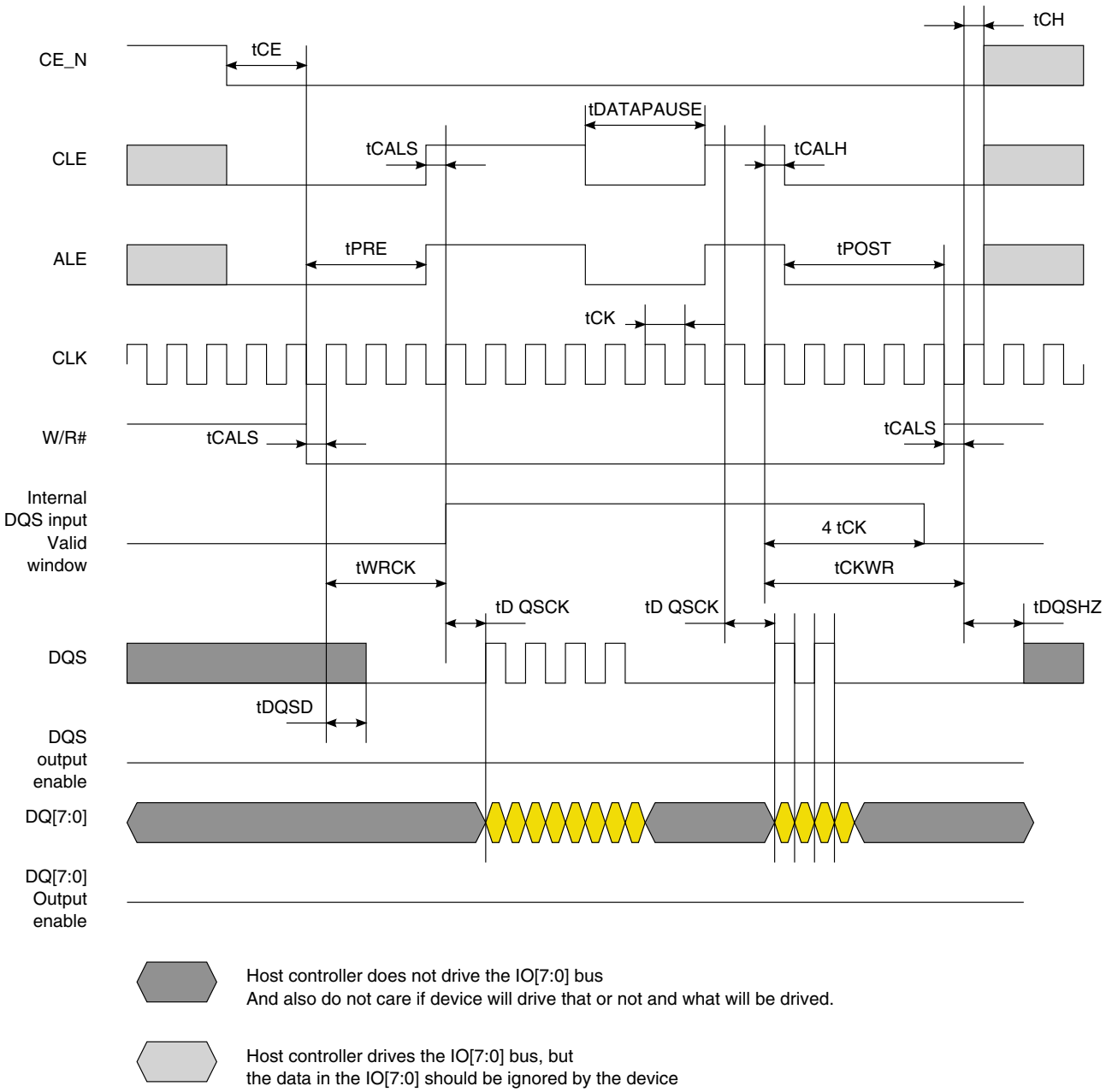


Figure 31-7. ONFI source synchronous mode comand + address write timing diagram



tCK is equal to device clock period
tCE = HW_GPI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tCALS = 0.5 * tCK
tCALH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY *tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK
tDATAPAUSE = HW_GPMI_TIMING2.DATA_PAUSE * tCK
tDQSS = tCK

Figure 31-8. ONFI source synchornous mode data write timing diagram



tCK is equal to device clock period
tCE = HW_GPI_TIMING2.CE_DELAY * tCK
tCH = 0.5 * tCK
tCALS = 0.5 * tCK
tCALH = 0.5 * tCK
tPRE = HW_GPMI_TIMING2.PREAMBLE_DELAY *tCK
tPOST = HW_GPMI_TIMING2.POST_DELAY * tCK
tDATAPAUSE = HW_GPMI_TIMING2.DATA_PAUSE * tCK
tWRCK/tCKWR/tDQSD/tDASCK/tDASHZ are device parameters

Figure 31-9. ONFI source synchronous mode data read timing diagram

31.3.3.4 NAND Toggle Mode Timing

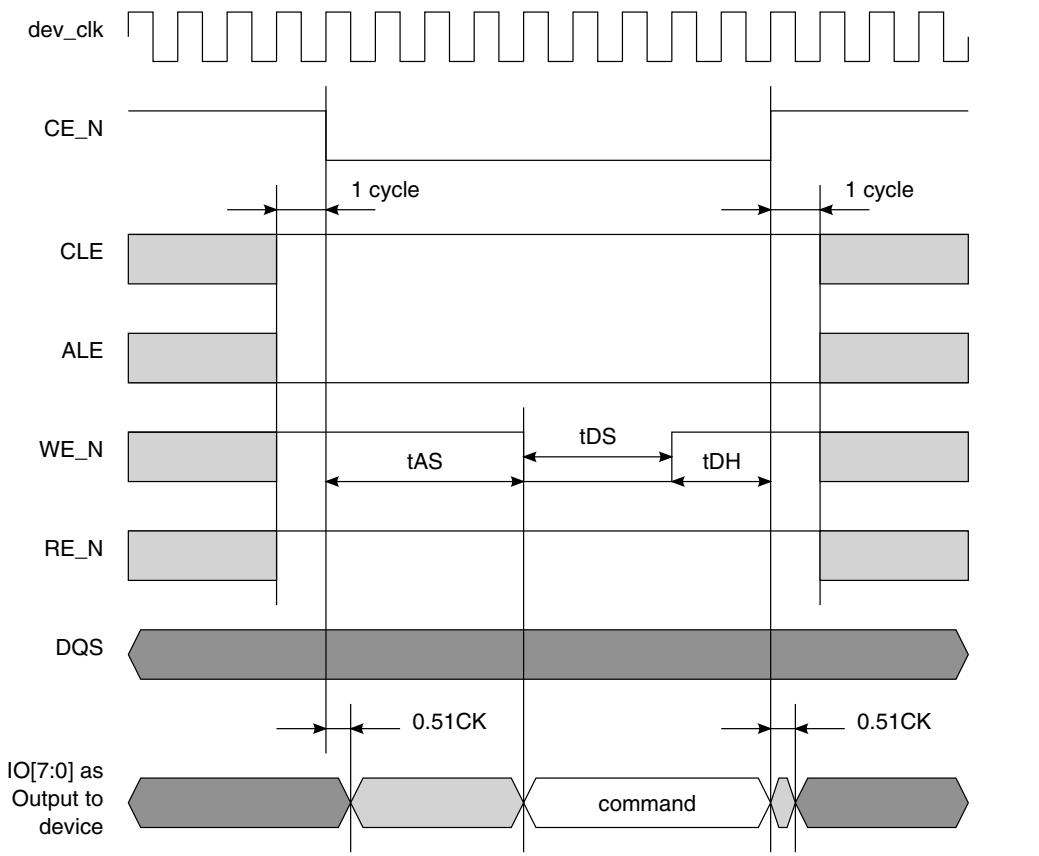
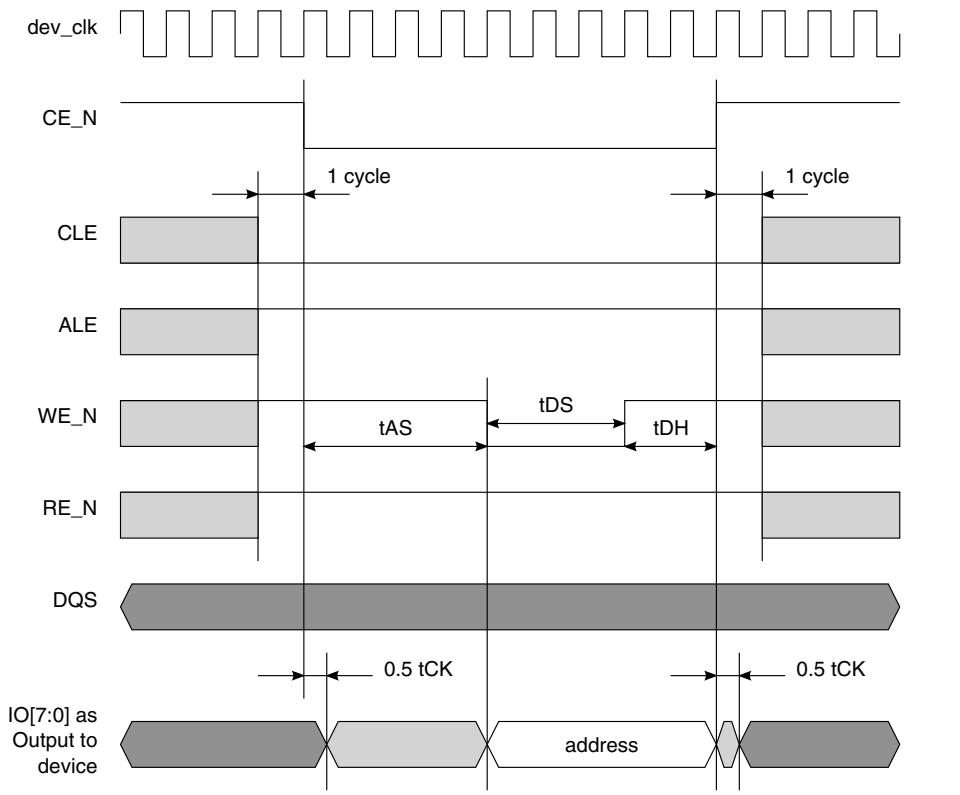


Figure 31-10. SAMSUNG toggle mode basic command write timing diagram



Host controller does not drive the IO[7:0] bus
And also does not care if device will drive that or not and what will be drived.

Host controller drives the IO[7:0] bus, but
the data in the IO[7:0] should be ignored by the device

- tAS is configurable by programming HW_GPMI_TIMING0 Address_Setup;
- tDS is configurable by programming HW_GPMI_TIMING0 Data_Setup;
- tDH is configurable br programming HW_GPMI_TIMING Data_Hold;
- tAS/tDS/tDH will extend, if the output data is not ready in device fifo.

Figure 31-11. SAMSUNG toggle mode basic address write timing diagram

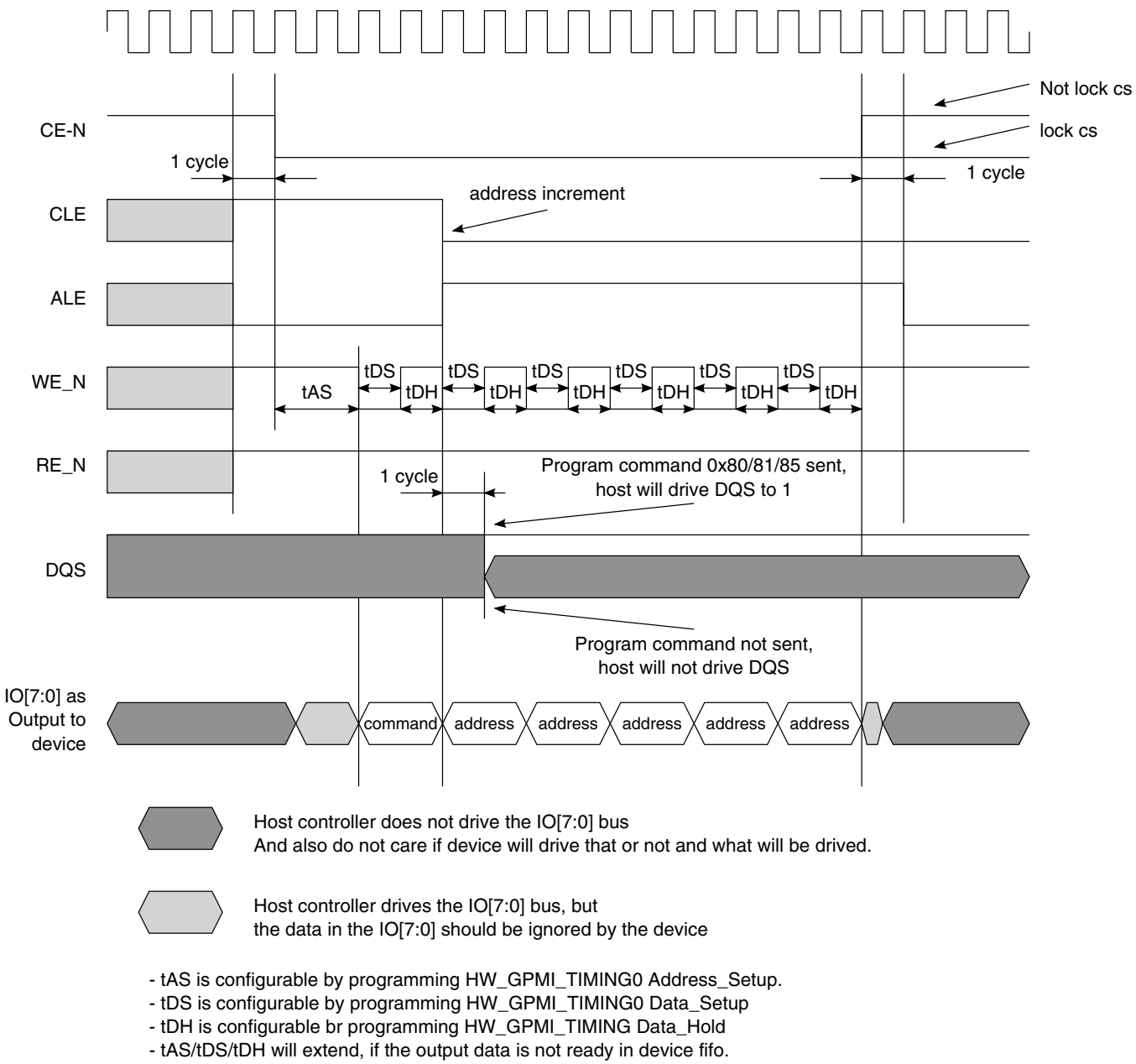


Figure 31-12. SAMSUNG toggle mode basic command + address timing diagram

31.3.4 Hardware BCH Interface

The GPMI provides an interface to the BCH module. This reduces the SOC bus traffic and the software involvement. When in BCH mode, parity information is inserted on-the-fly during writes to 8-bit NAND devices. The BCH will supply payload and parity to the

GPMI to write to the NAND. During NAND reads, parity is checked and ECC processing is performed after each read block. In this case the GPMI reads the NAND device and redirects the data and parity to the BCH module for ECC processing.

To program the BCH for NAND writes, remove the soft reset and clock gates from HW_BCH_CTRL_SFTRST and HW_BCH_CTRL_CLKGATE. The bulk of BCH programming is actually applied to the GPMI via PIO operations embedded in its DMA command structures. This has a subtle implication when writing to the GPMI ECC registers: access to the these registers must be written in progressive register order. Thus, to write to the HW_GPMI_ECCCOUNT register, write first (in order) to registers HW_GPMI_CTRL0, HW_GPMI_COMPARE, and HW_GPMI_ECCCTRL before writing to HW_GPMI_ECCCOUNT. These additional register writes need to be accounted for in the CMDWORDS field of the respective DMA channel command register.

Note that the HW_GPMI_PAYLOAD and HW_GPMI_AUXILIARY pointers need to be word-aligned for proper ECC operation. If those pointers are non-word-aligned, then the BCH engine will not operate properly and could possibly corrupt system memory in the adjoining memory regions.

31.4 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically.

31.5 Programmable Registers

GPMI Hardware Register Format Summary

GPMI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_6000	GPMI Control Register 0 (GPMI_CTRL0)	32	R/W	C000_0000h	31.5.1/ 1567
4100_6004	GPMI Control Register 0 (GPMI_CTRL0_SET)	32	R/W	C000_0000h	31.5.1/ 1567
4100_6008	GPMI Control Register 0 (GPMI_CTRL0_CLR)	32	R/W	C000_0000h	31.5.1/ 1567

Table continues on the next page...

GPMI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_600C	GPMI Control Register 0 (GPMI_CTRL0_TOG)	32	R/W	C000_0000h	31.5.1/ 1567
4100_6010	GPMI Compare Register (GPMI_COMPARE)	32	R/W	0000_0000h	31.5.2/ 1569
4100_6020	GPMI Integrated ECC Control Register (GPMI_ECCCTRL)	32	R/W	0000_0000h	31.5.3/ 1570
4100_6024	GPMI Integrated ECC Control Register (GPMI_ECCCTRL_SET)	32	R/W	0000_0000h	31.5.3/ 1570
4100_6028	GPMI Integrated ECC Control Register (GPMI_ECCCTRL_CLR)	32	R/W	0000_0000h	31.5.3/ 1570
4100_602C	GPMI Integrated ECC Control Register (GPMI_ECCCTRL_TOG)	32	R/W	0000_0000h	31.5.3/ 1570
4100_6030	GPMI Integrated ECC Transfer Count Register (GPMI_ECCCOUNT)	32	R/W	0000_0000h	31.5.4/ 1571
4100_6040	GPMI Payload Address Register (GPMI_PAYLOAD)	32	R/W	0000_0000h	31.5.5/ 1572
4100_6050	GPMI Auxiliary Address Register (GPMI_AUXILIARY)	32	R/W	0000_0000h	31.5.6/ 1572
4100_6060	GPMI Control Register 1 (GPMI_CTRL1)	32	R/W	0004_0004h	31.5.7/ 1573
4100_6064	GPMI Control Register 1 (GPMI_CTRL1_SET)	32	R/W	0004_0004h	31.5.7/ 1573
4100_6068	GPMI Control Register 1 (GPMI_CTRL1_CLR)	32	R/W	0004_0004h	31.5.7/ 1573
4100_606C	GPMI Control Register 1 (GPMI_CTRL1_TOG)	32	R/W	0004_0004h	31.5.7/ 1573
4100_6070	GPMI Timing Register 0 (GPMI_TIMING0)	32	R/W	0001_0203h	31.5.8/ 1576
4100_6080	GPMI Timing Register 1 (GPMI_TIMING1)	32	R/W	0000_0000h	31.5.9/ 1577
4100_6090	GPMI Timing Register 2 (GPMI_TIMING2)	32	R/W	0302_3336h	31.5.10/ 1577
4100_60A0	GPMI DMA Data Transfer Register (GPMI_DATA)	32	R/W	0000_0000h	31.5.11/ 1578
4100_60B0	GPMI Status Register (GPMI_STAT)	32	R	0000_0005h	31.5.12/ 1579
4100_60C0	GPMI Debug Information Register (GPMI_DEBUG)	32	R	0000_0000h	31.5.13/ 1581
4100_60D0	GPMI Version Register (GPMI_VERSION)	32	R	0501_0000h	31.5.14/ 1581

Table continues on the next page...

GPMI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_60E0	GPMI Debug2 Information Register (GPMI_DEBUG2)	32	R/W	0000_F100h	31.5.15/ 1582
4100_60F0	GPMI Debug3 Information Register (GPMI_DEBUG3)	32	R	0000_0000h	31.5.16/ 1584
4100_6100	GPMI Double Rate Read DLL Control Register (GPMI_READ_DDR_DLL_CTRL)	32	R/W	0000_0038h	31.5.17/ 1585
4100_6110	GPMI Double Rate Write DLL Control Register (GPMI_WRITE_DDR_DLL_CTRL)	32	R/W	0000_0038h	31.5.18/ 1586
4100_6120	GPMI Double Rate Read DLL Status Register (GPMI_READ_DDR_DLL_STS)	32	R	0000_0000h	31.5.19/ 1587
4100_6130	GPMI Double Rate Write DLL Status Register (GPMI_WRITE_DDR_DLL_STS)	32	R	0000_0000h	31.5.20/ 1588

31.5.1 GPMI Control Register 0 (GPMI_CTRL0*n*)

The GPMI control register 0 specifies the GPMI transaction to perform for the current command chain item.

Addresses: GPMI_CTRL0 is 4100_6000h base + 0h offset = 4100_6000h

GPMI_CTRL0_SET is 4100_6000h base + 4h offset = 4100_6004h

GPMI_CTRL0_CLR is 4100_6000h base + 8h offset = 4100_6008h

GPMI_CTRL0_TOG is 4100_6000h base + Ch offset = 4100_600Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	SFTRST	CLKGATE	RUN	DEV_IRQ_EN	LOCK_CS	UDMA	COMMAND_MODE		WORD_LENGTH	CS			ADDRESS			ADDRESS_INCREMENT
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	XFER_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_CTRL0*n* field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. This will not work if the CLKGATE bit is already set to '1'. CLKGATE must be cleared to '0' before issuing a soft reset. Also the GPMICLK must be running for this to work properly.

Table continues on the next page...

GPMI_CTRL0*n* field descriptions (continued)

Field	Description
	0x0 **RUN** — Allow GPMI to operate normally. 0x1 **RESET** — Hold GPMI in reset.
30 CLKGATE	Set this bit zero for normal operation. Setting this bit to one (default), gates all of the block level clocks off for miniminizing AC energy consumption. 0x0 **RUN** — Allow GPMI to operate normally. 0x1 **NO_CLKS** — Do not clock GPMI gates in order to minimize power consumption.
29 RUN	The GPMI is busy running a command whenever this bit is set to '1'. The GPMI is idle whenever this bit set to zero. This can be set to one by a ARM platform write. In addition, the DMA sets this bit each time a DMA command has finished its PIO transfer phase. 0x0 **IDLE** — The GPMI is idle. 0x1 **BUSY** — The GPMI is busy running a command.
28 DEV_IRQ_EN	When set to '1' and ATA_IRQ pin is asserted, the GPMI_IRQ output will assert.
27 LOCK_CS	For ATA/NAND mode: 0= Deassert chip select (CS) after RUN is complete. 1= Continue to assert chip select (CS) after RUN is complete. For Camera Mode: 0= Dont wait for VSYNC rising edge before capturing data. 1= Wait for VSYNC rising edge before capturing data (Camera mode only). 0x0 **DISABLED** — Deassert chip select (CS) after RUN is complete. 0x1 **ENABLED** — Continue to assert chip select (CS) after RUN is complete.
26 UDMA	0= Use ATA-PIO mode on the external bus. 1= Use ATA-Ultra DMA mode on the external bus. 0x0 **DISABLED** — Use ATA-PIO mode on the external bus. 0x1 **ENABLED** — Use ATA-Ultra DMA mode on the external bus.
25–24 COMMAND_ MODE	00= Write mode. 01= Read Mode. 10= Read and Compare Mode (setting sense flop). 11= Wait for Ready. 0x0 **WRITE** — Write mode. 0x1 **READ** — Read mode. 0x2 **READ_AND_COMPARE** — Read and Compare mode (setting sense flop). 0x3 **WAIT_FOR_READY** — Wait for Ready mode. For ATA WAIT_FOR_READY command set CS=01.
23 WORD_ LENGTH	0= 16-bit Data Bus Mode. 1= 8-bit Data Bus mode. This bit should only be changed when RUN==0. When GPMI works at DDR interface, that is in Source Synchronous Mode or Samsung Toggle Mode, this bit should be set to 0x0, which is 16-bit Data Bus Mode. 0x0 **16_BIT** — 16-bit Data Bus Mode. 0x1 **8_BIT** — 8-bit Data Bus mode.

Table continues on the next page...

GPMI_CTRL0*n* field descriptions (continued)

Field	Description
22–20 CS	Selects which chip select is active for this command. For ATA WAIT_FOR_READY command, this must be set to b01.
19–17 ADDRESS	Specifies the three address lines for ATA mode. In NAND mode, use A0 for CLE and A1 for ALE. 0x0 **NAND_DATA** — In NAND mode, this address is used to read and write data bytes. 0x1 **NAND_CLE** — In NAND mode, this address is used to write command bytes. 0x2 **NAND_ALE** — In NAND mode, this address is used to write address bytes.
16 ADDRESS_ INCREMENT	0= Address does not increment. 1= Increment address. In ATA mode, the address will increment with each cycle. In NAND mode, the address will increment once, after the first cycle (going from CLE to ALE). 0x0 **DISABLED** — Address does not increment. 0x1 **ENABLED** — Increment address.
15–0 XFER_COUNT	Number of words (8 or 16 bit wide) to transfer for this command. A value of zero will transfer 64K words. When GPMI woks at DDR interface, that is Source Synchronous Mode or Samsung Toggle Mode, XFER_COUNT should be calculated according to the WORD_LENGTH being set to 16 bits.

31.5.2 GPMI Compare Register (GPMI_COMPARE)

The GPMI compare register specifies the expect data and the xor mask for comparing to the status values read from the device. This register is used by the Read and Compare command.

Address: GPMI_COMPARE is 4100_6000h base + 10h offset = 4100_6010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MASK																REFERENCE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

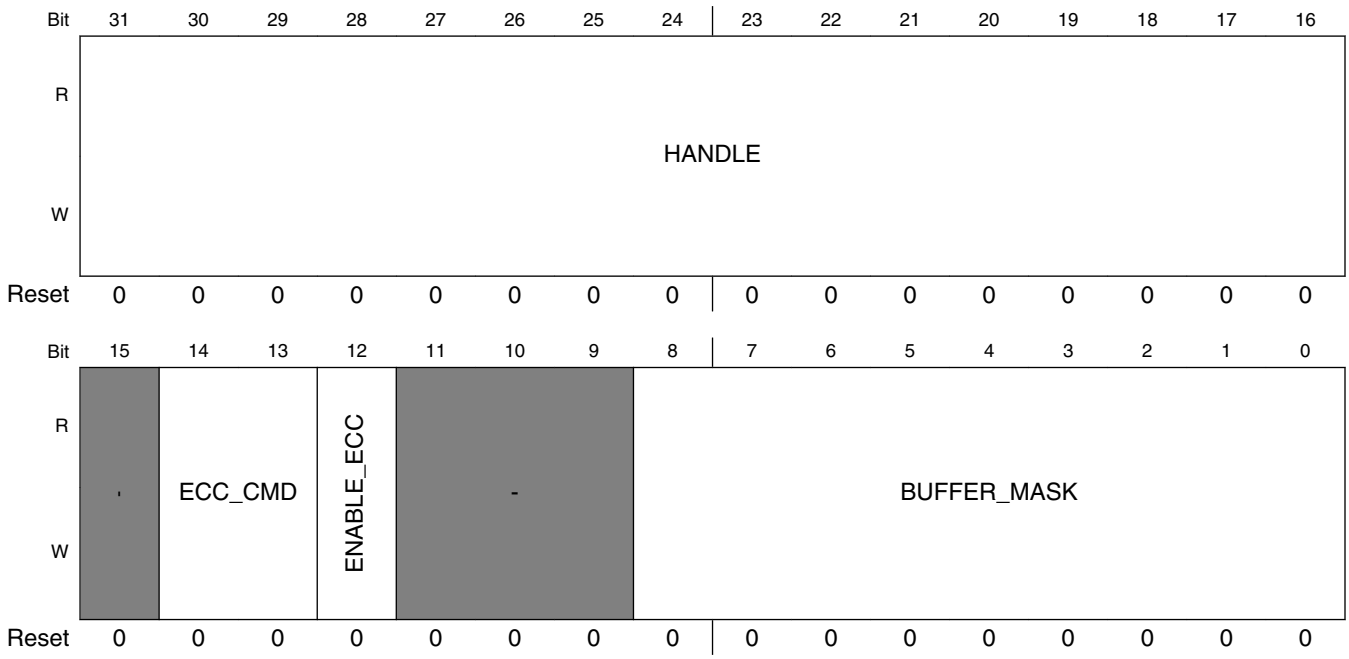
GPMI_COMPARE field descriptions

Field	Description
31–16 MASK	16-bit mask which is applied after the read data is XORed with the REFERENCE bit field.
15–0 REFERENCE	16-bit value which is XORed with data read from the NAND device.

31.5.3 GPMI Integrated ECC Control Register (GPMI_ECCCTRL*n*)

The GPMI ECC control register handles configuration of the integrated ECC accelerator.

Addresses: GPMI_ECCCTRL is 4100_6000h base + 20h offset = 4100_6020h
GPMI_ECCCTRL_SET is 4100_6000h base + 24h offset = 4100_6024h
GPMI_ECCCTRL_CLR is 4100_6000h base + 28h offset = 4100_6028h
GPMI_ECCCTRL_TOG is 4100_6000h base + 2Ch offset = 4100_602Ch



GPMI_ECCCTRL*n* field descriptions

Field	Description
31–16 HANDLE	This is a register available to software to attach an identifier to a transaction in progress. This handle will be available from the ECC register space when the completion interrupt occurs.
15 -	Always write zeroes to this bit field.
14–13 ECC_CMD	ECC Command information. 0x0 **DECODE** — Decode. 0x1 **ENCODE** — Encode. 0x2 **RESERVE2** — Reserved. 0x3 **RESERVE3** — Reserved.
12 ENABLE_ECC	Enable ECC processing of GPMI transfers. 0x1 **ENABLE** — Use integrated ECC for read and write transfers. 0x0 **DISABLE** — Integrated ECC remains in idle.
11–9 -	Always write zeroes to this bit field.

Table continues on the next page...

GPMI_ECCCTRL*n* field descriptions (continued)

Field	Description
8–0 BUFFER_MASK	ECC buffer information. The BCH error correction only allows two configurations of the buffer mask - software may either read just the first block on the flash page or the entire flash page. Write operations must be for the entire flash page. Invalid buffer mask values will cause the DMA descriptor command to be terminated. 0x100 **BCH_AUXONLY** — Set to request transfer from only the auxiliary buffer (block 0 on flash). 0x1FF **BCH_PAGE** — Set to request transfer to/from the entire page.

31.5.4 GPMI Integrated ECC Transfer Count Register (GPMI_ECCCOUNT)

The GPMI ECC Transfer Count Register contains the count of bytes that flow through the ECC subsystem.

Address: GPMI_ECCCOUNT is 4100_6000h base + 30h offset = 4100_6030h

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

GPMI_ECCCOUNT field descriptions

Field	Description
31–16 -	Always write zeroes to this bit field.
15–0 COUNT	Number of bytes to pass through ECC. This is the GPMI transfer count plus the syndrome count that will be inserted into the stream by the ECC. In DMA2ECC_MODE this count must match the GPMI_CTRL0_XFER_COUNT. A value of zero will transfer 64K words. When GPMI works at DDR interface, that is Source Synchronous Mode or Samsung Toggle Mode, XFER_COUNT should be calculated according to the WORD_LENGTH being set to 16 bits.

31.5.5 GPMI Payload Address Register (GPMI_PAYLOAD)

The GPMI payload address register specifies the location of the data buffers in system memory. This value must be word aligned.

Address: GPMI_PAYLOAD is 4100_6000h base + 40h offset = 4100_6040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADDRESS [31:2]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDRESS [31:2]														-	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_PAYLOAD field descriptions

Field	Description
31–2 ADDRESS [31:2]	Pointer to an array of one or more 512 byte payload buffers.
1–0 -	Always write zeroes to this bit field.

31.5.6 GPMI Auxiliary Address Register (GPMI_AUXILIARY)

The GPMI auxiliary address register specifies the location of the auxiliary buffers in system memory. This value must be word aligned.

Address: GPMI_AUXILIARY is 4100_6000h base + 50h offset = 4100_6050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADDRESS[31:2]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDRESS[31:2]														-	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_AUXILIARY field descriptions

Field	Description
31–2 ADDRESS[31:2]	Pointer to ECC control structure and meta-data storage.
1–0 -	Always write zeroes to this bit field.

31.5.7 GPMI Control Register 1 (GPMI_CTRL1*n*)

The GPMI control register 1 specifies additional control fields that are not used on a per-transaction basis.

Addresses: GPMI_CTRL1 is 4100_6000h base + 60h offset = 4100_6060h

GPMI_CTRL1_SET is 4100_6000h base + 64h offset = 4100_6064h

GPMI_CTRL1_CLR is 4100_6000h base + 68h offset = 4100_6068h

GPMI_CTRL1_TOG is 4100_6000h base + 6Ch offset = 4100_606Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DEV_CLK_STOP	SSYNC_CLK_STOP	WRITE_CLK_STOP	TOGGLE_MODE	CLK_DIV2_EN	UPDATE_CS	SSYNCMODE	DECOUPLE_CS	WRN_DLY_SEL		-	TIMEOUT_IRQ_EN	GANGED_RDYBUSY	BCH_MODE	DLL_ENABLE	HALF_PERIOD
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RDN_DELAY				DMA2ECC_MODE	DEV_IRQ	TIMEOUT_IRQ	BURST_EN	ABORT_WAIT_REQUEST	ABORT_WAIT_FOR_READY_CHANNEL			DEV_RESET	ATA_IRQRDY_POLARITY	CAMERA_MODE	MODE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

GPMI_CTRL1*n* field descriptions

Field	Description
31 DEV_CLK_ STOP	set this bit to 1 will stop gpmi io working clk.
30 SSYNC_CLK_ STOP	set this bit to 1 will stop the source synchronous mode clk.

Table continues on the next page...

GPMI_CTRL1*n* field descriptions (continued)

Field	Description
29 WRITE_CLK_STOP	In onfi source synchronous mode, host may save power during the data write cycles by holding the CLK signal high (i.e. stopping the CLK). The host may only stop the CLK during data write, by setting this bit to 1, if the device supports this feature as indicated in the parameter page.
28 TOGGLE_MODE	enable samsung toggle mode.
27 CLK_DIV2_EN	enable the gpmi clk divider. This bit should be reset to 0 in asynchronous mode. The frequence ratio of (device clock : ccm gpmi clock) will be (1 : 1). This bit should be set to 1, in source synchronous mode or toggle mode. The frequence raito of (device clock : ccm gpmi clock) will be (1 : 2). 0x0 **DISABLED** — internal factor-2 clock divider is disabled 0x1 **ENABLED** — internal factor-2 clock divider is enabled
26 UPDATE_CS	force the CS value is be updated to external chip select pin, even GPMI is idle.
25 SSYNCMODE	source synchronouse mode 1 or asynchrous mode 0. 0x0 **ASYNC** — Asynchronous mode. 0x1 **SSYNC** — Source Synchronous mode.
24 DECOUPLE_CS	Decouple Chip Select from DMA Channel. Setting this bit to 1 will allow a DMA channel to specify any value in the CTRL0_CS register field. Software can use one DMA channel to access all 8 Nand devices.
23–22 WRN_DLY_SEL	Since the GPMI write strobe (WRN) is a fast clock pin, the delay on this signal can be programmed to match the load on this pin. 0 = no delay; 1 = 4ns to 8ns; 2 = 6ns to 10ns; 3 = 7ns to 12ns.
21 -	Always write zeroes to this bit field.
20 TIMEOUT_IRQ_EN	Setting this bit to '1' will enable timeout IRQ for transfers in ATA mode only, and for WAIT_FOR_READY commands in both ATA and Nand mode. The Device_Busy_Timeout value is used for this timeout.
19 GANGED_RDYBUSY	Set this bit to 1 will force all Nand RDY_BUSY inputs to be sourced from (tied to) RDY_BUSY0. This will free up all, except one, RDY_BUSY input pins.
18 BCH_MODE	This bit selects which error correction unit will access GPMI. This bit must always be set to '1', since only the BCH unit is available in this design.
17 DLL_ENABLE	Set this bit to 1 to enable the GPMI DLL. This is required for fast NAND reads (above 30MHz read strobe). After setting this bit, wait 64 GPMI clock cycles for the DLL to lock before performing a NAND read.
16 HALF_PERIOD	Set this bit to 1 if the GPMI clock period is greater than 16ns for proper DLL operation. DLL_ENABLE must be zero while changing this field.
15–12 RDN_DELAY	This variable is a factor in the calculated delay to apply to the internal read strobe for correct read data sampling. The applied delay (AD) is between 0 and 1.875 times the reference period (RP). RP is one half of the GPMI clock period if HALF_PERIOD=1

Table continues on the next page...

GPMI_CTRL1*n* field descriptions (continued)

Field	Description
	otherwise it is the full GPMI clock period. The equation is: AD = RDN_DELAY x 0.125 x RP. This value must not exceed 16ns. This variable is used to achieve faster NAND access. For example if the Read Strobe is asserted from time 0 to 13ns but the read access time is 20ns, then choose AD=12ns will cause the data to be sampled at time 25ns (13+12) giving a 5ns data setup time. If RP=13ns then RDN_DELAY = 12/(0.125 x 13ns) = 7.38 (0111b). DLL_ENABLE must be zero while changing this field.
11 DMA2ECC_ MODE	This is mainly for testing HWECC without involving the Nand device. Setting this bit will cause DMA write data to redirected to HWECC module (instead of Nand Device) for encoding or decoding.
10 DEV_IRQ	This bit is set when an Interrupt is received from the ATA device. Write 0 to clear.
9 TIMEOUT_IRQ	This bit is set when a timeout occurs using the Device_Busy_Timeout value. Write 0 to clear.
8 BURST_EN	When set to 1 each DMA request will generate a 4-transfer burst on the APB bus.
7 ABORT_WAIT_ REQUEST	Request to abort "wait for ready" command on channel indicated by ABORT_WAIT_FOR_READY_CHANNEL. Hardware will clear this bit when abort is done.
6–4 ABORT_WAIT_ FOR_READY_ CHANNEL	Abort a wait for ready command on selected channel. Set the ABORT_WAIT_REQUEST to kick of operation.
3 DEV_RESET	0= Device Reset pin is held low (asserted). 1= Device Reset pin is held high (de-asserted). 0x0 **ENABLED** — Device Reset pin is held low (asserted). 0x1 **DISABLED** — Device Reset pin is held high (de-asserted).
2 ATA_IRQRDY_ POLARITY	For ATA MODE: 0= External ATA IORDY and IRQ are active low. 1= External ATA IORDY and IRQ are active high. For NAND MODE: 0= External RDY_BUSY[1] and RDY_BUSY[0] pins are ready when low and busy when high. 1= External RDY_BUSY[1] and RDY_BUSY[0] pins are ready when high and busy when low. Note NAND_RDY_BUSY[3:2] are not affected by this bit. 0x0 **ACTIVELOW** — ATA IORDY and IRQ are active low, or NAND_RDY_BUSY[1:0] are active low ready. 0x1 **ACTIVEHIGH** — ATA IORDY and IRQ are active high, or NAND_RDY_BUSY[1:0] are active high ready.
1 CAMERA_ MODE	When set to 1 and ATA UDMA is enabled the UDMA interface becomes a camera interface.

Table continues on the next page...

GPMI_CTRL1*n* field descriptions (continued)

Field	Description
0 MODE	0= NAND mode. 1= ATA mode. ATA mode is only supported on channel zero. If ATA mode is selected, then only channel three is available for NAND use. 0x0 **NAND** — NAND mode. 0x1 **ATA** — ATA mode.

31.5.8 GPMI Timing Register 0 (GPMI_TIMING0)

The GPMI timing register 0 specifies the timing parameters that are used by the cycle state machine to guarantee the various setup, hold and cycle times for the external media type.

Address: GPMI_TIMING0 is 4100_6000h base + 70h offset = 4100_6070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								ADDRESS_SETUP								DATA_HOLD								DATA_SETUP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

GPMI_TIMING0 field descriptions

Field	Description
31–24 -	Always write zeroes to this bit field.
23–16 ADDRESS_ SETUP	Number of GPMICLK cycles that the CE/ADDR signals are active before a strobe is asserted. A value of zero is interpreted as 0. For ATA PIO modes this is known in the ATA7 specification as "Address valid to DIOR-/DIOW- setup"
15–8 DATA_HOLD	Data bus hold time in GPMICLK cycles. Also the time that the data strobe is de-asserted in a cycle. A value of zero is interpreted as 256. For ATA PIO modes this is known in the ATA7 specification as "DIOR-/DIOW- recovery time"
7–0 DATA_SETUP	Data bus setup time in GPMICLK cycles. Also the time that the data strobe is asserted in a cycle. This value must be greater than 2 for ATA devices that use IORDY to extend transfer cycles. A value of zero is interpreted as 256. For ATA PIO modes this is known in the ATA7 specification as ""DIOR-/DIOW-"

31.5.9 GPMI Timing Register 1 (GPMI_TIMING1)

The GPMI timing register 1 specifies the timeouts used when monitoring the NAND READY pin or the ATA IRQ and IOWAIT signals.

Address: GPMI_TIMING1 is 4100_6000h base + 80h offset = 4100_6080h

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

GPMI_TIMING1 field descriptions

Field	Description
31–16 DEVICE_BUSY_TIMEOUT	Timeout waiting for NAND Ready/Busy or ATA IRQ. Used in WAIT_FOR_READY mode. This value is the number of GPMI_CLK cycles multiplied by 4096.
15–0 -	Always write zeroes to this bit field.

31.5.10 GPMI Timing Register 2 (GPMI_TIMING2)

The GPMI timing register 2 specifies the double data rate timing parameters that are used by the cycle state machine to guarantee the various cs delay, pre-amble delay, post-amble delay, command/address delay, data delay and read latency cycle times for the external media type.

Address: GPMI_TIMING2 is 4100_6000h base + 90h offset = 4100_6090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-					READ_ LATENCY			-			CE_DELAY					PREAMBLE_ DELAY				POSTAMBLE_ DELAY				CMDADD_ PAUSE				DATA_ PAUSE			
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	0

GPMI_TIMING2 field descriptions

Field	Description
31–27 -	Always write zeroes to this bit field.
26–24 READ_ LATENCY	This field is for double data rate read latency configuration. 3'b000 : READ LATENCY is 0 3'b001 : READ LATENCY is 1 3'b010 : READ LATENCY is 2

Table continues on the next page...

GPMI_TIMING2 field descriptions (continued)

Field	Description
	3'b011 : READ LATENCY is 3 3'b100 : READ LATENCY is 4 3'b101 : READ LATENCY is 5 others : READ LATENCY is 3
23–21 -	Always write zeroes to this bit field.
20–16 CE_DELAY	GPMI dealy from CEn assert to W/Rn changing edge. value of zero is interpreted as 32.
15–12 PREAMBLE_ DELAY	GPMI pre-amble delay in GPMICLK cycles. A value of zero is interpreted as 16.
11–8 POSTAMBLE_ DELAY	GPMI post-amble delay in GPMICLK cycles. A value of zero is interpreted as 16.
7–4 CMDADD_ PAUSE	GPMI delay time from command or addres pause to command or address resume in GPMICLK cycles. A value of zero is interpreted as 16.
3–0 DATA_PAUSE	GPMI delay time from data pause to data resume in GPMICLK cycles. A value of zero is interpreted as 16.

31.5.11 GPMI DMA Data Transfer Register (GPMI_DATA)

The GPMI DMA data transfer register is used by the DMA to read or write data to or from the ATA/NAND control state machine.

Address: GPMI_DATA is 4100_6000h base + A0h offset = 4100_60A0h

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

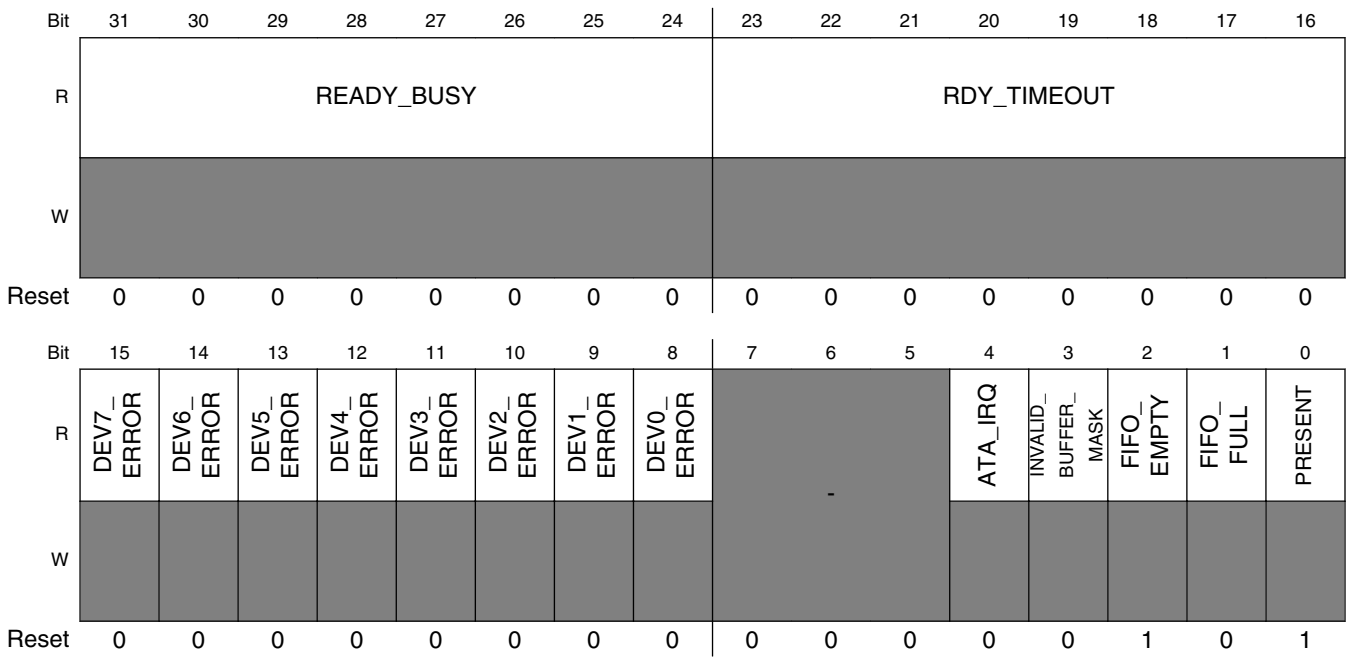
GPMI_DATA field descriptions

Field	Description
31–0 DATA	In 16-bit mode, this register can be accessed in two 16-bit operations, one bus cycle per operation. In 8-bit mode, one, two, three or four bytes can can be accessed to send the same number of bus cycles.

31.5.12 GPMI Status Register (GPMI_STAT)

The GPMI control and status register provides a read back path for various operational states of the GPMI controller.

Address: GPMI_STAT is 4100_6000h base + B0h offset = 4100_60B0h



GPMI_STAT field descriptions

Field	Description
31–24 READY_BUSY	Read-only view of NAND Ready_Busy Input pins.
23–16 RDY_TIMEOUT	State of the RDY/BUSY Timeout Flags. When any bit is set to '1' in this field, it indicates that a time out has occurred while waiting for the ready state of the requested NAND device. Multiple bits may be set simultaneously. When GPMI_CTRL1_DECOUPLE_CS = 0, RDY_TIMEOUT[n] is associated with the NAND device on chip_select[n]. When GPMI_CTRL1_DECOUPLE_CS = 1, these flags become associated to a DMA channel instead of a NAND device. For example if DMA channel 6 sends a WAIT_FOR_READY command for NAND Device 2, and a timeout occured on READY_BUSY2, then READY_TIMEOUT[6] will be set instead of READY_TIMEOUT[2].
15 DEV7_ERROR	0= No error condition present on ATA/NAND Device accessed by DMA channel 7. 1= An Error has occurred on ATA/NAND Device accessed by DMA channel 7 (Timeout or compare failure, depending on COMMAND_MODE).
14 DEV6_ERROR	0= No error condition present on ATA/NAND Device accessed by DMA channel 6.

Table continues on the next page...

GPMI_STAT field descriptions (continued)

Field	Description
	1= An Error has occurred on ATA/NAND Device accessed by DMA channel 6 (Timeout or compare failure, depending on COMMAND_MODE).
13 DEV5_ERROR	0= No error condition present on ATA/NAND Device accessed by DMA channel 5. 1= An Error has occurred on ATA/NAND Device accessed by DMA channel 5 (Timeout or compare failure, depending on COMMAND_MODE).
12 DEV4_ERROR	0= No error condition present on ATA/NAND Device accessed by DMA channel 4. 1= An Error has occurred on ATA/NAND Device accessed by DMA channel 4 (Timeout or compare failure, depending on COMMAND_MODE).
11 DEV3_ERROR	0= No error condition present on ATA/NAND Device accessed by DMA channel 3. 1= An Error has occurred on ATA/NAND Device accessed by DMA channel 3 (Timeout or compare failure, depending on COMMAND_MODE).
10 DEV2_ERROR	0= No error condition present on ATA/NAND Device accessed by DMA channel 2. 1= An Error has occurred on ATA/NAND Device accessed by DMA channel 2 (Timeout or compare failure, depending on COMMAND_MODE).
9 DEV1_ERROR	0= No error condition present on ATA/NAND Device accessed by DMA channel 1. 1= An Error has occurred on ATA/NAND Device accessed by DMA channel 1 (Timeout or compare failure, depending on COMMAND_MODE).
8 DEV0_ERROR	0= No error condition present on ATA/NAND Device accessed by DMA channel 0. 1= An Error has occurred on ATA/NAND Device accessed by DMA channel 0 (Timeout or compare failure, depending on COMMAND_MODE).
7–5 -	Always write zeroes to this bit field.
4 ATA_IRQ	Status of the ATA_IRQ input pin.
3 INVALID_ BUFFER_MASK	0= ECC Buffer Mask is not invalid. 1= ECC Buffer Mask is invalid.
2 FIFO_EMPTY	0= FIFO is not empty. 1= FIFO is empty. 0x0 **NOT_EMPTY** — FIFO is not empty. 0x1 **EMPTY** — FIFO is empty.
1 FIFO_FULL	0= FIFO is not full. 1= FIFO is full. 0x0 **NOT_FULL** — FIFO is not full. 0x1 **FULL** — FIFO is full.
0 PRESENT	0= GPMI is not present in this product. 1= GPMI is present is in this product. 0x0 **UNAVAILABLE** — GPMI is not present in this product. 0x1 **AVAILABLE** — GPMI is present in this product.

31.5.13 GPMI Debug Information Register (GPMI_DEBUG)

The GPMI debug information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

Address: GPMI_DEBUG is 4100_6000h base + C0h offset = 4100_60C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WAIT_FOR_READY_END								DMA_SENSE								DMAREQ								CMD_END							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_DEBUG field descriptions

Field	Description
31–24 WAIT_FOR_ READY_END	Read Only view of the Wait_For_Ready End toggle signals to DMA. One per channel
23–16 DMA_SENSE	Read-only view of sense state of the 8 DMA channels. A value of "1" in any bit position indicates that a read and compare command failed or a timeout occured for the corresponding channel.
15–8 DMAREQ	Read-only view of DMA request line for 8 DMA channels. A toggle on any bit position indicates a DMA request for the corresponding channel.
7–0 CMD_END	Read Only view of the Command End toggle signals to DMA. One per channel

31.5.14 GPMI Version Register (GPMI_VERSION)

This register reflects the version number for the GPMI.

Address: GPMI_VERSION is 4100_6000h base + D0h offset = 4100_60D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

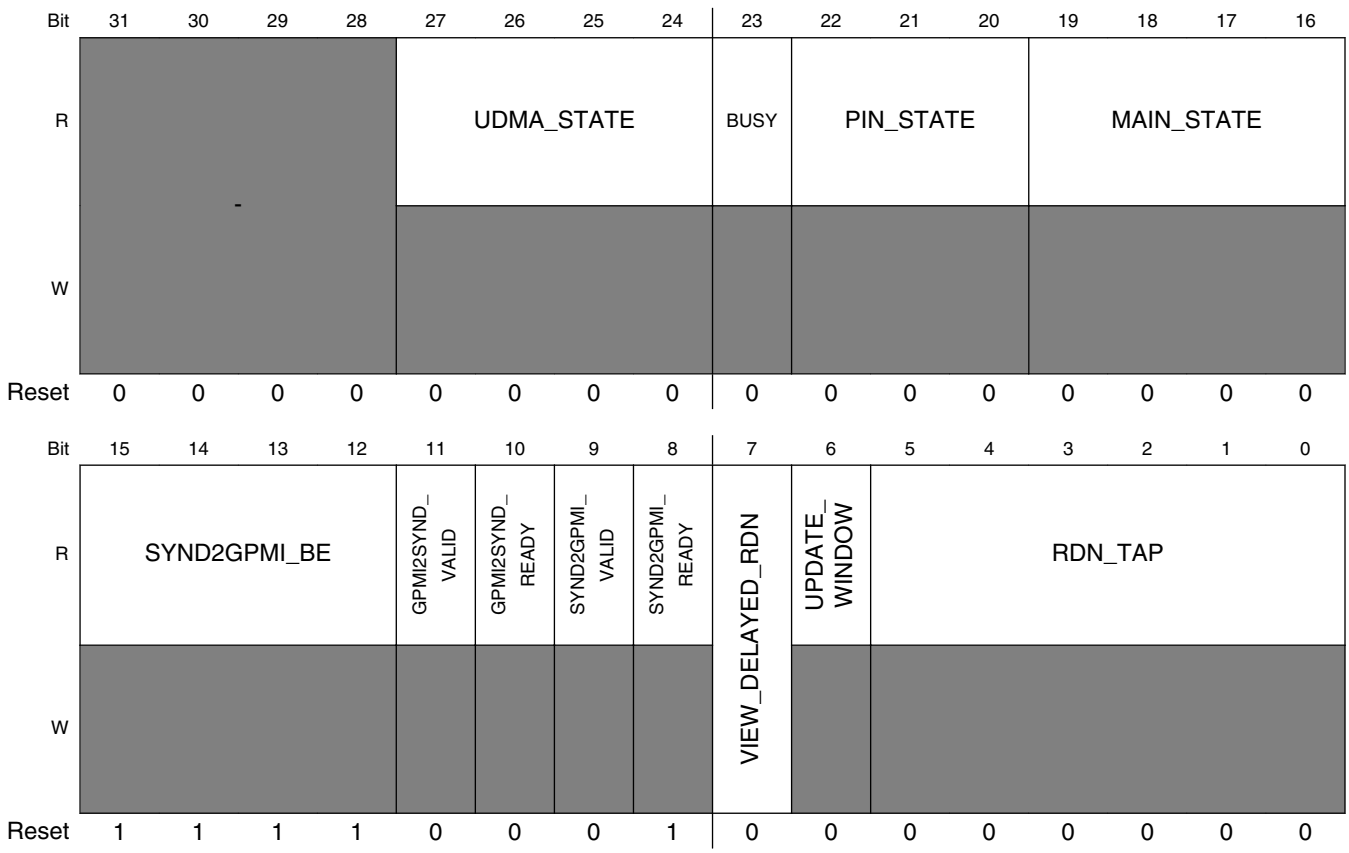
GPMI_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

31.5.15 GPMI Debug2 Information Register (GPMI_DEBUG2)

The GPMI Debug2 information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

Address: GPMI_DEBUG2 is 4100_6000h base + E0h offset = 4100_60E0h



GPMI_DEBUG2 field descriptions

Field	Description
31–28 -	Always write zeroes to this bit field.
27–24 UDMA_STATE	USM_IDLE = 4'h0, idle USM_DMARQ = 4'h1, DMA req USM_ACK = 4'h2, DMA ACK USM_FIFO_E = 4'h3, Fifo empty USM_WPAUSE = 4'h4, WR DMA Paused by device USM_TSTRB = 4'h5, Toggle HSTROBE USM_CAPTUR = 4'h6, Capture Stage, (data sampled with DSTROBE is valid) USM_DATOUT = 4'h7, Change Burst DATAOUT USM_CRC = 4'h8, Source CRC to Device

Table continues on the next page...

GPMI_DEBUG2 field descriptions (continued)

Field	Description
	USM_WAIT_R = 4'h9, Waiting for DDMARDY- USM_END = 4'ha; Negate DMAACK (end of DMA) USM_WAIT_S = 4'hb, Waiting for DSTROBE USM_RPAUSE = 4'hc, Rd DMA Paused by Host USM_RSTOP = 4'hd, Rd DMA Stopped by Host USM_WTERM = 4'he, Wr DMA Termination State USM_RTERM = 4'hf, Rd DMA Termination state
23 BUSY	When asserted the GPMI is busy. Undefined results may occur if any registers are written when BUSY is asserted. 0x0 **DISABLED** — The GPMI is not busy. 0x1 **ENABLED** — The GPMI is busy.
22–20 PIN_STATE	parameter PSM_IDLE = 3'h0, PSM_BYTCNT = 3'h1, PSM_ADDR = 3'h2, PSM_STALL = 3'h3, PSM_STROBE = 3'h4, PSM_ATARDY = 3'h5, PSM_DHOLD = 3'h6, PSM_DONE = 3'h7. 0x0 **PSM_IDLE** — 0x1 **PSM_BYTCNT** — 0x2 **PSM_ADDR** — 0x3 **PSM_STALL** — 0x4 **PSM_STROBE** — 0x5 **PSM_ATARDY** — 0x6 **PSM_DHOLD** — 0x7 **PSM_DONE** —
19–16 MAIN_STATE	parameter MSM_IDLE = 4'h0, MSM_BYTCNT = 4'h1, MSM_WAITFE = 4'h2, MSM_WAITFR = 4'h3, MSM_DMAREQ = 4'h4, MSM_DMAACK = 4'h5, MSM_WAITFF = 4'h6, MSM_LDFIFO = 4'h7, MSM_LDDMAR = 4'h8, MSM_RDCMP = 4'h9, MSM_DONE = 4'hA. 0x0 **MSM_IDLE** — 0x1 **MSM_BYTCNT** — 0x2 **MSM_WAITFE** — 0x3 **MSM_WAITFR** — 0x4 **MSM_DMAREQ** — 0x5 **MSM_DMAACK** — 0x6 **MSM_WAITFF** — 0x7 **MSM_LDFIFO** — 0x8 **MSM_LDDMAR** — 0x9 **MSM_RDCMP** — 0xA **MSM_DONE** —
15–12 SYND2GPMI_BE	Data byte enable Input from BCH.
11 GPMI2SYND_VALID	Data handshake output to BCH.

Table continues on the next page...

GPMI_DEBUG2 field descriptions (continued)

Field	Description
10 GPMI2SYND_ READY	Data handshake output to BCH.
9 SYND2GPMI_ VALID	Data handshake Input from BCH.
8 SYND2GPMI_ READY	Data handshake Input from BCH.
7 VIEW_ DELAYED_RDN	Set to a 1 to select the delayed feedback RDN to drive the GPMI_ADDR[0] (Nand CLE) pin. For debug purposes, this will allow you see if DLL is functioning properly.
6 UPDATE_ WINDOW	A 1 indicates that the DLL is busy generating the required delay.
5–0 RDN_TAP	This is the DLL tap calculated by the DLL controller. The selects the amount of delay form the DLL chain.

31.5.16 GPMI Debug3 Information Register (GPMI_DEBUG3)

The GPMI Debug3 information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

Address: GPMI_DEBUG3 is 4100_6000h base + F0h offset = 4100_60F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_WORD_CNTR																DEV_WORD_CNTR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPMI_DEBUG3 field descriptions

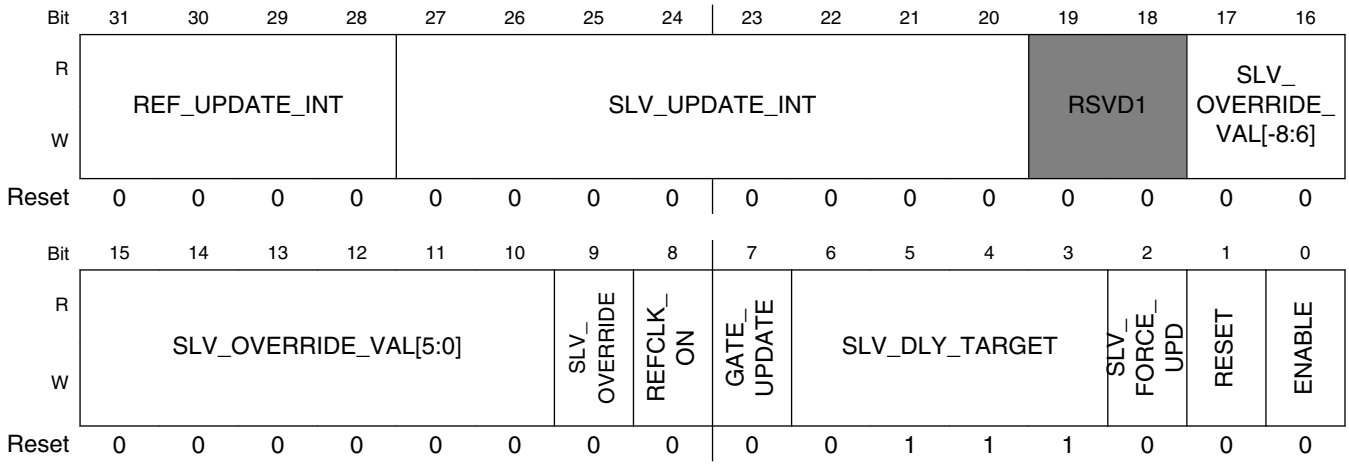
Field	Description
31–16 APB_WORD_ CNTR	Reflects the number of words (16 or 8-bit) remains to be transferred on the APB bus.
15–0 DEV_WORD_ CNTR	Reflects the number of words (16 or 8-bit) remains to be transferred on the ATA/Nand bus.

31.5.17 GPMI Double Rate Read DLL Control Register (GPMI_READ_DDR_DLL_CTRL)

GPMI DDR Read Delay Loop Lock Control Register.

This register provides programmability in DDR mode for data input timing and data formats.

Address: GPMI_READ_DDR_DLL_CTRL is 4100_6000h base + 100h offset = 4100_6100h



GPMI_READ_DDR_DLL_CTRL field descriptions

Field	Description
31–28 REF_UPDATE_ INT	This field allows the user to add additional delay cycles to the DLL control loop (reference delay line control). By default, the DLL control loop shall update every two SSPCLK cycles. Programming this field results in a DLL control loop update interval of (2 + REF_UPDATE_INT) * SSPCLK. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 SLV_UPDATE_ INT	Setting a value greater than 0 in this field, shall over-ride the default slave delay-line update interval of 256 SSPCLK cycles. A value of 0 results in an update interval of 256 SSPCLK cycles (default setting). A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–18 RSVD1	Reserved
17–10 SLV_ OVERRIDE_VAL	When SLV_OVERRIDE=1 This field is used to select 1 of 256 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 256.
9 SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to tbe enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0
8 REFCLK_ON	set this bit to 1 will turn on the reference clock

Table continues on the next page...

GPMI_READ_DDR_DLL_CTRL field descriptions (continued)

Field	Description
7 GATE_UPDATE	Setting this bit to 1, forces the slave delay line not update
6–3 SLV_DLY_ TARGET	The delay target for the SSP read clock is can be programmed in 1/16th increments of an SSPCLK half-period. So the input read-clock can be delayed relative input data from (SSPCLK/2)/16 to SSPCLK/2.
2 SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered).
1 RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an SSPCLK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

31.5.18 GPMI Double Rate Write DLL Control Register (GPMI_WRITE_DDR_DLL_CTRL)

GPMI DDR Write Delay Loop Lock Control Register.

This register provides programmability in DDR mode for data output timing and data formats.

Address: GPMI_WRITE_DDR_DLL_CTRL is 4100_6000h base + 110h offset = 4100_6110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REF_UPDATE_INT				SLV_UPDATE_INT								-		SLV_ OVERRIDE_ VAL[-8:6]	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SLV_OVERRIDE_VAL[5:0]						SLV_ OVERRIDE	REFCLK_ ON	GATE_ UPDATE	SLV_DLY_TARGET				SLV_ FORCE_ UPD	RESET	ENABLE
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

GPMI_WRITE_DDR_DLL_CTRL field descriptions

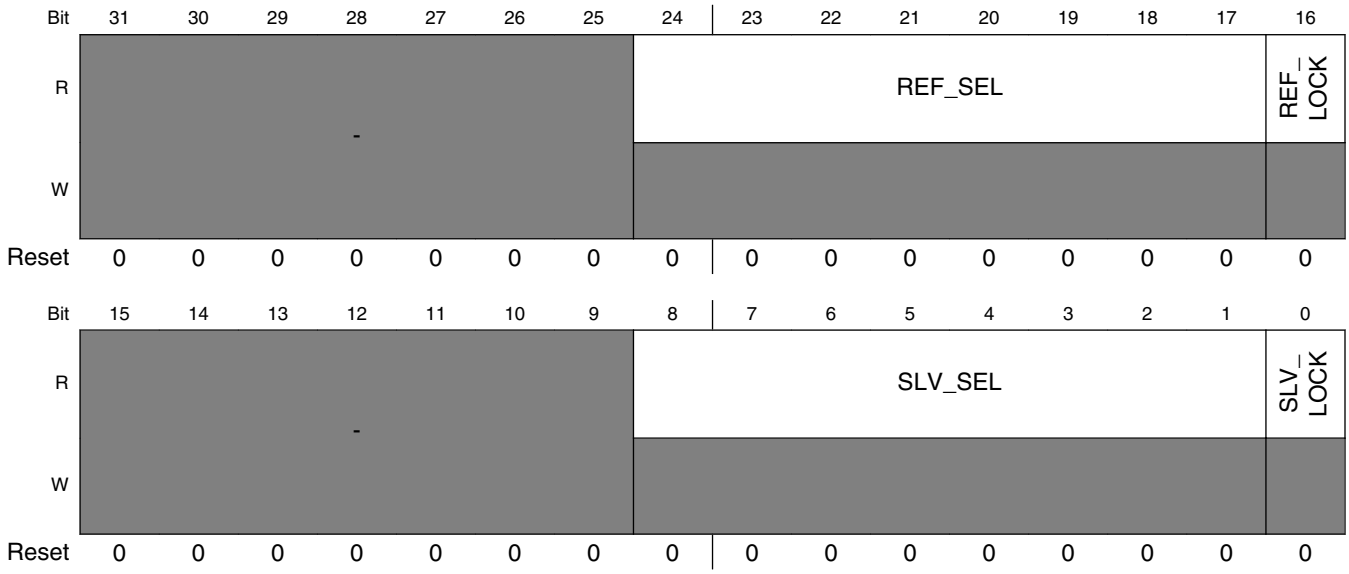
Field	Description
31–28 REF_UPDATE_INT	This field allows the user to add additional delay cycles to the DLL control loop (reference delay line control). By default, the DLL control loop shall update every two SSPCLK cycles. Programming this field results in a DLL control loop update interval of (2 + REF_UPDATE_INT) * SSPCLK. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 SLV_UPDATE_INT	Setting a value greater than 0 in this field, shall over-ride the default slave delay-line update interval of 256 SSPCLK cycles. A value of 0 results in an update interval of 256 SSPCLK cycles (default setting). A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–18 -	Reserved
17–10 SLV_OVERRIDE_VAL	When SLV_OVERRIDE=1 This field is used to select 1 of 256 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 256.
9 SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to tbe enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0
8 REFCLK_ON	set this bit to 1 will turn on the reference clock
7 GATE_UPDATE	Setting this bit to 1, forces the slave delay line not update
6–3 SLV_DLY_TARGET	The delay target for the SSP read clock is can be programmed in 1/16th increments of an SSPCLK half-period. So the input read-clock can be delayed relative input data from (SSPCLK/2)/16 to SSPCLK/2.
2 SLV_FORCE_UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered).
1 RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an SSPCLK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

31.5.19 GPMI Double Rate Read DLL Status Register (GPMI_READ_DDR_DLL_STS)

GPMI Double Rate Read DLL Status Register, Read Only.

GPMI DLL status fields are provided in this register.

Address: GPMI_READ_DDR_DLL_STS is 4100_6000h base + 120h offset = 4100_6120h



GPMI_READ_DDR_DLL_STS field descriptions

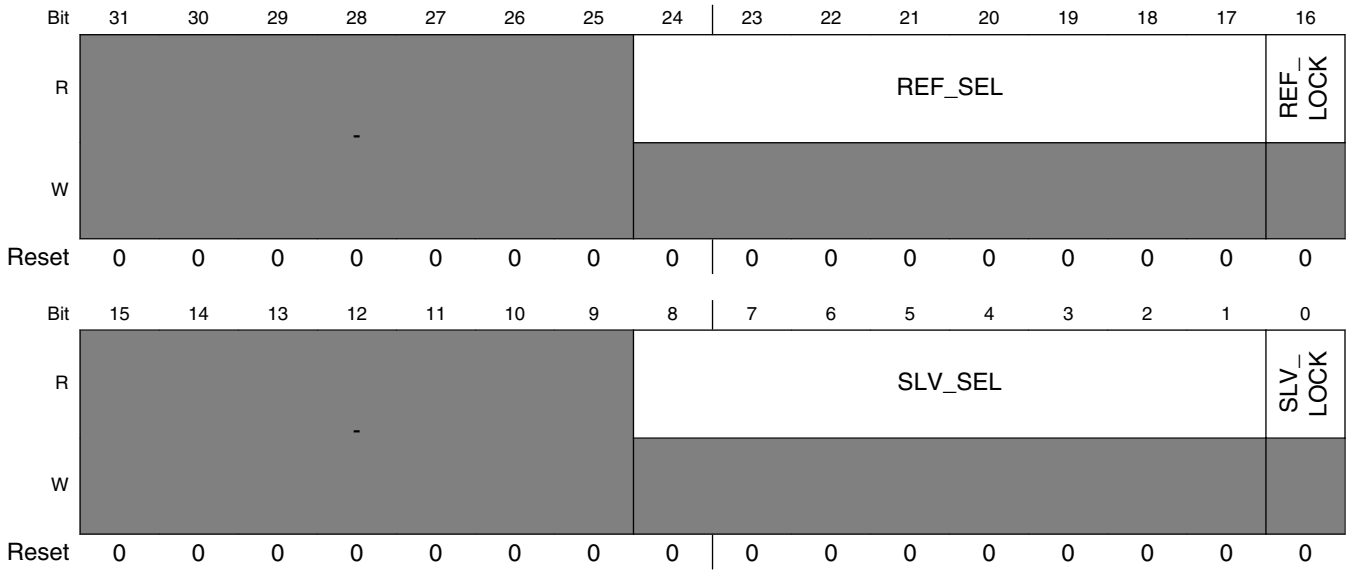
Field	Description
31–25 -	Reserved
24–17 REF_SEL	Reference delay line select status.
16 REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase GPMICLK shift, allowing the slave delay-line to perform programmed clock delays.
15–9 -	Reserved
8–1 SLV_SEL	Slave delay line select status
0 SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.

31.5.20 GPMI Double Rate Write DLL Status Register (GPMI_WRITE_DDR_DLL_STS)

GPMI Double Rate Write DLL Status Register, Read Only.

GPMI DLL status fields are provided in this register.

Address: GPMI_WRITE_DDR_DLL_STS is 4100_6000h base + 130h offset = 4100_6130h



GPMI_WRITE_DDR_DLL_STS field descriptions

Field	Description
31–25 -	Reserved
24–17 REF_SEL	Reference delay line select status.
16 REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase GPMICLK shift, allowing the slave delay-line to perform programmed clock delays.
15–9 -	Reserved
8–1 SLV_SEL	Slave delay line select status
0 SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.

Chapter 32
General Purpose Timer (GPT)

32.1 Overview

This chapter describes the General Purpose Timer (GPT) module interface. It is also a reference for software driver programming. The GPT has a 32-bit up-counter. The timer counter value can be captured in a register using an event on an external pin. The capture trigger can be programmed to be a rising or/and falling edge. The GPT can also generate an event on the DO_CMPOUT*n* pins and an interrupt when the timer reaches a programmed value. The GPT has a 12-bit prescaler, which provides a programmable clock frequency derived from multiple clock sources.

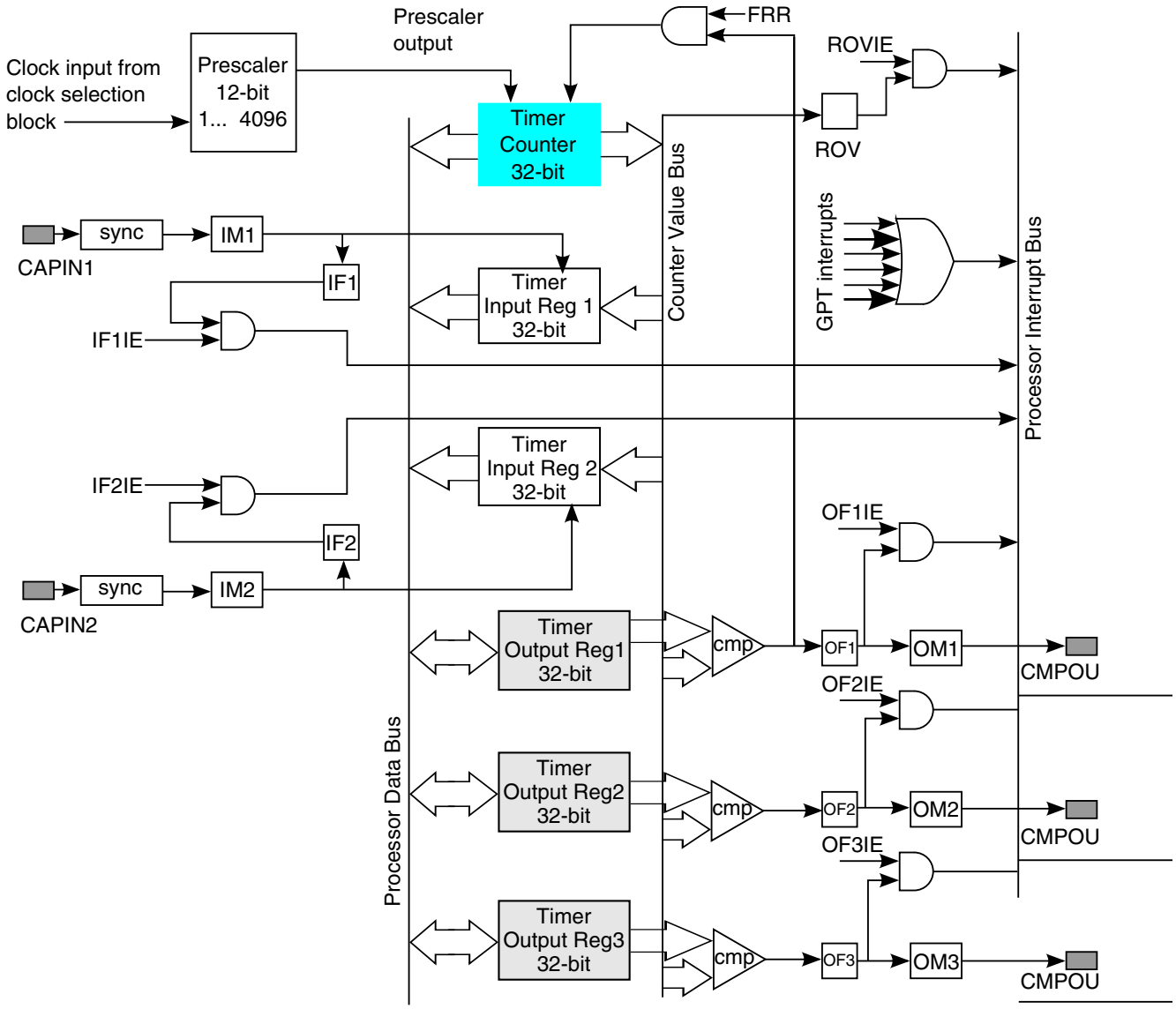


Figure 32-1. GPT Block Diagram

The following figure shows the GPT functional clocking scheme.

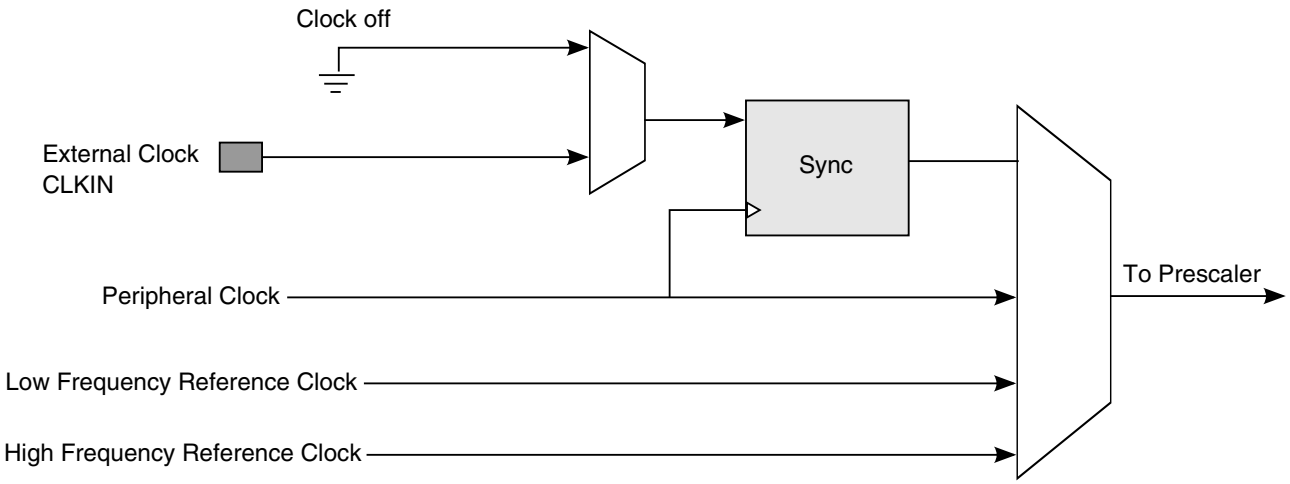


Figure 32-2. GPT Counter Clocks Diagram

32.1.1 Features

- One 32-bit up-counter with clock source selection, including external clock.
- Two input capture channels with a programmable trigger edge.
- Three output compare channels with a programmable output mode. A "forced compare" feature is also available.
- Can be programmed to be *active* in low power and debug modes.
- Interrupt generation at capture, compare, and rollover events.
- Restart or free-run modes for counter operations.

32.1.2 Modes and Operation

The GPT supports the modes described in the indicated sections:

- Operating Modes
  - Restart Mode
  - Free-Run Mode

32.2 External Signals

The GPT follows the IP Bus protocol for interfacing with the processor core. The GPT does not have *any interface signals with any other module inside the chip,* except for the clock and reset inputs (from the clock and reset controller module) and for the interrupt signals *to* the processor interrupt handler. There are functional and clock inputs, and functional output signals going outside the chip boundary.

The following table describes all block signals that connect off-chip.

Table 32-1. Off-Chip Module Signals

Name	Direction	Function	Reset State	Pull-Up
CLKIN	I	Input pin for an external clock that the counter can be operated at.	-	Passive Hysteresis
CAPIN1	I	Input pin for a capture event for Input Capture Channel 1.	-	Passive
CAPIN2	I	Input pin for a capture event for Input Capture Channel 2.	-	Passive
CMPOUT1	O	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1.	0	Passive
CMPOUT2	O	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2.	0	Passive
CMPOUT3	O	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3.	0	Passive

There are six signals (three input, three output) in the GPT module that *can be* connected to the chip pads.

32.2.1 External Clock Input: CLKIN

The GPT counter can be operated using an external clock from outside the device, and this is the input pin used for that purpose. This clock is treated as asynchronous to the peripheral clock. To ensure proper operations of GPT, the external clock input frequency should be less than 1/4 of frequency of the peripheral clock. Hysteresis characteristics on this pad will be required because this is a clock input.

32.2.2 Input Capture Trigger Signals: CAPIN1, CAPIN2

The GPT counter value can be stored in a register, triggered by an event from *outside the device*. A positive or/and negative edge on these signals can trigger this capture event. These signals are treated as asynchronous to the peripheral clock. Only those transitions which occur *at least a single clock cycle* (the clock selected to run the counter) *after the previous recorded transition* are guaranteed to trigger a capture event.

32.2.3 Output Compare Signals: DO_CMPOUT1, DO_CMPOUT2, DO_CMPOUT3

The output compare signals indicate that output compare events have gone through a specified transition.

32.3 Functional Description

This section provides a complete functional description of the GPT.

32.3.1 Operating Modes

The GPT counter can be programmed to work in either of two modes: Restart mode or Free-Run mode.

32.3.1.1 Restart Mode

In Restart mode (selectable through the GPT Control Register GPT_CR), when the counter reaches the compared value, the counter resets and starts again from 0x00000000. The Restart feature is associated only with Compare Channel 1.

Any write access to the Compare register of Channel 1 will reset the GPT counter. This is done to avoid possibly missing a compare event when compare value is changed from a higher value to lower value while counting is proceeding.

For the other two compare channels, when the compare event occurs the counter is *not reset*.

32.3.1.2 Free-Run Mode

In Free-Run mode, when compare events occur for all 3 channels, the counter is *not reset*; instead the counter continues to count until 0xffffffff, and then rolls over (to 0x00000000).

32.3.2 Operation

The General Purpose Timer (GPT) has a single counter (GPT_CNT) that is a 32-bit free-running *up-counter,* which starts counting *after it is enabled by software* (EN=1).

The counter's clock source is the output of the prescaler labelled "Prescaler output" in Figure 32-1.

- If the GPT timer is disabled (EN=0), then the Main Counter *and* Prescaler Counter freeze their current count values. The ENMOD bit determines the value of the GPT counter when the EN bit is set and the Counter is enabled again.
  - If the ENMOD bit is set (=1), then the Main Counter and Prescaler Counter values are reset to 0, when GPT is enabled (EN=1).
  - If ENMOD bit is programmed to 0, then the Main Counter and Prescaler Counter restart counting from their frozen values, when GPT is enabled again (EN=1).
- If GPT is programmed to be disabled in a low power mode (STOP/WAIT), then the Main Counter and Prescaler Counter freeze at their current count values *when* GPT enters low power mode. When GPT exits a low power mode, the Main Counter and Prescaler Counter start counting from their frozen values *regardless* of the ENMOD bit value. Note that the GPT_CNT can be read *at any time* by the processor, and that *both* Input Capture Channels use the *same* counter (GPT_CNT).
- A hardware reset resets all the GPT registers to their respective reset values. All registers except the Output Compare Registers (OCR1, OCR2, OCR3) obtain a value of 0x0. The Compare registers are reset to 0xffffffff.
- The software reset (SWR bit in the GPT_CR control register) resets *all* of the register bits *except* the EN, ENMOD, STOPEN, WAITEN, and DBGEN bits. The state of these bits is not affected by a software reset. Note that a software reset can be given *while the GPT is disabled*.

32.3.2.1 Clocks

The clock that is input to the prescaler can be selected from 4 clock sources:

- High Frequency Clock

Provided by the Clock Controller Module (CCM), the High Frequency Clock is intended to be ON in Normal Power mode when the Peripheral Clock is turned OFF, thereby enabling the GPT to be operated using the High Frequency Clock *in Normal Power mode*. The CCM is expected to provide this clock *after* synchronizing it to the System Bus Clock in Normal functional mode; the CCM is also expected to switch to the *unsynchronized* version of the High Frequency Clock in a Low Power mode.

- Low Reference Clock

  This 32 kHz Low Reference Clock (provided by the CCM) is intended to be ON in Low Power mode when the Peripheral Clock is turned OFF, thereby enabling the GPT to be operated using the Low Reference Clock in Low Power mode. The CCM is expected to provide the Low Reference Clock *after* synchronizing it to the System Bus Clock in Normal functional mode; the CCM is also expected to switch to the *unsynchronized* version of the Low Reference Clock in a Low Power mode.

- External Clock

  The External Clock comes from *outside the device* and can be selected to run the GPT counter. The External Clock is treated as *asynchronous to the Peripheral Clock,* and is synchronized to the Peripheral Clock, *inside* the module. Therefore, the External Clock frequency is limited to < 1/4 frequency of the Peripheral Clock, for proper GPT operations. Note that in Low Power modes, *if* the Peripheral Clock is not available, then the External Clock *cannot be used* to run the counter.

- Peripheral Clock

  If the Peripheral Clock or the External Clock is selected(CLKSRC=001 or 011) as Clock Source, then the Peripheral Clock will be ON in normal GPT operations. In Low Power modes, if the GPT is programmed to be disabled (STOPEN or WAITEN or DOZEN=0), then the Peripheral Clock can be switched OFF.

- Crystal Oscillator Clock

  This 24MHz Crystal Oscillator Clock (provided by the CCM) is intended to be used against frequency change of Peripheral Clock changes to provide a more accruate timer clock for operation system. The CCM is expected to provide the 24M Crystal Oscillator Clock *without* synchronizing it to the System Bus Clock in Normal functional mode. Synchronization is done in GPT module. Before synchronization, the 24M Crystal Oscillator Clock is divided by a 24M clock prescaler, to make sure the clock frequency less than half of System Bus Clock .

The clock input source is configured using the clock source field (CLKSRC, in the GPT_CR control register). The clock input to the prescaler can be disabled by programming the CLKSRC bits (of the GPT_CR control register) to 000. **The CLKSRC field value should be changed** *only after disabling the GPT* (by setting the EN bit in the GPT_CR to 0).

The PRESCALER field selects the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value (from 1 to 4096) and can be changed *at any time*. A change in the value of the PRESCALER field *immediately affects* the output clock frequency.

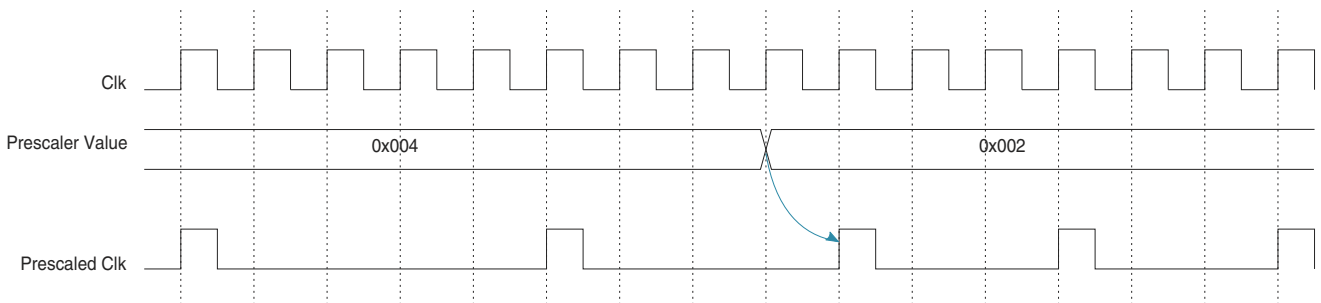


Figure 32-3. Prescaler Value Change Timing Diagram

32.3.2.2 Input Capture

There are two Input Capture Channels, and each Input Capture Channel has a dedicated capture pin, capture register and input edge detection/selection logic. Each input capture function has an associated status flag, and can cause the processor to make an interrupt service request.

When a selected edge transition occurs on an Input Capture pin, the contents of the GPT_CNT is captured on the corresponding capture register and the appropriate interrupt status flag is set. An interrupt request can be generated when the transition is detected *if* its corresponding enable bit is set (in the Interrupt Register). The capture can be programmed to occur on the input pin's rising edge, falling edge, on both rising and falling edges, or the capture can be disabled. The events are synchronized with the clock that was selected to run the counter. Only those transitions that occur at least one clock cycle (clock selected to run the counter) *after* the previous recorded transition will be guaranteed to trigger a capture event. There can be up to one clock cycle of uncertainty in the latching of the input transition. The Input Capture registers can be read *at any time* without affecting their values.

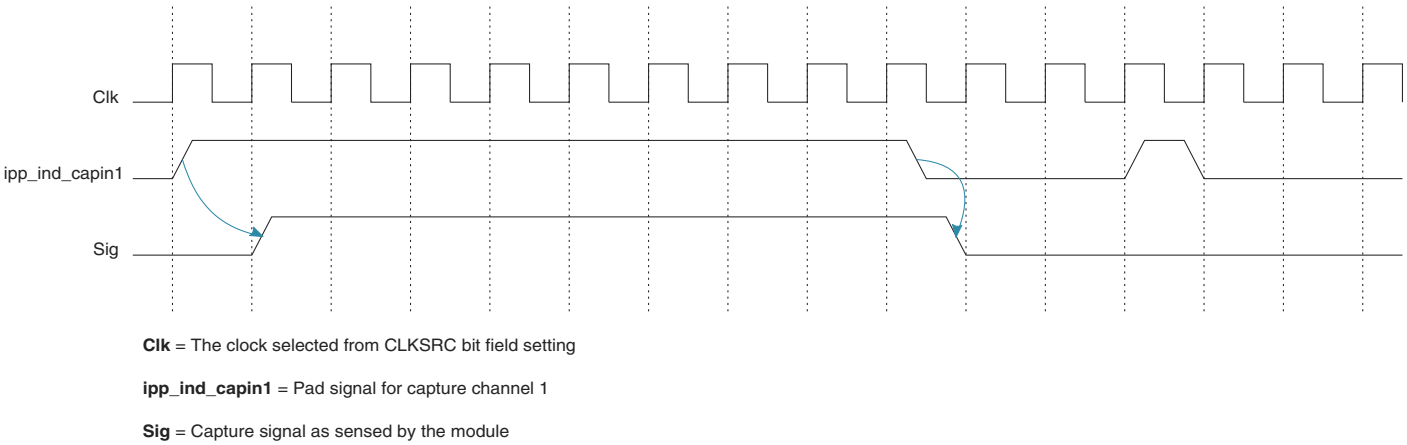


Figure 32-4. Input Capture Event Timing

32.3.2.3 Output Compare

The three Output Compare Channels *use the same counter* (GPT_CNT) as the Input Capture Channels. When the programmed content of an Output Compare register matches the value in GPT_CNT, an output compare status flag is set and an interrupt is generated (if the corresponding bit is set in the interrupt register). Consequently, the Output Compare timer pin will be set, cleared, toggled, not affected at all or provide an active-low pulse for one input clock period (subject to the restriction on the maximum frequency allowed on the pad) according to the mode bits (that were programmed).

There is also a "forced-compare" feature that allows the software to generate a compare event when required, *without the condition of the counter value that is equal to the compare value*. The action taken as a result of a forced compare is the same as when an output compare match occurs, *except that the status flags are not set and no interrupt can be generated*. Forced channels take programmed action immediately after the write to the force-compare bits. These bits are self-negating and always read as zeros.

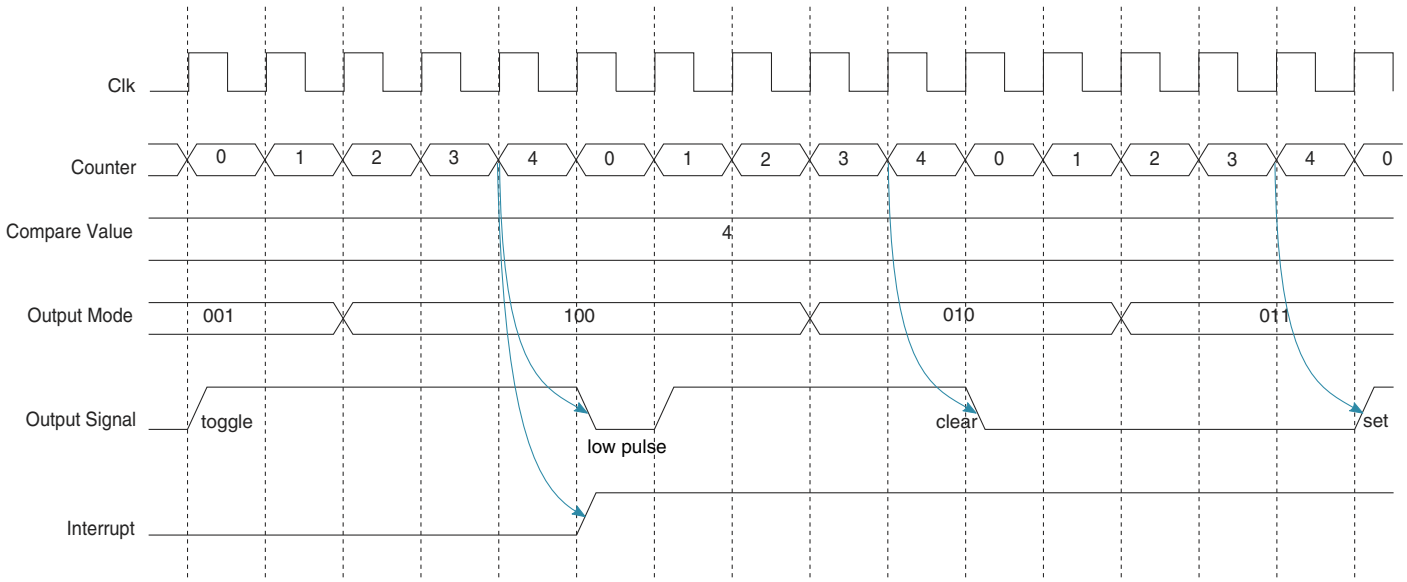


Figure 32-5. Output Compare and Interrupt Timing

32.3.2.4 Interrupts

There are 6 different interrupts that are generated by the GPT. If the selected clock for running the counter is available, then *all interrupts can be generated in Low Power and Debug modes*.

- Rollover Interrupt

  The Rollover Interrupt is generated when the GPT counter reaches 0xffffffff, then resets to 0x00000000 and continues counting. The Rollover Interrupt is enabled by the ROVIE bit in the GPT_IR register; the associated status bit is the ROV bit in the GPT_SR register.

- Input Capture Interrupt 1, 2

  After a capture event occurs, the associated Input Capture Channel generates an interrupt. The "capture event" interrupts are enabled by the IF2IE and IF1IE bits (in the GPT_IR register); the associated status bits are IF2 and IF1 (in the GPT_SR register). The capture of the counter value because of a capture event is *not affected by a pending capture interrupt*. The Capture register is updated with a new counter value when a capture event occurs, regardless of whether that Capture Channels' interrupt has been serviced or not.

- Output Compare Interrupt 1, 2, 3

After a compare event occurs, the associated Output Compare Channel generates an interrupt. The "compare event" interrupts are enabled by the OF3IE, OF2IE, and OF1IE bits (in the GPT_IR register); the associated status bits are OF3, OF2, and OF1 (in the GPT_SR register). A "forced compare" does not generate an interrupt.

A *cumulative* interrupt line is also present, which is asserted whenever any of the above interrupts are posted. The cumulative interrupt line has *no* associated enables or status bits.

32.3.2.5 Low Power Mode Behavior

In Low Power modes, if the clock from the selected clock source is available (except for the External Clock, which can be used *only if* the Peripheral Clock is available), the counter will continue to run depending on whether the control bit for that mode is set. If the clock is not present or if the corresponding low power bit in the GPT_CR control register is 0, the Main Counter and the Prescaler Counter freeze at their current values and resume counting (from their frozen values) when the Low Power mode is exited.

32.3.2.6 Debug Mode Behavior

In Debug mode, the modules in the device have the option of continuing to run or be halted.

- If the DBGEN bit is set, then the GPT timer will continue to run in Debug mode.
- If the DBGEN bit is not set (in the GPT_CR control register), then the GPT timer is halted.

32.4 Programmable Registers

The GPT has 10 user-accessible 32-bit registers, which are used to configure, operate, and monitor the state of the GPT.

An IP bus write access to the GPT Control Register (GPT_CR) and the GPT Output Compare Register1 (GPT_OCR1) results in *one cycle of wait state*, while other valid IP bus accesses incur 0 wait states.

Irrespective of the Response Select signal value, a Write access to the GPT Status Registers (Read-only registers GPT_ICR1, GPT_ICR2, GPT_CNT) will generate a bus exception.

- If the Response Select signal is driven Low, then the Read/Write access to the *unimplemented* address space of GPT (*ips_addr* is greater than or equal to $BASE + $028) will generate a bus exception.
- If the Response Select is driven High, then the Read/Write access to the unimplemented address space of GPT will *not* generate any error response (like a bus exception).

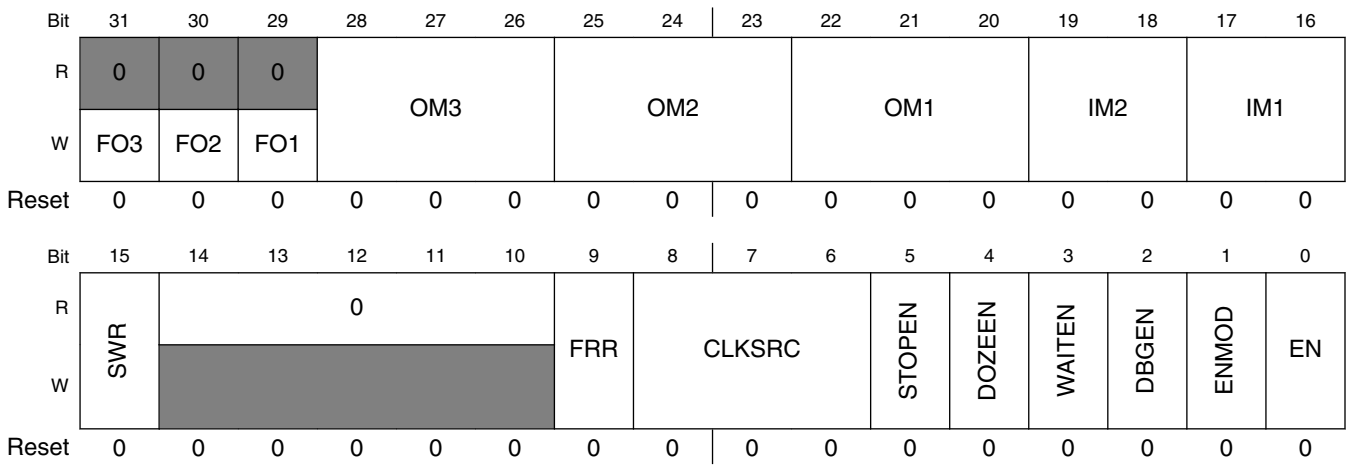
GPT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_0000	GPT Control Register (GPT_CR)	32	R/W	0000_0000h	32.4.1/ 1603
53FA_0004	GPT Prescaler Register (GPT_PR)	32	R/W	0000_0000h	32.4.2/ 1606
53FA_0008	GPT Status Register (GPT_SR)	32	R/W	0000_0000h	32.4.3/ 1607
53FA_000C	GPT Interrupt Register (GPT_IR)	32	R/W	0000_0000h	32.4.4/ 1608
53FA_0010	GPT Output Compare Register 1 (GPT_OCR1)	32	R/W	FFFF_ FFFFh	32.4.5/ 1609
53FA_0014	GPT Output Compare Register 2 (GPT_OCR2)	32	R/W	FFFF_ FFFFh	32.4.6/ 1610
53FA_0018	GPT Output Compare Register 3 (GPT_OCR3)	32	R/W	FFFF_ FFFFh	32.4.7/ 1610
53FA_001C	GPT Input Capture Register 1 (GPT_ICR1)	32	R	0000_0000h	32.4.8/ 1611
53FA_0020	GPT Input Capture Register 2 (GPT_ICR2)	32	R	0000_0000h	32.4.9/ 1611
53FA_0024	GPT Counter Register (GPT_CNT)	32	R	0000_0000h	32.4.10/ 1612

32.4.1 GPT Control Register (GPT_CR)

The GPT Control Register (GPT_CR) is used to program and configure GPT operations. An IP Bus Write to the GPT Control Register occurs after one cycle of wait state, while an IP Bus Read occurs after 0 wait states.

Address: GPT_CR is 53FA_0000h base + 0h offset = 53FA_0000h



GPT_CR field descriptions

Field	Description
31 FO3	FO3 Force Output Compare Channel 3 FO2 Force Output Compare Channel 2 FO1 Force Output Compare Channel 1 The FO*n* bit causes the pin action *programmed* for the timer Output Compare *n* pin (according to the OM*n* bits in this register). • The OF*n* flag (OF3, OF2, OF1) in the status register is **not affected**. • This bit is self-negating and always read as zero. 0 Writing a 0 has no effect. 1 Causes the programmed pin action on the timer Output Compare *n* pin; the OF*n* flag is not set.
30 FO2	See F03
29 FO1	See F03
28–26 OM3	OM3 (bits 28-26) controls the Output Compare Channel 3 operating mode. OM2 (bits 25-23) controls the Output Compare Channel 2 operating mode. OM1 (bits 22-20) controls the Output Compare Channel 1 operating mode. The OM*n* bits specify the response that a compare event will generate on the output pin of Output Compare Channel *n*.

Table continues on the next page...

GPT_CR field descriptions (continued)

Field	Description
	• The toggle, clear, and set options cause a change on the output pin *only* if a compare event occurs. • When OM*n* is programmed as 1xx (active low pulse), the output pin is set to one immediately on the next input clock; a low pulse (that is an input clock in width) occurs when there is a compare event. Note that here, "input clock" refers to the clock selected by the CLKSRC bits of the GPT Control Register. 000 Output disconnected. No response on pin. 001 Toggle output pin 010 Clear output pin 011 Set output pin 1xx Generate an active low pulse (that is one input clock wide) on the output pin.
25–23 OM2	See OM3
22–20 OM1	See OM3
19–18 IM2	IM2 (bits 19-18, Input Capture Channel 2 operating mode) IM1 (bits 17-16, Input Capture Channel 1 operating mode) The IM*n* bit field determines the transition on the input pin (for Input capture channel *n*), which will trigger a capture event. 00 capture disabled 01 capture on rising edge only 10 capture on falling edge only 11 capture on both edges
17–16 IM1	See IM2
15 SWR	Software reset. This is the software reset of the GPT module. It is a self-clearing bit. • The SWR bit is set when the module is in reset state. • The SWR bit is cleared when the reset procedure finishes. • Setting the SWR bit resets **all of the registers** to their default reset values, except for the CLKSRC, EN, ENMOD, STOPEN, WAITEN, and DBGEN bits in the GPT Control Register (this control register). 0 GPT is not in reset state 1 GPT is in reset state
14–10 Reserved	This read-only field is reserved and always has the value zero. Reserved bits. • Writing a value to these reserved bits will not affect GPT operations. • These reserved bits are always read as zero. • It is recommended that all writes to these reserved bits be 0 (for forward compatibility).
9 FRR	Free-Run or Restart mode. The FFR bit determines the behavior of the GPT when a compare event in channel 1 occurs. • In Restart mode, after a compare event, the counter resets to 0x00000000 and resumes counting (after the occurrence of a compare event). • In Free-Run mode, after a compare event, the counter continues counting until 0xFFFFFFFF and then rolls over to 0.

Table continues on the next page...

GPT_CR field descriptions (continued)

Field	Description
	0 Restart mode 1 Free-Run mode
8–6 CLKSRC	Clock Source select. The CLKSRC bits select which clock will go to the prescaler (and subsequently be used to run the GPT counter). • The CLKSRC bit field value should only be changed after disabling the GPT by clearing the EN bit in this register (GPT_CR). • A software reset does not affect the CLKSRC bit. 000 No clock 001 Peripheral Clock 010 High Frequency Reference Clock 011 External Clock (CLKIN) 100 Low Frequency Reference Clock 101 Crystal oscillator as Reference Clock others Reserved
5 STOPEN	GPT Stop Mode enable. The STOPEN read/write control bit enables GPT operation *during Stop mode*. • A hardware reset resets the STOPEN bit. • A software reset *does not affect* the STOPEN bit. 0 GPT is disabled in Stop mode. 1 GPT is enabled in Stop mode.
4 DOZEEN	GPT Doze Mode Enable. • A hardware reset resets the DOZEEN bit. • A software reset *does not affect* the DOZEEN bit. 0 GPT is disabled in doze mode. 1 GPT is enabled in doze mode.
3 WAITEN	GPT Wait Mode enable. The WAITEN read/write control bit enables GPT operation *during Wait mode*. • A hardware reset resets the WAITEN bit. • A software reset *does not affect* the WAITEN bit. 0 GPT is disabled in wait mode. 1 GPT is enabled in wait mode.
2 DBGEN	GPT debug mode enable. The DBGEN read/write control bit enables GPT operation *during Debug mode*. • A hardware reset resets the DBGEN bit. • A software reset *does not affect* the DBGEN bit. 0 GPT is disabled in debug mode. 1 GPT is enabled in debug mode.
1 ENMOD	GPT Enable mode.

Table continues on the next page...

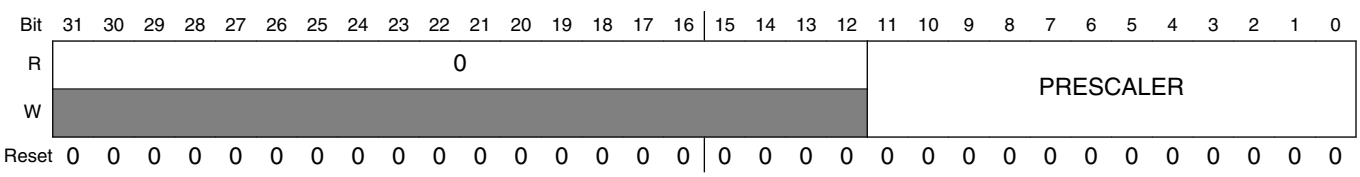
GPT_CR field descriptions (continued)

Field	Description
	When the GPT is disabled (EN=0), then both the Main Counter and Prescaler Counter *freeze their current count values*. The ENMOD bit determines the value of the GPT counter when Counter is enabled again (if the EN bit is set). • If the ENMOD bit is 1, then the Main Counter and Prescaler Counter values are reset to 0 after GPT is enabled (EN=1). • If the ENMOD bit is 0, then the Main Counter and Prescaler Counter restart counting *from their frozen values* after GPT is enabled (EN=1). • If GPT is programmed to be disabled in a low power mode (STOP/WAIT), then the Main Counter and Prescaler Counter *freeze at their current count values* when the GPT enters low power mode. • When GPT exits low power mode, the Main Counter and Prescaler Counter start counting from their frozen values, regardless of the ENMOD bit value. • Setting the SWR bit will clear the Main Counter and Prescaler Counter values, regardless of the value of EN or ENMOD bits. • A hardware reset resets the ENMOD bit. • A software reset *does not affect* the ENMOD bit. 0 GPT counter will retain its value when it is disabled. 1 GPT counter value is reset to 0 when it is disabled.
0 EN	GPT Enable. The EN bit is the GPT module enable bit. **Before setting the EN bit**, we recommend that *all registers be properly programmed*. • A hardware reset resets the EN bit. • A software reset *does not affect* the EN bit. 0 GPT is disabled. 1 GPT is enabled.

32.4.2 GPT Prescaler Register (GPT_PR)

The GPT Prescaler Register (GPT_PR) contains bits that determine the divide value of the clock that runs the counter.

Address: GPT_PR is 53FA_0000h base + 4h offset = 53FA_0004h



GPT_PR field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value zero. Reserved bits.

Table continues on the next page...

GPT_PR field descriptions (continued)

Field	Description
	• Writing a value to these reserved bits will not affect GPT operations. • These reserved bits are always read as zero.
11–0 PRESCALER	Prescaler bits. The clock selected by the CLKSRC field is divided by [PRESCALER + 1], and then used to run the counter. • A change in the value of the PRESCALER bits cause the Prescaler counter to reset and a new count period to start immediately. • See Figure 32-3 for the timing diagram. 0x000 Divide by 1 0x001 Divide by 2 ... ... 0xFFF Divide by 4096

32.4.3 GPT Status Register (GPT_SR)

The GPT Status Register (GPT_SR) contains bits that indicate that a counter has rolled over, and if any event has occurred on the Input Capture and Output Compare channels. The bits are cleared by writing a 1 to them.

Address: GPT_SR is 53FA_0000h base + 8h offset = 53FA_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										ROV	IF2	IF1	OF3	OF2	OF1
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPT_SR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value zero. Reserved bits. • Writing a value to these reserved bits will not affect GPT operations. • These reserved bits are always read as zero.
5 ROV	Rollover Flag. The ROV bit indicates that the counter has reached its *maximum possible value* and *rolled over* to 0 (from which the counter continues counting). The ROV bit is only set if the counter has reached 0xFFFFFFFF in both Restart and Free-Run modes.

Table continues on the next page...

GPT_SR field descriptions (continued)

Field	Description
	0 Rollover has not occurred. 1 Rollover has occurred.
4 IF2	IF2 Input capture 2 Flag IF1 Input capture 1 Flag The IF*n* bit indicates that a capture event has occurred on Input Capture channel *n*. 0 Capture event has not occurred. 1 Capture event has occurred.
3 IF1	See IF2
2 OF3	OF3 Output Compare 3 Flag OF2 Output Compare 2 Flag OF1 Output Compare 1 Flag The OF*n* bit indicates that a compare event has occurred on Output Compare channel *n*. 0 Compare event has not occurred. 1 Compare event has occurred.
1 OF2	See OF3
0 OF1	See OF3

32.4.4 GPT Interrupt Register (GPT_IR)

The GPT Interrupt Register (GPT_IR) contains bits that control whether interrupts are generated after rollover, input capture and output compare events.

Address: GPT_IR is 53FA_0000h base + Ch offset = 53FA_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																										ROVIE	IF2IE	IF1IE	OF3IE	OF2IE	OF1IE
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPT_IR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value zero. Reserved bits. • Writing a value to these reserved bits will not affect GPT operations. • These reserved bits are always read as zero.
5 ROVIE	Rollover Interrupt Enable.

Table continues on the next page...

GPT_IR field descriptions (continued)

Field	Description
	The ROVIE bit controls the Rollover interrupt. 0 Rollover interrupt is disabled. 1 Rollover interrupt enabled.
4 IF2IE	IF2IE Input capture 2 Interrupt Enable IF1IE Input capture 1 Interrupt Enable The IF*n*IE bit controls the IF*n*IE Input Capture *n* Interrupt Enable. 0 IF2IE Input Capture *n* Interrupt Enable is disabled. 1 IF2IE Input Capture *n* Interrupt Enable is enabled.
3 IF1IE	See IF2IE
2 OF3IE	OF3IE Output Compare 3 Interrupt Enable OF2IE Output Compare 2 Interrupt Enable OF1IE Output Compare 1 Interrupt Enable The OF*n*IE bit controls the Output Compare Channel *n* interrupt. 0 Output Compare Channel *n* interrupt is disabled. 1 Output Compare Channel *n* interrupt is enabled.
1 OF2IE	See OF3IE
0 OF1IE	See OF3IE

32.4.5 GPT Output Compare Register 1 (GPT_OCR1)

The GPT Compare Register 1 (GPT_OCR1) holds the value that determines when a compare event will be generated on Output Compare Channel 1. Any write access to the Compare register of Channel 1 while in Restart mode (FRR=0) will reset the GPT counter.

An IP Bus Write access to the GPT Output Compare Register1 (GPT_OCR1) occurs *after* one cycle of wait state; an IP Bus Read access occurs *immediately* (0 wait states).

Address: GPT_OCR1 is 53FA_0000h base + 10h offset = 53FA_0010h

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

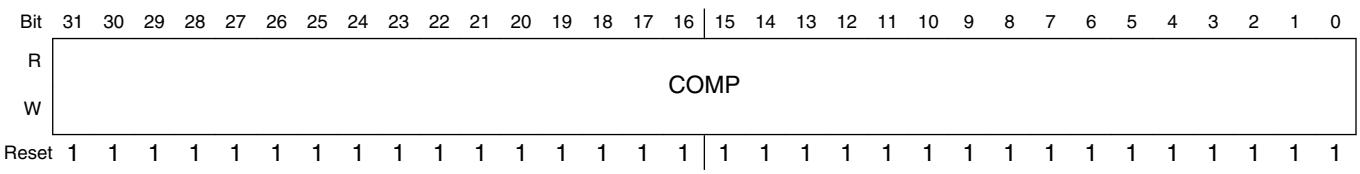
GPT_OCR1 field descriptions

Field	Description
31–0 COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 1.

32.4.6 GPT Output Compare Register 2 (GPT_OCR2)

The GPT Compare Register 2 (GPT_OCR2) holds the value that determines when a compare event will be generated on Output Compare Channel 2.

Address: GPT_OCR2 is 53FA_0000h base + 14h offset = 53FA_0014h



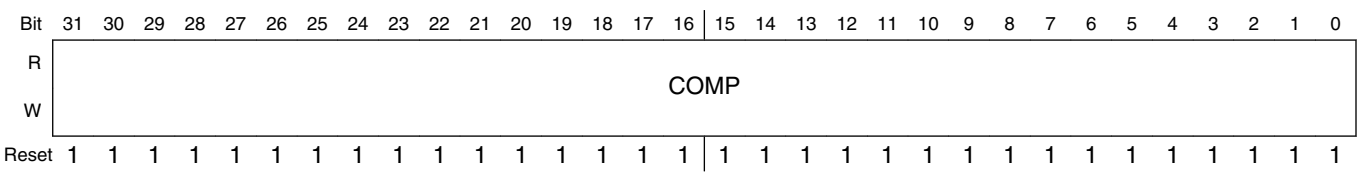
GPT_OCR2 field descriptions

Field	Description
31–0 COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 2.

32.4.7 GPT Output Compare Register 3 (GPT_OCR3)

The GPT Compare Register 3 (GPT_OCR3) holds the value that determines when a compare event will be generated on Output Compare Channel 3.

Address: GPT_OCR3 is 53FA_0000h base + 18h offset = 53FA_0018h



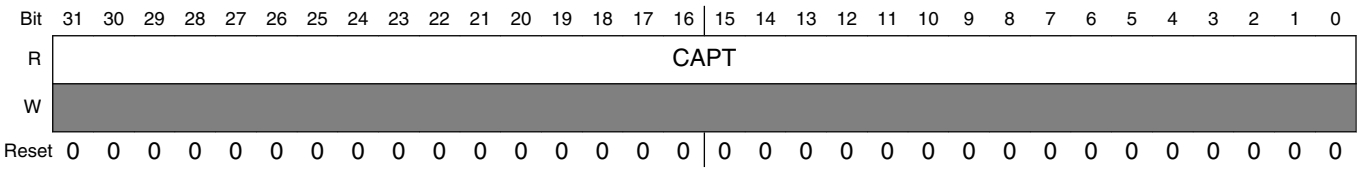
GPT_OCR3 field descriptions

Field	Description
31–0 COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 3.

32.4.8 GPT Input Capture Register 1 (GPT_ICR1)

The GPT Input Capture Register 1 (GPT_ICR1) is a read-only register that holds the value that was in the counter during the last capture event on Input Capture Channel 1.

Address: GPT_ICR1 is 53FA_0000h base + 1Ch offset = 53FA_001Ch



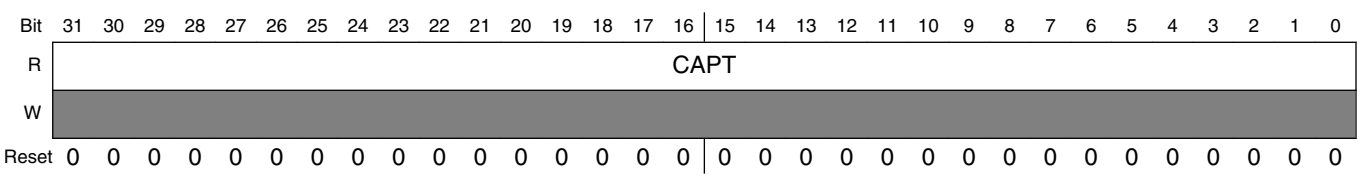
GPT_ICR1 field descriptions

Field	Description
31–0 CAPT	Capture Value. After a capture event on Input Capture Channel 1 occurs, the current value of the counter is loaded into GPT Input Capture Register 1.

32.4.9 GPT Input Capture Register 2 (GPT_ICR2)

The GPT Input capture Register 2 (GPT_ICR2) is a read-only register which holds the value that was in the counter during the last capture event on input capture channel 2.

Address: GPT_ICR2 is 53FA_0000h base + 20h offset = 53FA_0020h



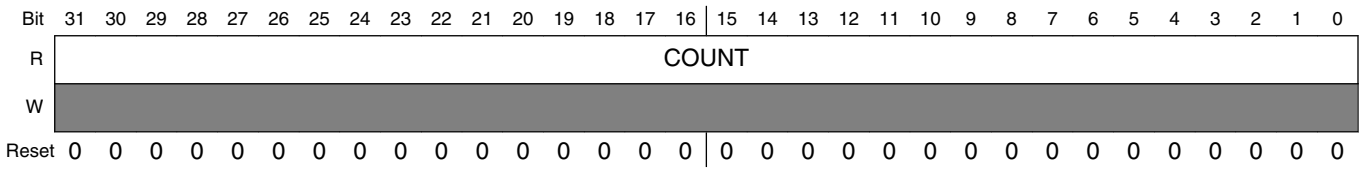
GPT_ICR2 field descriptions

Field	Description
31–0 CAPT	Capture Value. After a capture event on Input Capture Channel 2 occurs, the current value of the counter is loaded into GPT Input Capture Register 2.

32.4.10 GPT Counter Register (GPT_CNT)

The GPT Counter Register (GPT_CNT) is the main counter's register. GPT_CNT is a read-only register and can be read without affecting the counting process of the GPT.

Address: GPT_CNT is 53FA_0000h base + 24h offset = 53FA_0024h



GPT_CNT field descriptions

Field	Description
31–0 COUNT	Counter Value. The COUNT bits show the current count value of the GPT counter.

Chapter 33
2D Graphics Processing Unit (GPU2D)

33.1 Overview

This block guide describes key architectural features of the G12 OpenVG core as well as details of the G12 OpenVG IP customizing and integration in the i.MX50.

The ATI^{TM} Z160(G12) IP is an embedded, 2D and vector graphics accelerator targeting the OpenVG 1.0.1 graphics API and feature set.

It accelerates 2D bitmap graphics operations, such as BitBlt, fill and raster operations using a separate 2D graphics acceleration unit.

Vector graphics rendering is accelerated by a separate anti-aliasing polygon rasterizer, which is connected to the 2D graphics acceleration unit.

The core has a rich, but well-chosen set of features, with emphasis being on very high image quality and low memory bandwidth consumption.

The GPU top level block diagram is presented at GPU2D Block Diagram.

33.2 GPU2D Feature List

The following chapters describe the functional features of the graphics processor.

33.2.1 Frame Buffer

- Frame buffer sizes supported up-to 2048x2048
- ARGB4444, RGB565, ARGB1555, ARGB5551, ARGB8888 frame buffer modes
- Configurable ARGB order in frame buffer: ARGB, BGRA, ABGR, RGBA
- Linear and block-based (4x4 pixels) frame buffer modes
- Fast buffer clears
- Support for OpenVG render to Image

33.2.2 2D Bitmap Graphics (Separate 2D Unit)

- Parallel operation with the 3D pipeline, independent command input
- BitBlt (surface-to-surface copy)
  - Format conversion from monochrome/ARGB/YUV to ARGB during BitBlt
- Block fill
- Internal 32-bit color precision
- Source bitmap format:
  - 1/4/8-bit monochrome
  - ARGB4444, RGB565, ARGB1555, ARGB5551, ARGB8888
  - Configurable ARGB order: ARGB, BGRA, ABGR, RGBA
  - Packed YUV 4:2:2 formats (FOURCC codes YUY2, UYVY, YVYU), two pixels per 32 bits of data.
  - 1-bit bitmap maps to foreground and background colors.
  - 4-bit bitmap is optionally gamma corrected to 8-bit alpha values and can be combined with foreground color to draw anti-aliased fonts.
- Destination bitmap format:
  - ARGB4444, RGB565, ARGB1555, ARGB5551, ARGB8888, B8, A8, AB88
  - Configurable ARGB order: ARGB, BGRA, ABGR, RGBA
- Supports three source bitmaps for separate mask/pattern/alpha bitmap support plus reading destination for ROP, blend and color key operations
- Supports masking source coordinates for wrapping patterns
- Supports ROP4 (ROP3 with separate ROPs for masked and unmasked pixels) logical operations
- Supports inverting mask and alpha values from source
- Supports destination rotation by 0/90/180/270 degrees Supports programmable blending with optional alpha un-premultiply
- Supports per pixel and constant alpha with optional modulation by source color alpha for OpenVG alpha masking Supports color keying by source and destination colors, with optional ignoring of alpha channel
- Supports one scissor rectangle for destination coordinates
- Dithering (ordered)
- Color component masking
- RGB reads and writes
- Non-power of two source and destination bitmap sizes supported (stride must be a multiple of 32-bits)
- BitBlt with scaling implemented with the 3D rendering pipeline, bilinear filtering with texture lookups, programmable filter kernels possible with the programmable Pixel processor

33.2.3 Vector Graphics

- Parallel operation with the 3D pipeline, independent command input
- Rasterization of convex and concave polygons with anti-aliasing
- Efficient native polygon rendering (no tessellation to triangles)
- Non-zero and odd-even fill rules
- Primitives supported:
  - Polygons
  - OpenVG path primitives (except Elliptical Arcs): Horizontal/vertical lines, generic lines, curves, smooth curves, moveto, path closing
  - Curve types supported: cubic and quadratic BÈzier
  - Strokes with thickness, joints and end caps, unlimited stroke thickness
  - Special case handling of singularities for thick strokes
  - Supports paths with a maximum of 256 crossings along a horizontal or a vertical line
- Input coordinates:
  - Absolute and relative coordinate input in floating point
  - Fixed-point (byte, short, int) and floating-point coordinate input - 0.8, 0.16, 16.16 formats
  - Little- and Big-endian support separately selectable for command stream and data.
- Geometry
  - User to surface transform for vertices and stroke shape
  - Hardware curve tessellation
  - Adjustable accuracy for curve and round cap splitting
  - OpenVG/SVG join types: Miter (with miter limit), round, bevel
  - OpenVG/SVG cap types: Butt, round, square
- Pixel processing:
  - Programmable gradient and texturing processor
  - Linear and radial gradients (with focal point)
  - Perspective texture mapping with filtering
  - Two textures supported
  - sRGB and pre-multiply support for textures
  - 16-sample anti-aliasing
  - 4x RGSS AA (Rotated Grid Super Sample)
  - Per-pixel alpha-masking
  - Maximum texture size: 1024x1024 pixels
- Vector graphics rendering system ARM platform load:
  - Display list generation during path creation ñ commands and vertices are stored to an internal format/buffer, no format conversion is performed

- Filling or stroking a path only requires a few register writes to start the operation in hardware
- Display lists are transferred to the vector graphics rasterizer using DMA without ARM platform interaction

33.3 GPU2D Block Diagram

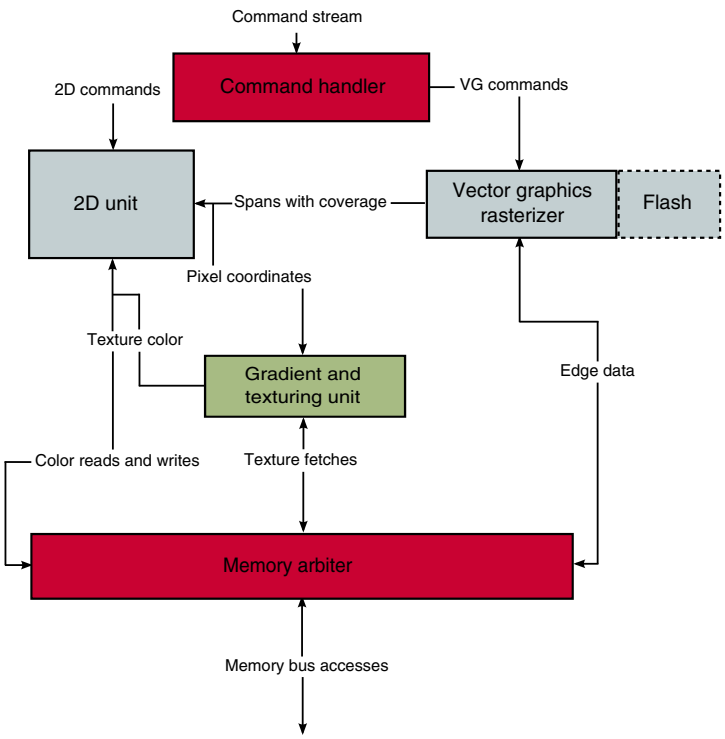


Figure 33-1. GPU2D Block Diagram

33.4 GPU SoC Interface

33.4.1 GPU2D Top Level Diagram

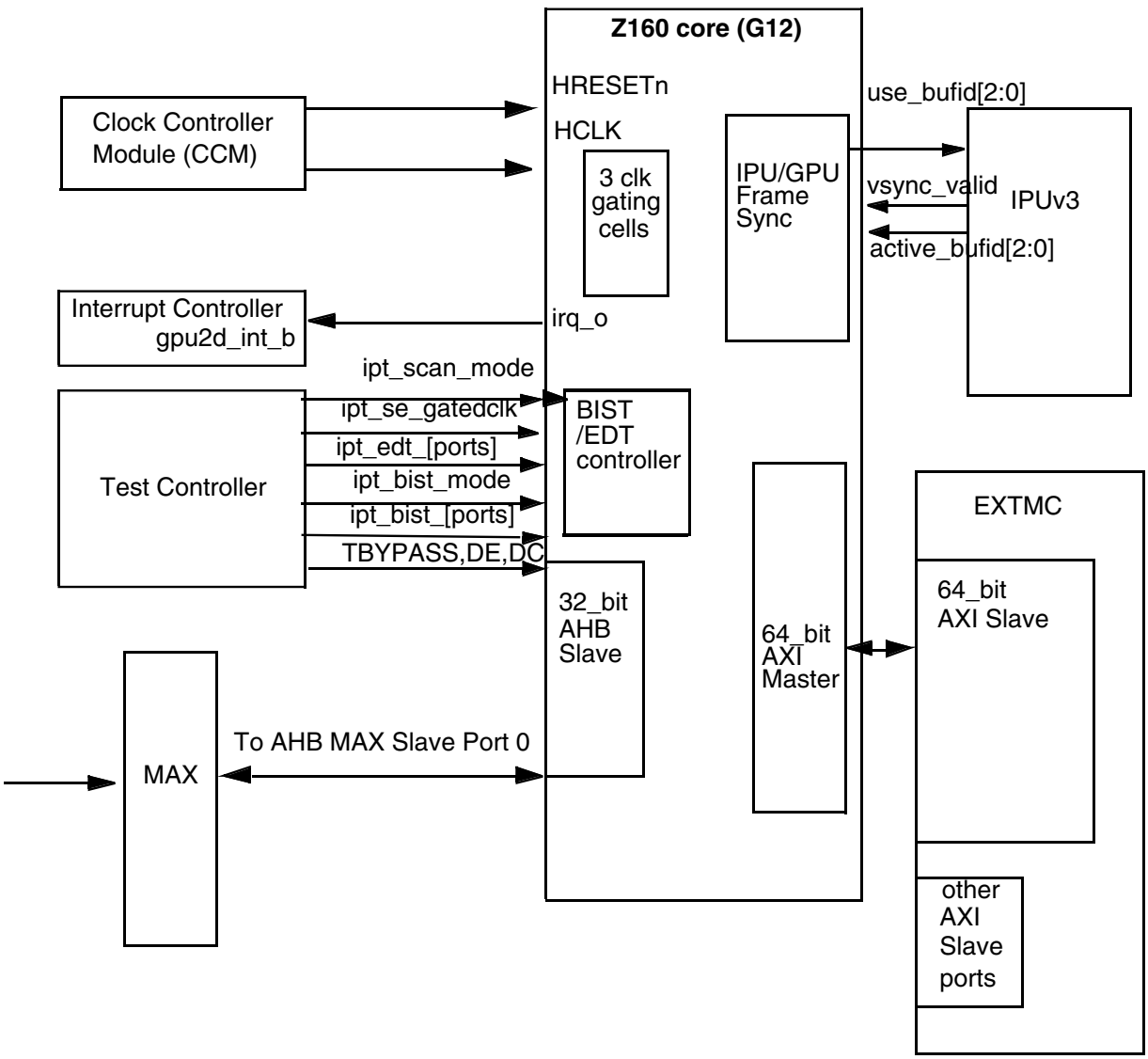


Figure 33-2. GPU2D System Connectivity

33.4.2 SoC Bus Connection

The GPU2D has the following two bus interfaces in SoC:

- 32_bit AHB slave bus.

This port is connected to ARM through MAX slave port 0, it allows access to the control registers and is the command stream input to Z160 core. The AHB interface is a fully compliant ARM AMBA AHB bus interface. It has been designed to comply with Rev 2.0 of the AMBA specification and it implements the Retry capable Slave part of the specification

- 64_bit AXI master bus

The external memory master bus is a standard 64_bit AMBA AXI master bus.

33.5 Other Signals Connection

Table 33-1. Z160(G12) miscellaneous signals

Name	Type	Source/ destination	Description
gpu2d_int_b	Output	Interrupt controller	Interrupt output from the Z160 (G12) OpenVG Graphics Core. Active LOW.
gpu2d_use_bufid[2:0]	Output	Image Processing Unit (IPU) frame sync	Tell IPU which buffer can be displayed
ipu_vsync_valid	Input	IPU frame sync	Tell GPU2D one frame is finished
ipu_active_bufid[2:0]	Input	IPU frame sync	Tell GPU2D which frame is free to be overwritten

33.6 Clocking Architecture

There is one clock input to the GPU2D, namely HCLK. In fact Z160(G12) core needs 5 clock inputs, named bus_clk, clk_0,clk_1,clk_2,clk_3, which are divided from HCLK but balanced and therefore, count as only one clock domain.

33.7 Power Management

Table 33-2. GPU2D miscellaneous signals

Clock name	Enable signal	Clocked sub-blocks	Related area
bus_clk(hclk)		Busif	4%
clk_0(hclk)	Input	Arbiter, Input	1%
clk_1	clock1_ena	Bcache	20%

Table continues on the next page...

Table 33-2. GPU2D miscellaneous signals (continued)

Clock name	Enable signal	Clocked sub-blocks	Related area
clk_2	clock2_ena	2D, V3	45%
clk_3	clock3_ena	V1, V2	30%

The different use cases and the associated clock settings are presented at the following table. For the use-cases where the core is not used at all, also bus_clk and clk_0 should be gated off to maximize the power savings. This cannot be controlled by the core itself.

Use case	bus clk	clk 0	clk 1	clk 2	clk 3
Idle	on	on	off	off	off
Bitmap case - standalone	on	on	on	on	off
Vector graphics	on	on	on	on	on

Figure 33-3. Graphics Core Operating Mode

If the whole chip go into low power mode, all clk source will be shut down by CCM.

33.8 Modes of Operation

The GPU2D supports

- 2D bitmap acceleration mode
- Vector Graphic acceleration mode
- low power mode

33.9 Reset

The GPU2D has only 1 reset port named HRESETn. The internal Z160 core (G12) contains asynchronous reset signals for each corresponding clock domain - namely rst_n for the clk clocks and bus_rst_n for the busclk clock, which are directly connected to HRESETn.

Resets are active low. The reset signals asynchronously reset all of the DFFs in the design.

33.10 Interrupts

The Z160 Core (G12) generates individual interrupts internally. Each interrupt can be enabled or disabled by changing its own enable bit. Setting the enable bit HIGH enables the corresponding interrupt.

The interrupts from all sources in the Z160 Core (G12) are combined (ORed) and output as the active LOW **gpu2d_int_b** interrupt.

33.11 DMA

GPU2D is a DMA master in fact, it read command and write data to system memory through EXTMC.

GPU2D also can be feed by DMA through MAX slave port 0.

33.12 Memory Map

The Z160 Graphics Core (G12) memory map is described in the following sections:

- AHB slave interface
- AXI master memory interface (EXTMC port).

33.12.1 AHB Slave Interface

This is based on the GPU2D occupying slave port 0 of the crossbar., whose base address is 0x20000000. GPU2D has 2 kinds of registers, one is "Interface" registers which can be read/write accessible, the others are "Internal" registers which only can be written through Z160's slave ports.

GPU2D Memory Map

Description	Memory Address Base	Memory Address End
Registers	0x2000 0000	0x200007FC

Description of all user-accessible registers in the design can be found from regs.html

Interface registers can only be accessed by status read / write channel directly through the slave port, which is mapped to 0x400 - 0x7fc address range. These registers are used to read and clear interrupts by the ARM platform, for example.

Internal 2D / VG registers are write only. They can only be accessed by two writing commands through the slave port (GPU2D's base address). The first command is a register address write, followed by a data write command.

In practice, this means that the command stream, which is prepared by the software driver is written through the 0x000-0x3ff address range either directly, or through the DMA operation by the core. These registers are not accessible directly by slave ports channel, like interface registers are not accessible through this channel.

33.12.2 AXI Master Memory Interface (EXTMC Port)

The Z160 Graphics Core (G12 or GPU2D) can access memory with or without a *Memory Management Unit* (MMU):

Translation is performed using a table with 8KB entries, one for each 4KB page in a 32MB linear address space.

Chapter 34
I2C Controller (I2C)

34.1 Overview

This chapter describes block-level operation and programming of I2C. The chapter is intended for a block driver software developer. To understand how the block is integrated at the SoC level, a system software developer can refer to discussions of the block in the appropriate SoC-level chapter(s).

References: This document assumes an understanding of the following reference:

1. The I2C Bus Specification, Version 2.1

The Inter IC (I2C) provides functionality of a standard I2C slave and master. The I2C is designed to be compatible with the standard Philips I2C bus protocol.

I2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I2C standard allows additional devices to be connected to the bus for expansion and system development. See the connection diagram in the figure below.

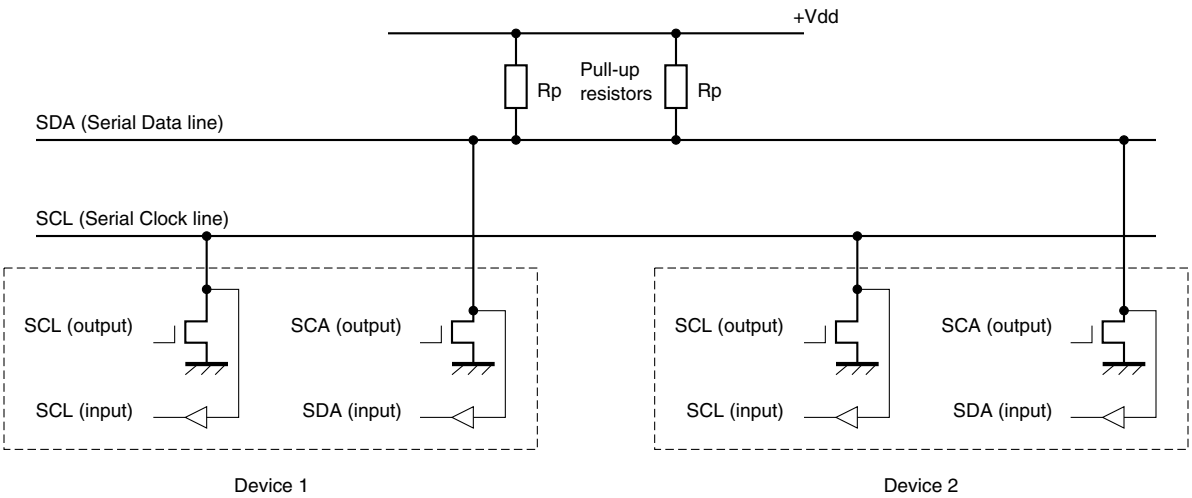


Figure 34-1. Connection of Devices to I2C Bus

The I2C interface operates up to 400 kbps, but it depends on the pin loading and timing characteristics. For pin requirement details, refer to Philips I2C Bus Specification, Version 2.1. The I2C system is a true multiple-master bus including arbitration and collision detection that prevents data corruption if multiple devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer. The figure below shows the block diagram of I2C.

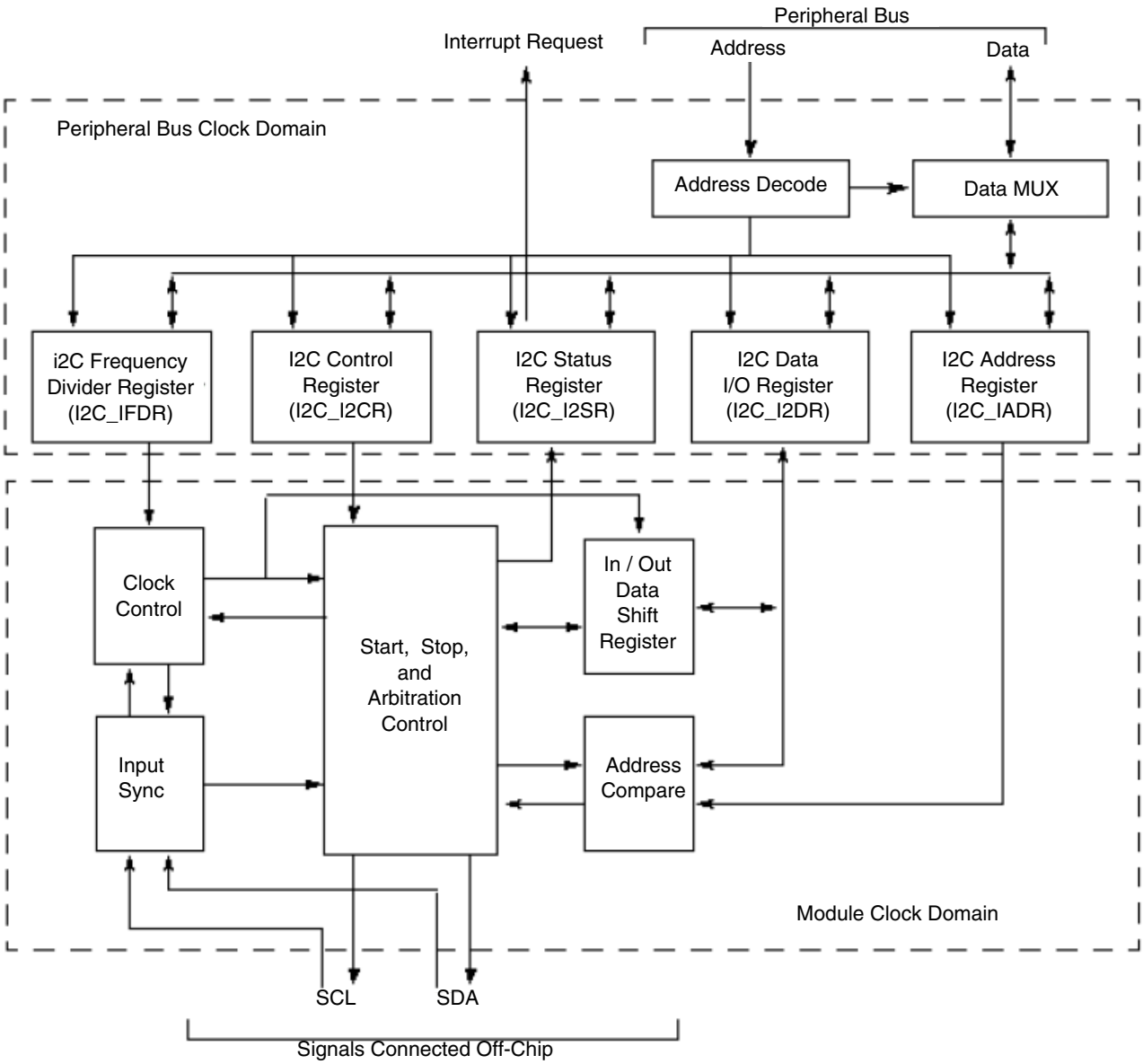


Figure 34-2. I2C Block Diagram

34.1.1 Features

The I2C has the following key features:

- Compatibility with I2C bus standard
- Multiple-master operation
- Software-programmable for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave

- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

34.1.2 Modes and Operations

The I2C primarily operates in two different functional modes.

34.1.2.1 Standard Mode

In Standard mode, I2C supports the data transfer rates up to 100 Kbits/s.

34.1.2.2 Fast Mode

In Fast Mode, data transfer rates up to 400 Kbits/s can be achieved.

As per block operation, there is no special configuration required for Fast and Standard mode. It is the data transfer rate which distinguishes Standard and Fast mode.

34.2 External Signals

The table below describes all I2C signals that connect off-chip.

For I2C compliance, all devices connected to the SCL and SDA signals must have open-drain or open-collector outputs. The logic AND function is exercised on both lines with external pull-up resistors.

Input of SCL and SDA also need to be manually opened by set SION bit in IOMUX after corresponding PADs were selected as I2C function.

Table 34-1. Off-Chip Block Signals

Signal	I/O	Description	Reset State[1]	Pull-Up/Down[1]
SCL	I/O	Serial Clock	1	Active
SDA	I/O	Serial Data	1	Active

1. The reset state values and pull-up/down requirements provided in this table are from the block-level perspective. To understand how the block is integrated at the SoC level, the system software developer must see discussions of the block in the appropriate SoC-level chapter(s). For example, a block signal that requires a pull-up could be integrated with a pull-

up option built into the SoC. In this case, the system software developer must ensure the proper programming at the SoC level.

34.3 Functional Description

34.3.1 I2C System Configuration

Out of a reset, the I2C defaults to slave receive operations. Thus, when not operating as a master or responding to a slave transmit address, the I2C will default to the slave receiver state.

For exceptions, see Initialization Sequence.

NOTE

The I2C is designed to be compatible with the Philips™ I2C bus protocol. For information on system configuration, protocol, and restrictions, refer to the I2C Bus Specification, Version 2.1. The I2C supports Standard and Fast modes only.

34.3.2 I2C Protocol

The I2C communication protocol consists of six components, as follows:

- START
- Data Source/Recipient
- Data Direction
- Slave Acknowledge
- Data Acknowledge
- STOP

See the figure below for the I2C standard communication protocol, as defined in the following sections.

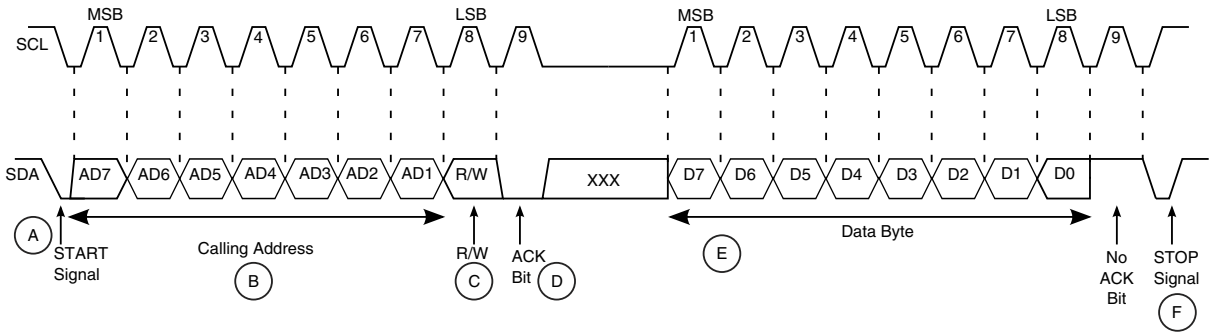


Figure 34-3. I2C Standard Communication Protocol

34.3.2.1 START Signal

When no other device is a bus master (both SCL and SDA lines are at logic high), a device can initiate communication by sending a START signal (see A in Figure 34-3).

A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a data transfer (each data transfer can be several bytes long) and awakens all slaves.

34.3.2.2 Slave Address Transmission

The master sends the slave address in the first byte after the START signal (B). After the seven-bit calling address, it sends the R/W bit (C), which tells the slave data transfer direction.

Each slave must have a unique address. An I2C master must not transmit an address that is the same as its slave address; it cannot be master and slave at the same time.

The slave whose address matches that sent by the master pulls SDA low at the ninth clock (D) to return an acknowledge bit.

34.3.2.3 Data Transfer

When successful slave addressing is achieved, the data transfer can proceed (E) on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master.

Data can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 34-3. SCL is pulsed once for each data bit, most-significant bit first. The receiving device must acknowledge each byte by pulling SDA low at the ninth clock; therefore, a data byte transfer takes nine clock pulses.

If it does not acknowledge the master, the slave receiver must leave SDA high. The master can then generate a STOP signal to abort the data transfer or generate a START signal (a repeated start, as shown in Figure 34-4) to start a new calling sequence.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means end-of-data to the slave. The slave releases SDA for the master to generate a STOP or START signal.

NOTE

Writing to the data register triggers transmit operation.

Transmit data should always be written after MTX bit is programmed. Transmit data is not latched inside until the transfer is initiated on the interface bus.

After the transmit data write in I2C, software can either wait for a transfer-done interrupt or it can poll for ICF bit for zero, if new data has to be written during the previous data transfer. The IIF bit may not be polled if the IIEN bit is set because the I2C will generate an interrupt when IIF is set.

34.3.2.4 STOP Signal

The master can terminate communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical high (F).

NOTE

A master can generate a STOP even if the slave has made an acknowledgment, at which point the slave must release the bus.

34.3.2.5 Repeat Start

Instead of signalling a STOP, the master can repeat the START signal, followed by a calling command

(see A in the figure below). A repeated START occurs when a START signal is generated without first generating a STOP signal to end the communication. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

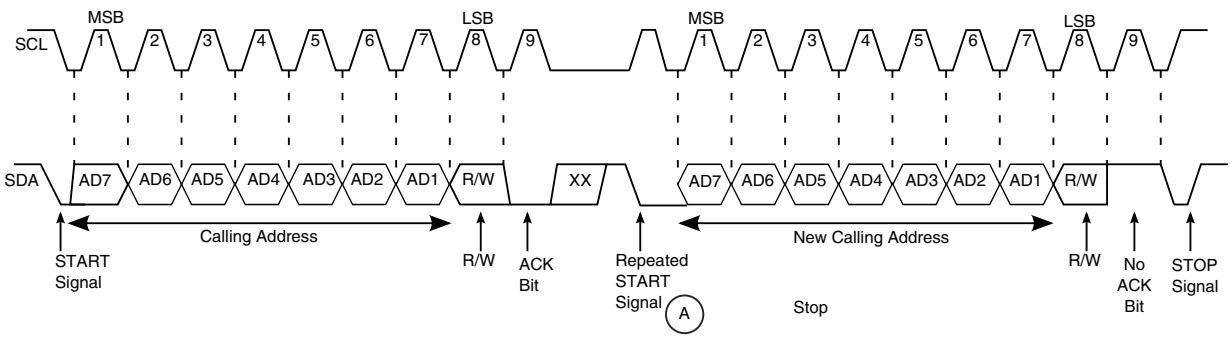


Figure 34-4. Repeated START

34.3.3 Arbitration Procedure

If multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the low period equals the longest clock-low period among the devices, and the high period equals the shortest. A data arbitration procedure determines the relative priority of competing devices.

A device loses arbitration if it sends logic high while another sends logic low; it immediately switches to slave-receive mode and stops driving SDA. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets the arbitration lost bit in the I2C Status register (I2C_I2SR[IAL] to indicate loss of arbitration).

34.3.4 Clock Synchronization

Because wire-AND logic is used, a high-to-low transition on SCL affects devices connected to the bus. Devices start counting their low period when the master drives SCL low. When a device clock goes low, it holds SCL low until the clock high state is reached. However, the low-to-high change in this device clock may not change the state of SCL if another device clock is still in its low period. Therefore, the device with the longest low period holds the synchronized clock SCL low.

Devices with shorter low periods enter a high wait state during this time (see the figure below). When all devices involved have counted off their low period, the synchronized clock SCL is released and pulled high. There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

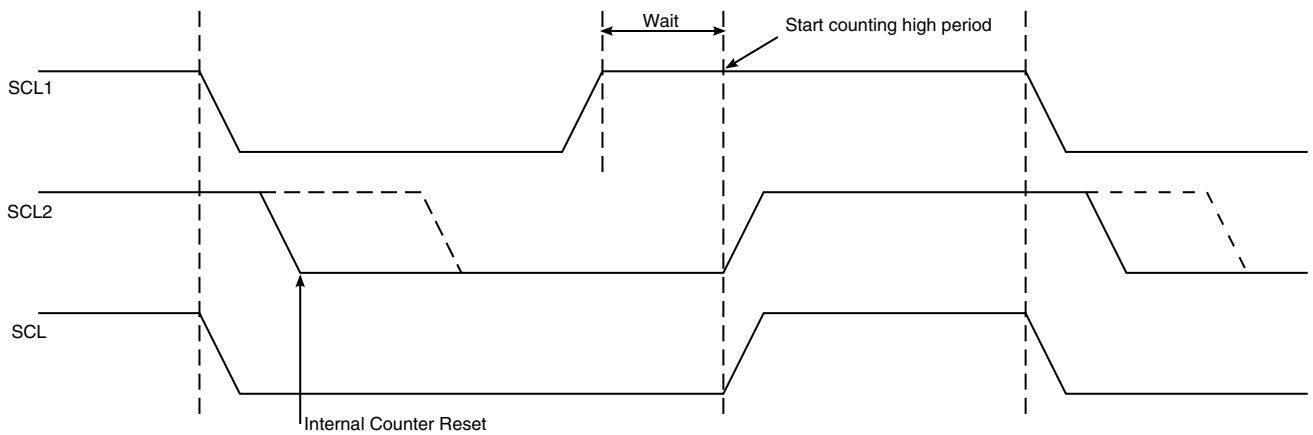


Figure 34-5. Synchronized Clock SCL

34.3.5 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold SCL low after completing one byte transfer (9 bits). In such a case, the clock mechanism halts the bus clock and forces the master clock into a wait state until the slave releases SCL.

34.3.6 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is longer than the master SCL low period, the resulting SCL bus signal low period is stretched.

34.3.7 Peripheral Bus Accesses

I2C is a 16-bit block. Only half-word accesses should be performed to the block.

34.3.8 Generation of Transfer Error on IP Bus

If an address is received on the Peripheral slave bus interface that is not implemented, an access error is generated.

34.3.9 Clocks

There are two input clocks for I2C.

1. Peripheral Clock:The clock is used for Peripheral bus register read/writes.
2. Module Clock:This is the functional clock of the I2C. The serial bit clock frequency is derived from the module clock. The module clock and peripheral clocks are synchronous to each other. The minimum frequency of module clock should be 12.8 MHz for Fast Mode to achieve 400 Kbps operation.

34.3.10 Reset

The I2C can be reset in two ways.

1. Global reset: A hard asynchronous reset of the whole I2C.
2. Software reset: An internal reset for whole I2C, except I2C_IADR and I2C_IFDR registers, is initiated by deasserting the I2C_I2CR[IEN] bit.

34.3.11 Interrupts

There is only one interrupt from the block. The interrupt is enabled by setting the I2C_I2CR[IIEN] bit.

The interrupt is generated in any one of the following conditions.

- One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock).
- An address is received that matches its own specific address in slave-receive mode.
- Arbitration is lost.

34.3.12 Byte Order

The block only supports the little endian mode.

34.4 Initialization

34.4.1 Initialization Sequence

Before the interface can transfer serial data, registers must be initialized, as follows:

1. Set the data sampling rate (I2C_IFDR[IC] to obtain SCL frequency from the system bus clock.
2. Update the address in the (I2C_IADR) to define its slave address (address can range from 0 to 0x7f).
3. Set the I2C enable bit (I2C_I2CR[IEN]) to enable the I2C bus interface system.
4. Modify the bits in the I2C_I2CR to select master/slave mode, transmit/receive mode, and interrupt-enable or not.

34.4.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the master transmitter mode. On a multiple-master bus system, the busy bus (I2C_I2SR[IBB]) must be tested to determine whether the serial bus is free. If the bus is free (IBB = 0), the START signal and the first byte (the slave address) can be sent. The data written to the data register comprises the address of the desired slave and the LSB indicates the transfer direction.

The free time between a STOP and the next START condition is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I2C is busy after writing the calling address to the data register (I2C_I2DR) before proceeding to load data into the data register (I2C_I2DR).

34.4.3 Post-Transfer Software Response

Sending or receiving a byte sets the data transferring bit (I2C_I2SR[ICF]), which indicates one byte communication is finished. Upon completion, the interrupt status (I2C_I2SR[IIF]) is also set. An external interrupt is generated if the interrupt enable (I2C_I2CR[IIEN]) is set. The software must first clear the interrupt status (I2C_I2SR[IIF]) in the interrupt routine.

(See the flow chart in Figure 34-7.) The data transferring bit (I2C_I2SR[ICF]) is cleared either by reading from I2C_I2DR in receive mode or by writing to this register in transmit mode.

The software can service the I2C I/O in the main program by monitoring the interrupt status (I2C_I2SR[IIF]) if the interrupt enable is deasserted. In this case, the interrupt status should be polled of the data transferring bit (I2C_I2SR[ICF]) because the operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle, the master is always in transmit mode; that is, the address is sent. If master receive mode is required, then I2C_I2CR[MTX] should be toggled and dummy read of I2C_I2DR register has to be done for triggering receive data.

During slave-mode address cycles (I2C_I2SR[IAAS] = 1), the slave read/write bit I2C_I2SR[SRW] is read to determine the direction of the next transfer. The transmit/receive bit (I2C_I2CR[MTX]) should also be programmed accordingly. For slave-mode data cycles (IAAS = 0), SRW is invalid. MTX should be read to determine the current transfer direction.

34.4.4 Generation of STOP

A data transfer ends when the master signals a STOP, which can occur after all data is sent.

For a master receiver to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting the transmit acknowledge bit (I2C_I2CR[TXAK]) before reading the next-to-last byte. Before the last byte is read, a STOP signal must be generated.

34.4.5 Generation of Repeated START

After the data transfer, if the master still wants the bus, it can signal another START followed by another slave address without signalling a STOP.

34.4.6 Slave Mode

In the slave interrupt service routine (see Figure 34-7), the block addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (I2C_I2CR[MTX]) according to the I2C_I2SR[SRW]. Writing to the I2C_I2CR clears the IAAS automatically. The only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer can now be initiated by writing information to I2C_I2DR for slave transmits, or read from I2C_I2DR in slave-receive mode. A dummy read of I2C_I2DR in slave/receive mode releases SCL, allowing the master to send data.

In the slave transmitter routine, the receive acknowledge bit (I2C_I2SR[RXAK]) must be tested before sending the next byte of data. Setting RXAK means an end-of-data signal from the master receiver, after which the software must switch it from transmitter to receiver mode. Reading the data register (I2C_I2DR) then releases SCL so that the master can generate a STOP signal.

34.4.7 Arbitration Lost

If several devices try to engage the bus at the same time, one becomes master. Hardware immediately switches devices that lose arbitration to slave receive mode. Data output to SDA stops, but SCL is still generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer if the arbitration is lost (I2C_I2SR[IAL] = 1), and the slave mode is selected (I2C_I2CR[MSTA] = 0).

See the flow chart in Figure 34-7.

If a device that is not a master tries to transmit or do a START, hardware inhibits the transmission, clears MSTA without signalling a STOP, generates an interrupt to the ARM platform, and sets I2C_I2SR[IAL] to indicate a failed attempt to engage the bus. When considering these cases, the slave service routine should first test I2C_I2SR[IAL], and the software should clear it if it is set.

For Multi-master mode, when an I2C is enabled when the bus is busy and asserts START, the I2C_I2SR[IAL] bit gets set only for SDA=0, SCL=0/1, SDA=1, and SCL=0, but not for SDA=1 and SCA=1, which is the same as bus idle state.

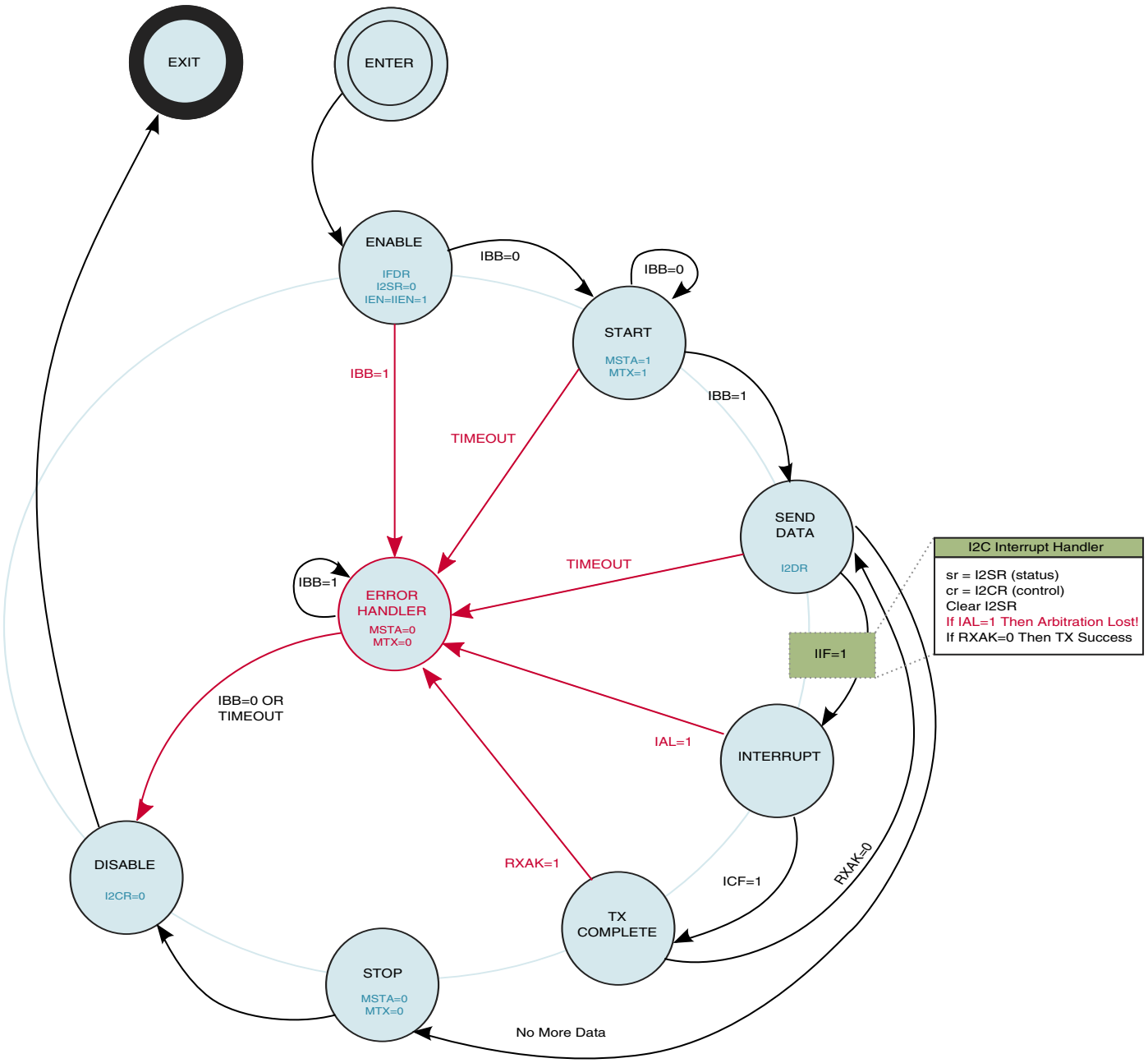


Figure 34-6. I2C Programming State Diagram

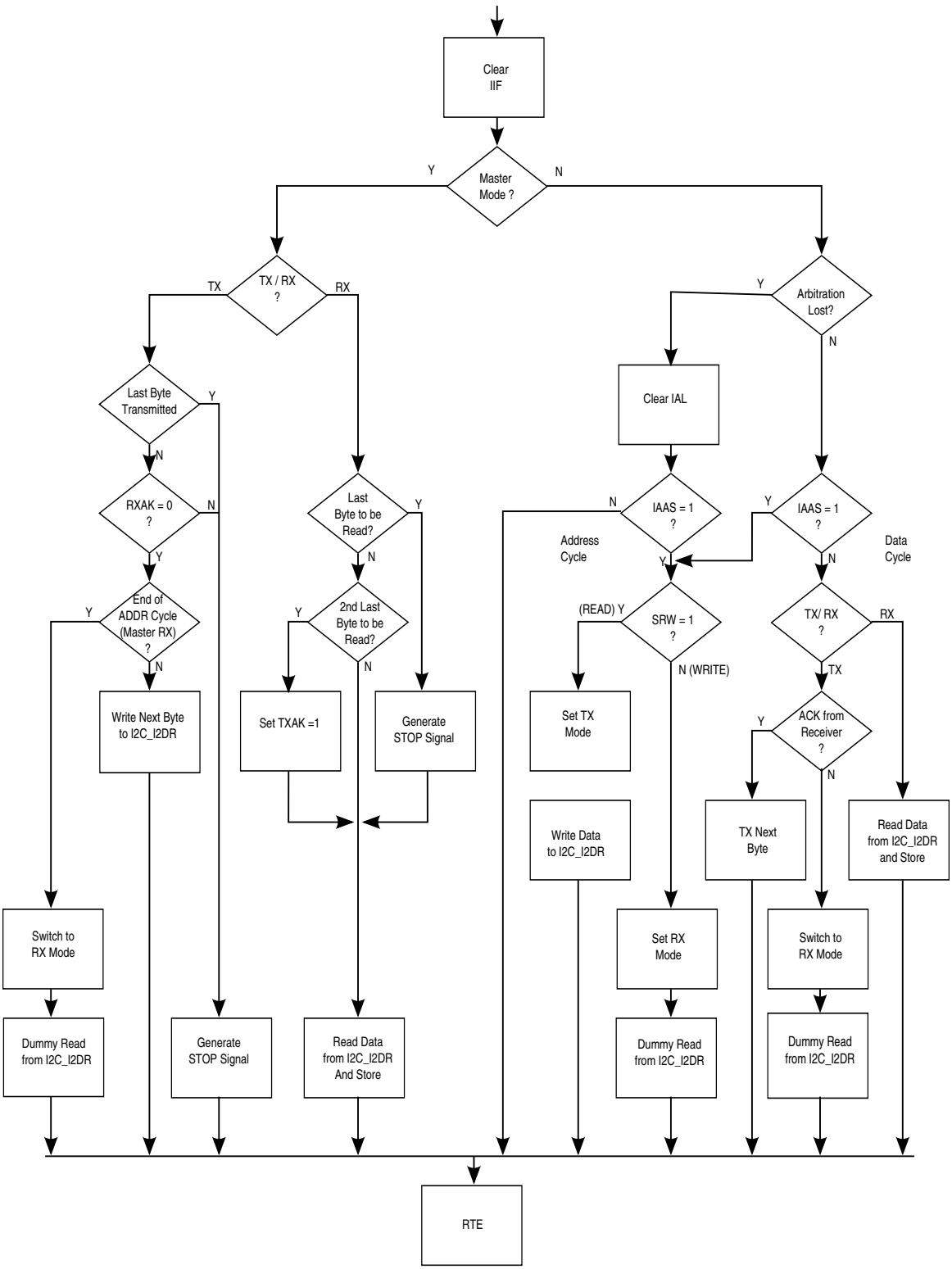


Figure 34-7. Flow-Chart of Typical I2C Interrupt Routine

NOTE

For a repeated start-only, the stop generation stage will not occur in master mode. A loop will repeat itself without stopping for the next start.

For Master Rx mode, I2C is programmed as master transmit during address mode and after slave address transfer, MTX bit should be cleared and Dummy read on I2C_I2DR register should be performed so that I2C can read the next receive data.

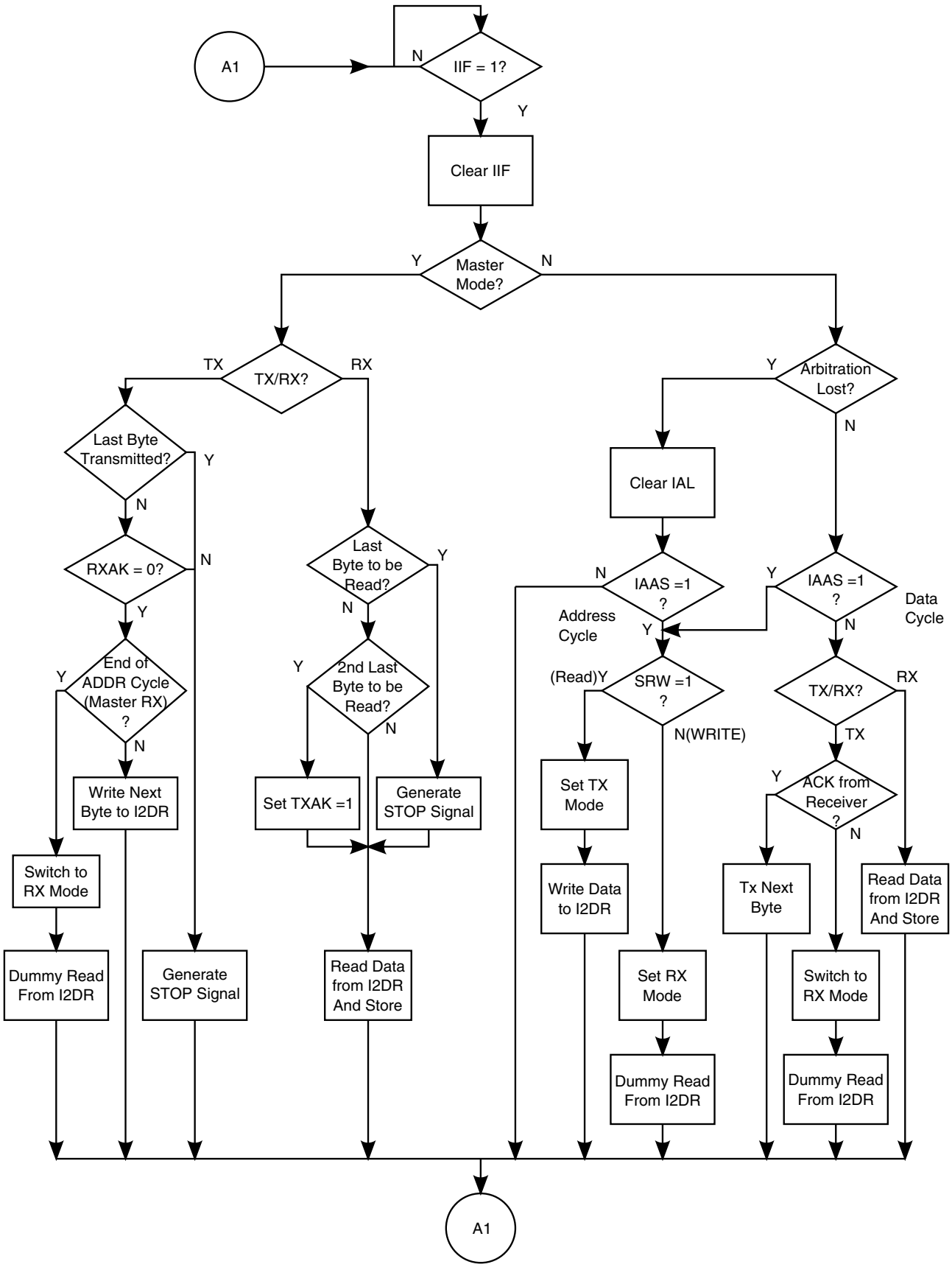


Figure 34-8. Flow Chart for Typical I2C Polling Routine

NOTE

The time-out value will depend on the bus frequency at which I2C is operating. The Min. Time-out for polling IIF bit at I2C Max bus frequency of 400KHz is, T_{min} = 25 μs (= 2.5*10 μs). This value can be interpolated for any bus frequency.

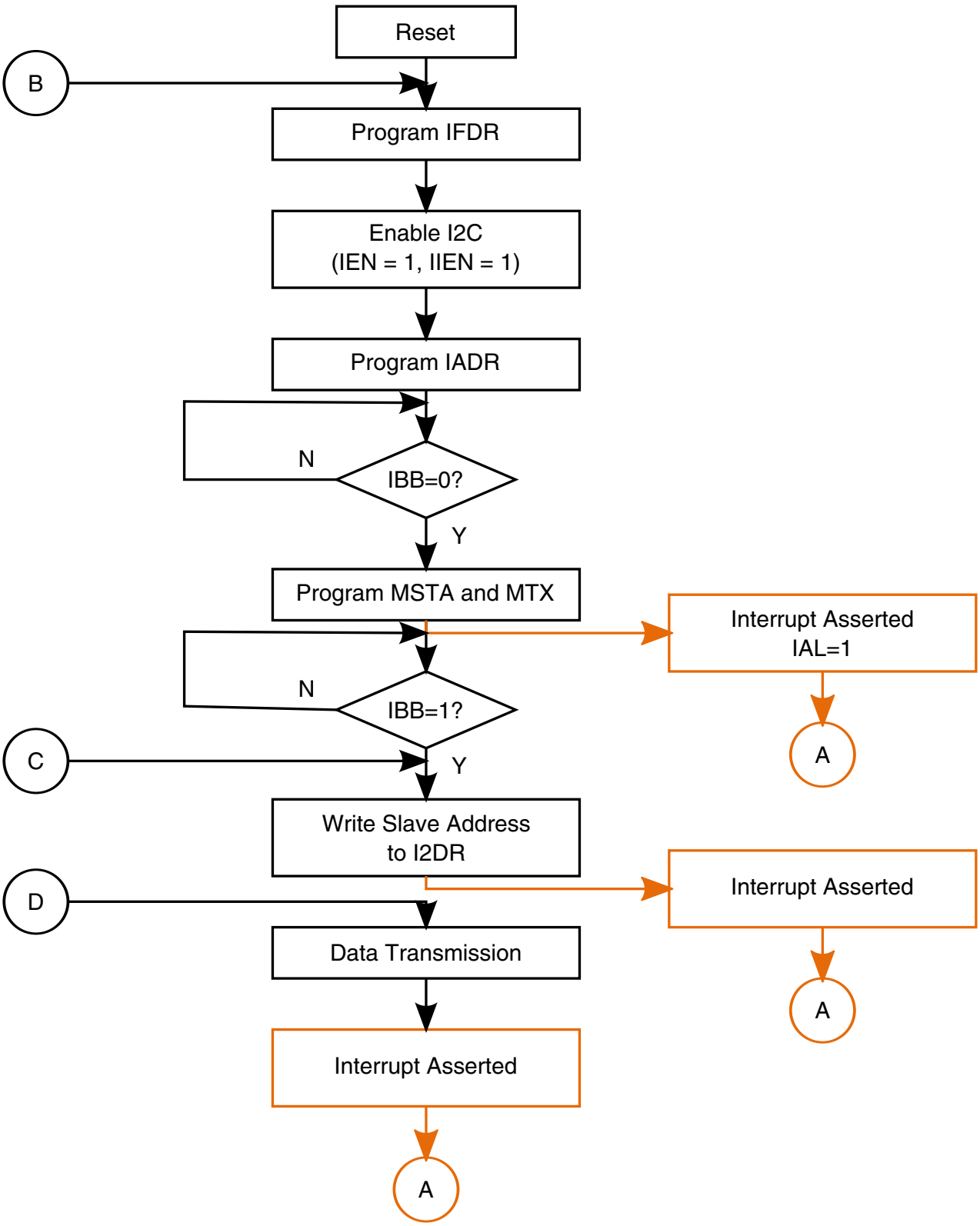


Figure 34-9. Detailed Flow Chart of a Typical I2C Master Tx Mode, Part 1

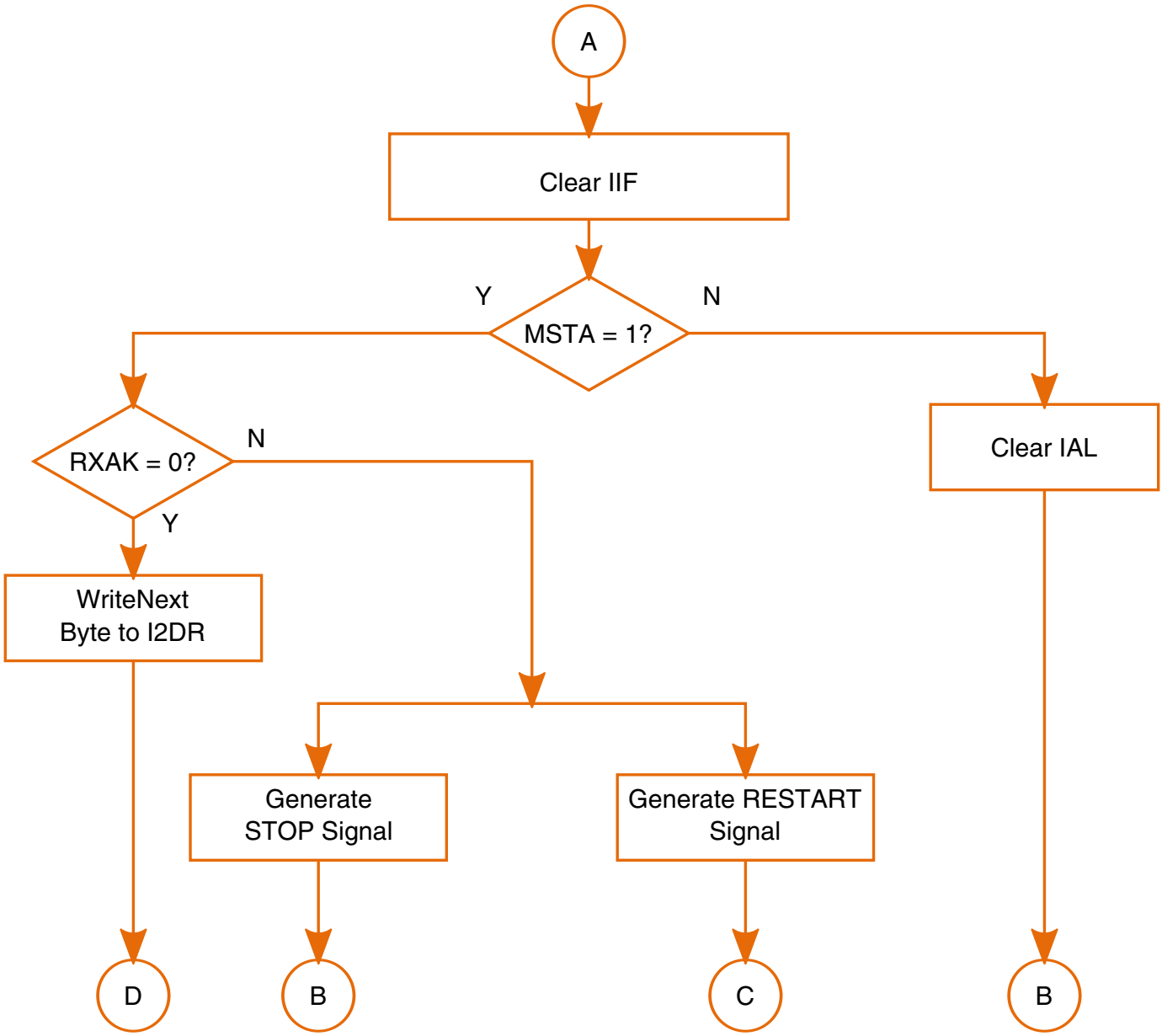


Figure 34-10. Detailed Flow Chart of a Typical I2C Master Tx Mode, Part 2

Figure 34-9 and Figure 34-10 show the Master Transmit mode operation with interrupt sub-routine. In case interrupt is generated and the MSTA bit is 0, then bus arbitration is lost and IAL is set. Software can clear IAL bit and re-program I2C. If MSTA bit is 1, then it will be a transfer done interrupt and software can check RXAK bit for data receive acknowledgement by slave and accordingly decide to either generate STOP, REPEAT START by writing in I2C_I2CR register or next data transfer by writing into I2C_I2DR register.

NOTE

The IBB bit is asserted by START condition on the bus, and it is deasserted by STOP condition on bus. Therefore, if arbitration is lost due to an unexpected STOP condition during transfer, then IBB is cleared. If arbitration is lost due to data mismatch, then it will not be cleared. Software should always clear the IEN bit and then set it if arbitration is lost.

34.5 Software Restriction

Software should take care that there is a delay of at least two Module Clock cycles after it sets the I2C_I2CR[RSTA] bit before writing to the I2C_I2DR register. Maximum possible clock period of Module Clock is 78 ns.

34.6 Programmable Registers

34.6.1 I2C Memory Map/Register Definition

The I2C contains five 16-bit registers.

NOTE

Registers at offsets 0x0002, 0x0006, 0x000A, and 0x000E are reserved for future additions.

I2C memory map

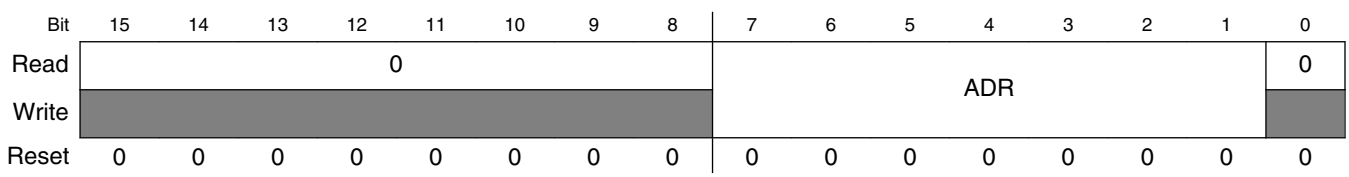
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FE_C000	I2C Address Register (I2C-3_IADR)	16	R/W	0000h	34.61.1/ 1645
53FE_C004	I2C Frequency Divider Register (I2C-3_IFDR)	16	R/W	0000h	34.61.2/ 1646
53FE_C008	I2C Control Register (I2C-3_I2CR)	16	R/W	0000h	34.61.3/ 1647
53FE_C00C	I2C Status Register (I2C-3_I2SR)	16	R/W	0081h	34.61.4/ 1649
53FE_C010	I2C Data I/O Register (I2C-3_I2DR)	16	R/W	0000h	34.61.5/ 1650

I2C memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FC_4000	I2C Address Register (I2C-2_IADR)	16	R/W	0000h	34.61.1/ 1645
63FC_4004	I2C Frequency Divider Register (I2C-2_IFDR)	16	R/W	0000h	34.61.2/ 1646
63FC_4008	I2C Control Register (I2C-2_I2CR)	16	R/W	0000h	34.61.3/ 1647
63FC_400C	I2C Status Register (I2C-2_I2SR)	16	R/W	0081h	34.61.4/ 1649
63FC_4010	I2C Data I/O Register (I2C-2_I2DR)	16	R/W	0000h	34.61.5/ 1650
63FC_8000	I2C Address Register (I2C-1_IADR)	16	R/W	0000h	34.61.1/ 1645
63FC_8004	I2C Frequency Divider Register (I2C-1_IFDR)	16	R/W	0000h	34.61.2/ 1646
63FC_8008	I2C Control Register (I2C-1_I2CR)	16	R/W	0000h	34.61.3/ 1647
63FC_800C	I2C Status Register (I2C-1_I2SR)	16	R/W	0081h	34.61.4/ 1649
63FC_8010	I2C Data I/O Register (I2C-1_I2DR)	16	R/W	0000h	34.61.5/ 1650

34.61.1 I2C Address Register (I2Cx_IADR)

Addresses: I2C-3_IADR is 53FE_C000h base + 0h offset = 53FE_C000h
I2C-2_IADR is 63FC_4000h base + 0h offset = 63FC_4000h
I2C-1_IADR is 63FC_8000h base + 0h offset = 63FC_8000h



I2Cx_IADR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–1 ADR	Slave address. Contains the specific slave address to be used by the I2C. Slave mode is the default I2C mode for an address match on the bus.

Table continues on the next page...

I2Cx_IADR field descriptions (continued)

Field	Description
	NOTE: The I2C_IADR holds the address the I2C responds to when addressed as a slave. The slave address is not the address sent on the bus during the address transfer. The register is not reset by a software reset.
0 Reserved	This read-only field is reserved and always has the value zero. Reserved

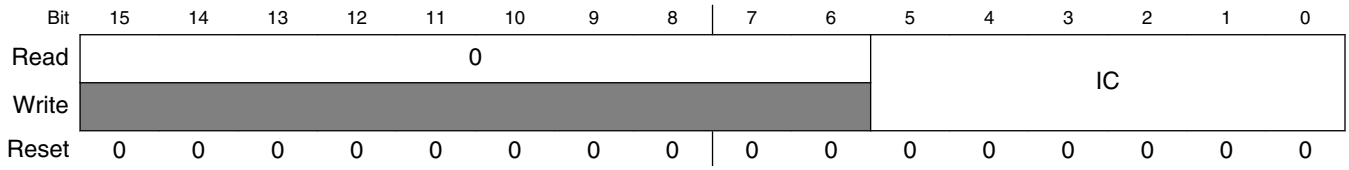
34.61.2 I2C Frequency Divider Register (I2Cx_IFDR)

The I2C_IFDR provides a programmable prescaler to configure the clock for bit-rate selection. The register does not get reset by software reset. The following table describes the Divider values for register field "IC". Table below describes the register values for field "IC".

Table 34-12. I2C_IFDR Register Field Values

IC	Divider		IC	Divider		IC	Divider		IC	Divider
0x00	30		0x10	288		0x20	22		0x30	160
0x01	32		0x11	320		0x21	24		0x31	192
0x02	36		0x12	384		0x22	26		0x32	224
0x03	42		0x13	480		0x23	28		0x33	256
0x04	48		0x14	576		0x24	32		0x34	320
0x05	52		0x15	640		0x25	36		0x35	384
0x06	60		0x16	768		0x26	40		0x36	448
0x07	72		0x17	960		0x27	44		0x37	512
0x08	80		0x18	1152		0x28	48		0x38	640
0x09	88		0x19	1280		0x29	56		0x39	768
0x0A	104		0x1A	1536		0x2A	64		0x3A	896
0x0B	128		0x1B	1920		0x2B	72		0x3B	1024
0x0C	144		0x1C	2304		0x2C	80		0x3C	1280
0x0D	160		0x1D	2560		0x2D	96		0x3D	1536
0x0E	192		0x1E	3072		0x2E	112		0x3E	1792
0x0F	240		0x1F	3840		0x2F	128		0x3F	2048

Addresses: I2C-3_IFDR is 53FE_C000h base + 4h offset = 53FE_C004h
I2C-2_IFDR is 63FC_4000h base + 4h offset = 63FC_4004h
I2C-1_IFDR is 63FC_8000h base + 4h offset = 63FC_8004h



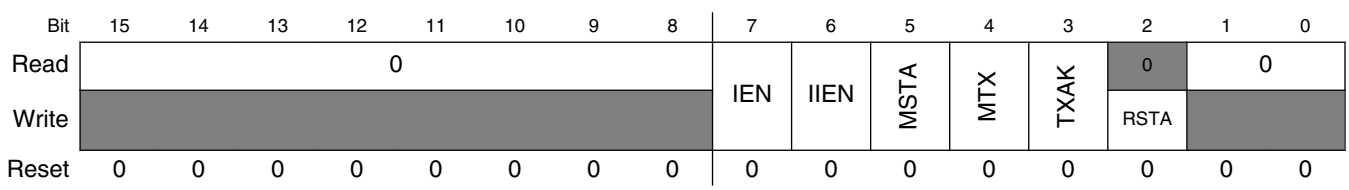
I2Cx_IFDR field descriptions

Field	Description
15–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5–0 IC	I2C clock rate. Pre scales the clock for bit-rate selection. Due to potentially slow SCL and SDA rise and fall times, bus signals are sampled at the prescaler frequency. The serial bit clock frequency is equal to IPG_CLK_ROOT divided by the divider shown in I2C Data I/O Register . **NOTE:** The IC value should not be changed during the data transfer, however, it can be changed before REPEAT START or START programming sequence in I2C. The I2C protocol supports bit rates up to 400 kbps. The IC bits need to be programmed in accordance with this constraint.

34.61.3 I2C Control Register (I2Cx_I2CR)

The I2C_I2CR is used to enable the I2C and the I2C interrupt. It also contains bits that govern operation as a slave or a master.

Addresses: I2C-3_I2CR is 53FE_C000h base + 8h offset = 53FE_C008h
I2C-2_I2CR is 63FC_4000h base + 8h offset = 63FC_4008h
I2C-1_I2CR is 63FC_8000h base + 8h offset = 63FC_8008h



I2Cx_I2CR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 IEN	I2C enable. Also controls the software reset of the entire I2C. Resetting the bit generates an internal reset to the block. If the block is enabled in the middle of a byte transfer, slave mode ignores the current bus transfer and starts operating when the next start condition is detected. Master mode is not aware that the bus is busy so initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I2C to lose arbitration. After which, bus operation returns to normal.

Table continues on the next page...

I2Cx_I2CR field descriptions (continued)

Field	Description
	0 The block is disabled, but registers can still be accessed. 1 The I2C is enabled. This bit must be set before any other I2C_I2CR bits have any effect.
6 IIEN	I2C interrupt enable. **NOTE:** If data is written during the START condition, that is, just after setting the I2C_I2CR[MSTA] and I2C_I2CR[MTX] bits, then the ICF bit is cleared at the falling edge of SCLK after START. If data is written after the START condition and falling edge of SCLK, then ICF bit is cleared as soon as data is written. 0 I2C interrupts are disabled, but the status flag I2C_I2SR[IIF] continues to be set when an interrupt condition occurs. 1 I2C interrupts are enabled. An I2C interrupt occurs if I2C_I2SR[IIF] is also set.
5 MSTA	Master/slave mode select bit. If the master loses arbitration, MSTA is cleared without generating a STOP signal. **NOTE:** Module clock should be on for writing to the MSTA bit. **NOTE:** The MSTA bit is cleared by software to generate a STOP condition; it can also be cleared by hardware when the I2C loses the bus arbitration. 0 Slave mode. Changing MSTA from 1 to 0 generates a STOP and selects slave mode. 1 Master mode. Changing MSTA from 0 to 1 signals a START on the bus and selects master mode.
4 MTX	Transmit/receive mode select bit. Selects the direction of master and slave transfers. 0 Receive. When a slave is addressed, the software should set MTX according to the slave read/write bit in the I2C status register (I2C_I2SR[SRW]). 1 Transmit. In master mode, MTX should be set according to the type of transfer required. Therefore, for address cycles, MTX is always 1.
3 TXAK	Transmit acknowledge enable. Specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. **NOTE:** Writing TXAK applies only when the I2C bus is a receiver. 0 An acknowledge signal is sent to the bus at the ninth clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, the acknowledge bit = 1).
2 RSTA	Repeat start. Always reads as 0. Attempting a repeat start without bus mastership causes loss of arbitration. 0 No repeat start 1 Generates a repeated START condition
1–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

34.61.4 I2C Status Register (I2Cx_I2SR)

The I2C_I2SR contains bits that indicate transaction direction and status.

Addresses: I2C-3_I2SR is 53FE_C000h base + Ch offset = 53FE_C00Ch
I2C-2_I2SR is 63FC_4000h base + Ch offset = 63FC_400Ch
I2C-1_I2SR is 63FC_8000h base + Ch offset = 63FC_800Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
Write																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

I2Cx_I2SR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 ICF	Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer is in progress. 1 Transfer is complete. This bit is set by the falling edge of the ninth clock of the last byte transfer.
6 IAAS	I2C addressed as a slave bit. The ARM platform is interrupted if the interrupt enable (I2C_I2CR[IIEN]) is set. The ARM platform must check the slave read/write bit (SRW) and set its TX/RX mode accordingly. Writing to I2C_I2CR clears this bit. 0 Not addressed 1 Addressed as a slave. Set when its own address (I2C_IADR) matches the calling address.
5 IBB	I2C bus busy bit. Indicates the status of the bus. **NOTE:** When I2C is enabled (I2C_I2CR[IEN] = 1), it continuously polls the bus data (SDAK) and clock (SCLK) signals to determine a START or STOP condition. 0 Bus is idle. If a STOP signal is detected, IBB is cleared. 1 Bus is busy. When START is detected, IBB is set.
4 IAL	Arbitration lost. Set by hardware in the following circumstances (IAL must be cleared by software by writing a "0" to it at the start of the interrupt service routine): • SDA input sampled low when the master drives high during an address or data-transmit cycle. • SDA input sampled low when the master drives high during the acknowledge bit of a data-receive cycle. For the above two cases, the bit is set at the falling edge of 9th SCL clock during the ACK cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. **NOTE:** Software cannot set the bit. 0 No arbitration lost. 1 Arbitration is lost.

Table continues on the next page...

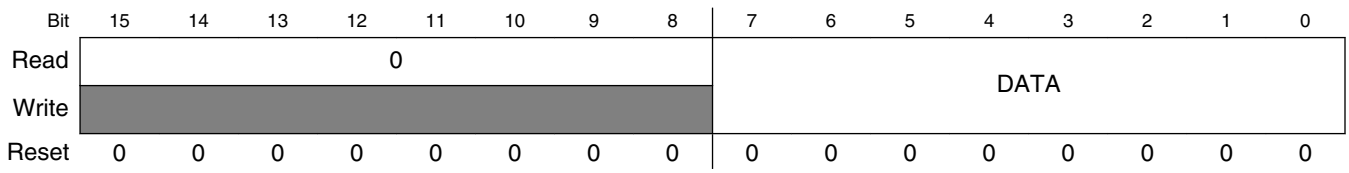
I2Cx_I2SR field descriptions (continued)

Field	Description
3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2 SRW	Slave read/write. When the I2C is addressed as a slave, IAAS is set, and the slave read/write bit (SRW) indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I2C is a slave and has an address match. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IIF	I2C interrupt. Must be cleared by the software by writing a "0" to it in the interrupt routine. **NOTE:** The software cannot set the bit. 0 No I2C interrupt pending. 1 An interrupt is pending. This causes a processor interrupt request (if the interrupt enable is asserted [IIEN = 1]). The interrupt is set when one of the following occurs: • One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock). • An address is received that matches its own specific address in slave-receive mode. • Arbitration is lost.
0 RXAK	Received acknowledge. This is the value received of the SDA input for the acknowledge bit during a bus cycle. 0 An "acknowledge" signal was received after the completion of an 8-bit data transmission on the bus. 1 A "No acknowledge" signal was detected at the ninth clock.

34.61.5 I2C Data I/O Register (I2Cx_I2DR)

In master-receive mode, reading the data register allows a read to occur and initiates the next byte to be received. In slave mode, the same function is available after it is addressed.

Addresses: I2C-3_I2DR is 53FE_C000h base + 10h offset = 53FE_C010h
I2C-2_I2DR is 63FC_4000h base + 10h offset = 63FC_4010h
I2C-1_I2DR is 63FC_8000h base + 10h offset = 63FC_8010h



I2Cx_I2DR field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–0 DATA	Data Byte. Holds the last data byte received or the next data byte to be transferred. Software writes the next data byte to be transmitted or reads the data byte received. **NOTE:** The core-written value in I2C_I2DR cannot be read back by the core. Only data written by the I2C bus side can be read.

Chapter 35
IOMUXC

35.1 Overview

The IOMUX Controller (IOMUXC), together with the IOMUX, enables the IC to share one PAD to several functional blocks. The sharing is done by multiplexing the PAD input/output signals. For each PAD there are up to 8 muxing options (called ALT modes). Since different modules require different PAD settings (like pull up, keeper, and so on) the IOMUXC controls also the PAD settings parameters.

The IOMUX consist of only combinatorial logic combined from several basic iomux cells, each basic iomux cell handles only one pad signals muxing.

Figure 35-1 illustrates the IOMUX/IOMUXC connectivity in the system.

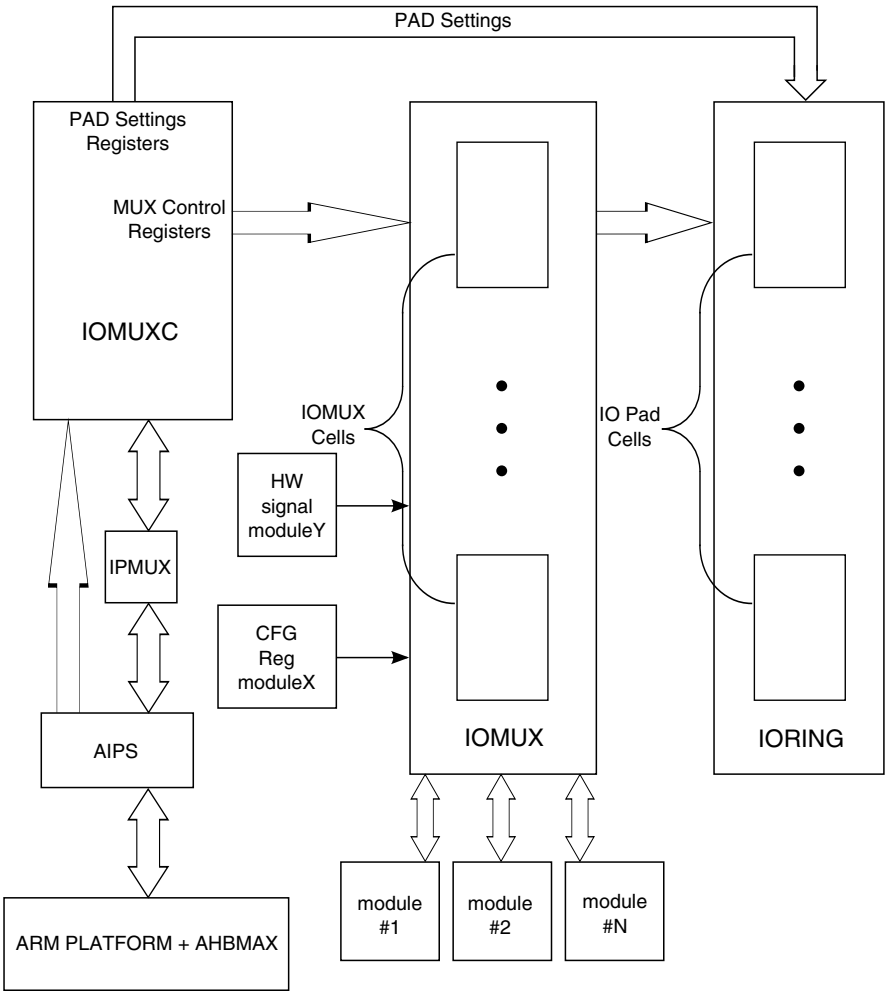


Figure 35-1. IOMUX SoC Level Block Diagram

35.1.1 Features

The IOMUXC features are:

- 32 bits SW Mux control registers (IOMUXC_SW_MUX_CTL_PAD_<PAD NAME> or IOMUXC_SW_MUX_CTL_GRP_<GROUP NAME>) to configure a specific mux mode of each pad or a predefined group of pads and to enable forcing the input path of the pad/s (SION bit).
- 32 bits SW Pad control registers (IOMUXC_SW_PAD_CTL_PAD_<PAD_NAME> or IOMUXC_SW_PAD_CTL_GRP_<GROUP NAME>) to configure specific pad settings of each pad or a predefined group of pads.
- 32 bits General Purpose Registers-Two 32 bits registers according to SOC requirements for any usage.

- 32 bits "Daisy Chain" Control registers-register to control the input path to a module when more than one pad may drive this module input.
- IPS Interface-Sky Blue peripheral bus to allow registers configuration and readiness (all registers are read/write registers).

Each SW MUX/PAD CTL IOMUXC register handles only one pad or one pads group.

Only the minimum number of registers required by SW will be implemented by HW, for example, if only ALT0 and ALT1 modes are used on Pad X then only one bit register will be generated as the MUX_MODE control field in the SW Mux control register of Pad X.

The SW Mux control registers may allow also forcing the pads to become input (input path enabled) regardless of the functional direction driven. This may be useful for loopback and GPIO data capture.

35.1.2 Modes of Operation

There is only one mode of operation-normal mode.

35.2 External Signal Description

There are no signals in the IOMUX that connect off chip (they all control the PADS functionality).

35.3 Functional Description

The IOMUXC Detailed Block Diagram is illustrated in Figure 35-2.

The IOMUXC is consist of 2 sub blocks:

- IOMUXC_REGISTERS: includes all the IOMUXC registers (6 main types are illustrated).
- IOMUXC_LOGIC: includes all the IOMUXC combinatorial logic (IP interface controls, address decoder, observability muxes).

The IOMUXC external signals and the IOMUXC sub blocks connections internal signals are also detailed in this diagram. The following are the naming conventions used along with some explanation:

- <PAD>-any PAD logic name.

- <GRP>-any PAD group logic name.
- <FUNC>-one of the following pad control functions:
  - SRE (1 bit slew rate control).
  - DSE (2 bits drive strength control).
  - ODE (1 bit open drain control).
  - HYS (1 bit hysteresis control).
  - PULL_KEEP_CTL (4 bits pull up/down and keeper controls)
  - PUS (2 bits pull up/down configuration value)
  - PUE (1 bit pull/keep select)
  - PKE (1 bit enable/disable pull up, pull down or keeper capability)
  - DDR_MODE_SEL (1 bit ddr_mode control)
  - DDR_INPUT (1 bit ddr_input control)
- ips_addr[X:2] - since the IOMUXC registers are all 32 bits width address lines 0,1 are not used. X will be calculated automatically by the tool according to the final register number: X is smallest integer value which satisfy the constraint: 2^(X-1) >= (Total number of IOMUXC registers).

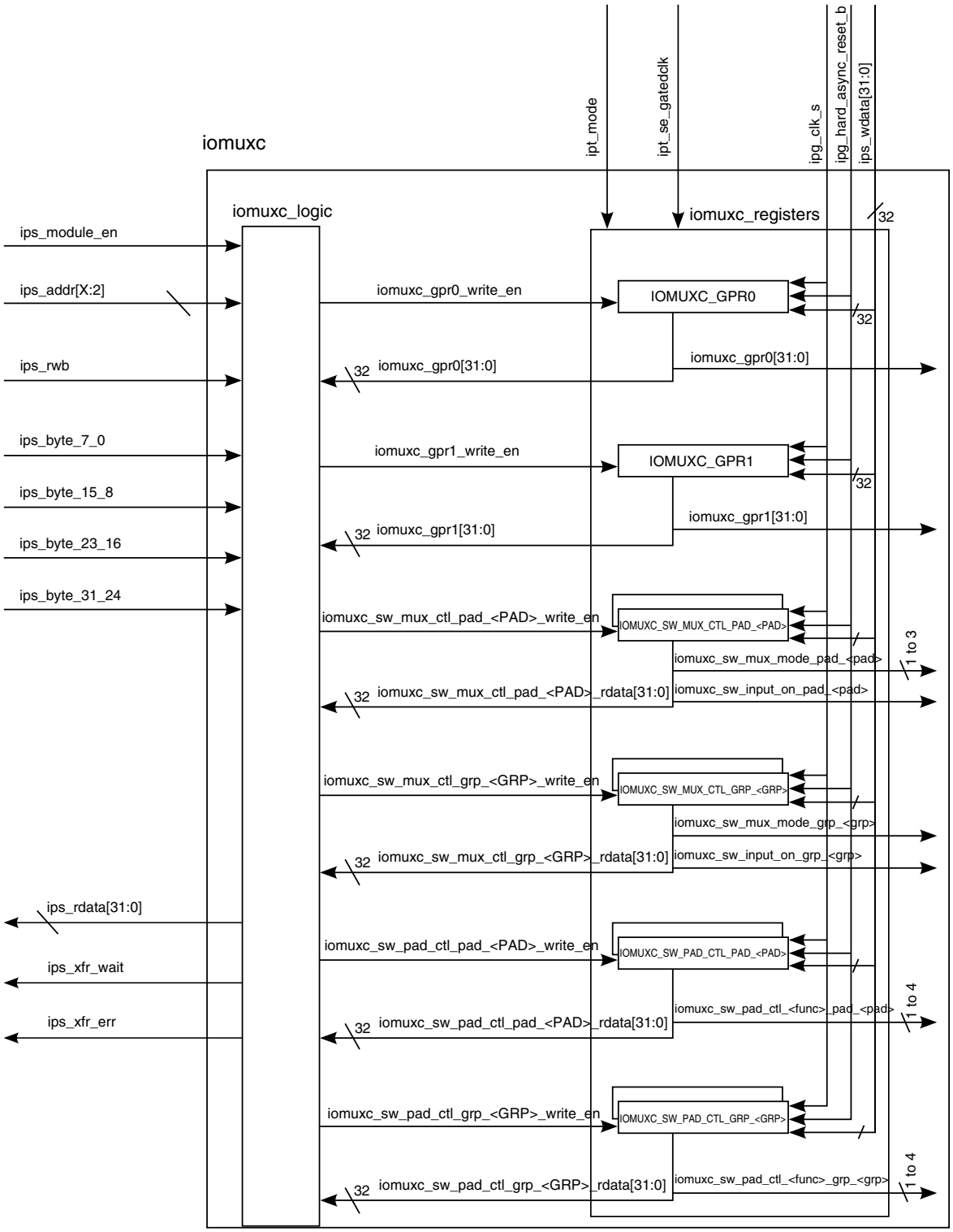


Figure 35-2. IOMUXC Block Diagram

The IOMUX consist of a number (about the number of pads in the SoC) of basic iomux_cell units. If only one functional mode is required for a specific pad then there is no need for IOMUXing and the signals can be connected directly from the module to the IORING. IOMUX cell is required whenever 2 or more functional modes are required for a specific pad or when one functional mode and the one test mode are required.

The basic iomux_cell design, which allows 2 levels of HW signal control (in ALT6 and ALT7 modes - ALT7 gets highest priority) is shown in Figure 35-3.

35.3.1 ALT6 and ALT7 Extended Muxing Modes

The ALT7 and ALT6 Extended Muxing Modes allows any signal in the system (Fuse, Pad input, JTAG, SW register, etc) to override any SW configuration and to force the ALT6/ALT7 Muxing Mode. It also allows an IOMUX SW register to control a group of pads.

35.3.2 SW Loopback through SION Bit

A limited but satisfying option exist to override the regular pad functionality and force the input path to be active (ipp_ibe=1'b1) regardless of the value driven by the corresponding module. This can be done by setting the SION (SW Input On) bit in the IOMUXC_SW_MUX_CTL register (when available) to "1" and can be used for:

- LoopBack-Module X drives the pad and also receive pad value as an input.
- GPIO Capture-Module X drives the pad and the value is captured by GPIO.

NOTE

Software loopback is not affecting DRAM MC PADS.

35.3.3 "Daisy Chain"-Multi Pads Driving Same Module Input Pin

In some cases, more than one pad may drive a single module input pin.

Such cases require adding one more level of IOMUXing: all these input signals are muxed and a dedicated SW controlled register controls the mux in order to select the required input path. This means that a pad involved in such "Daisy Chain" situation may require 2 SW configuration commands: one for selecting the mode for this pad and one for defining it as the input path.

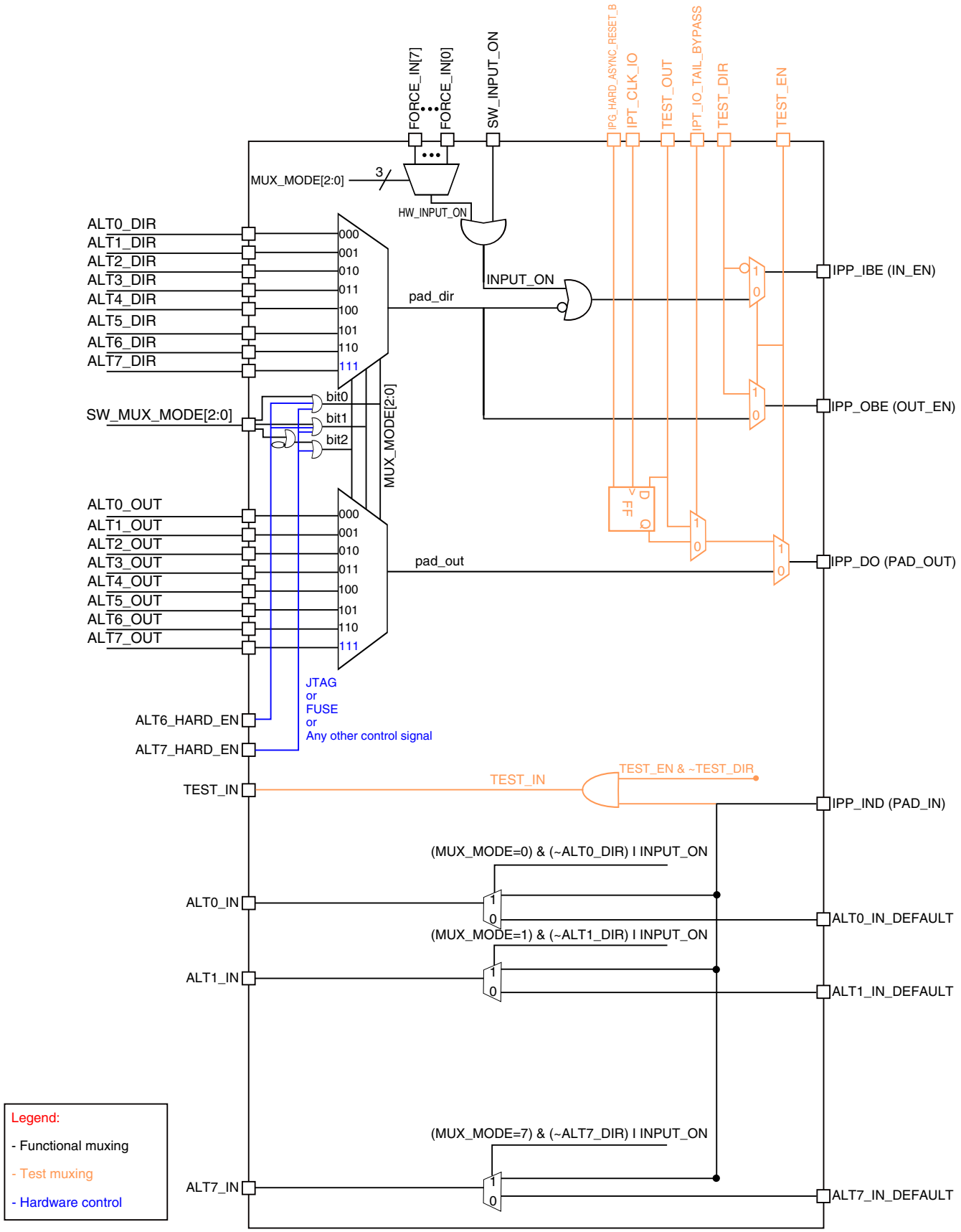


Figure 35-3. IOMUX Cell Block Diagram

35.3.4 Interrupts

There are no interrupts that are generated by the IOMUX controller.

35.4 Programmable Registers

IOMUXC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_8000	GPR0 (IOMUXC_GPR0)	32	R/W	0000_0000h	35.4.1/ 1686
53FA_8004	GPR1 (IOMUXC_GPR1)	32	R/W	0019_0005h	35.4.2/ 1687
53FA_8008	GPR2 (IOMUXC_GPR2)	32	R/W	0000_0000h	35.4.3/ 1688
53FA_800C	OBSERVE_MUX_0 (IOMUXC_OBSMUX0)	32	R/W	0000_0000h	35.4.4/ 1689
53FA_8010	OBSERVE_MUX_1 (IOMUXC_OBSMUX1)	32	R/W	0000_0000h	35.4.5/ 1690
53FA_8014	OBSERVE_MUX_2 (IOMUXC_OBSMUX2)	32	R/W	0000_0000h	35.4.6/ 1691
53FA_8018	OBSERVE_MUX_3 (IOMUXC_OBSMUX3)	32	R/W	0000_0000h	35.4.7/ 1692
53FA_801C	OBSERVE_MUX_4 (IOMUXC_OBSMUX4)	32	R/W	0000_0000h	35.4.8/ 1693
53FA_8020	SW_MUX_CTL_PAD_KEY_COL0 (IOMUXC_SMUXC_PKC0)	32	R/W	0000_0001h	35.4.9/ 1695
53FA_8024	SW_MUX_CTL_PAD_KEY_ROW0 (IOMUXC_SMUXC_PKR0)	32	R/W	0000_0001h	35.4.10/ 1696
53FA_8028	SW_MUX_CTL_PAD_KEY_COL1 (IOMUXC_SMUXC_PKC1)	32	R/W	0000_0001h	35.4.11/ 1697
53FA_802C	SW_MUX_CTL_PAD_KEY_ROW1 (IOMUXC_SMUXC_PKR1)	32	R/W	0000_0001h	35.4.12/ 1698
53FA_8030	SW_MUX_CTL_PAD_KEY_COL2 (IOMUXC_SMUXC_PKC2)	32	R/W	0000_0001h	35.4.13/ 1699
53FA_8034	SW_MUX_CTL_PAD_KEY_ROW2 (IOMUXC_SMUXC_PKR2)	32	R/W	0000_0001h	35.4.14/ 1700
53FA_8038	SW_MUX_CTL_PAD_KEY_COL3 (IOMUXC_SMUXC_PKC3)	32	R/W	0000_0001h	35.4.15/ 1701

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_803C	SW_MUX_CTL_PAD_KEY_ROW3 (IOMUXC_SMUXC_PKR3)	32	R/W	0000_0001h	35.4.16/ 1702
53FA_8040	SW_MUX_CTL_PAD_I2C1_SCL (IOMUXC_SMUXC_PI2C1_SCL)	32	R/W	0000_0001h	35.4.17/ 1703
53FA_8044	SW_MUX_CTL_PAD_I2C1_SDA (IOMUXC_SMUXC_PI2C1_SDA)	32	R/W	0000_0001h	35.4.18/ 1704
53FA_8048	SW_MUX_CTL_PAD_I2C2_SCL (IOMUXC_SMUXC_PI2C2_SCL)	32	R/W	0000_0001h	35.4.19/ 1705
53FA_804C	SW_MUX_CTL_PAD_I2C2_SDA (IOMUXC_SMUXC_PI2C2_SDA)	32	R/W	0000_0001h	35.4.20/ 1706
53FA_8050	SW_MUX_CTL_PAD_I2C3_SCL (IOMUXC_SMUXC_PI2C3_SCL)	32	R/W	0000_0001h	35.4.21/ 1707
53FA_8054	SW_MUX_CTL_PAD_I2C3_SDA (IOMUXC_SMUXC_PI2C3_SDA)	32	R/W	0000_0001h	35.4.22/ 1708
53FA_8058	SW_MUX_CTL_PAD_PWM1 (IOMUXC_SMUXC_PPWM1)	32	R/W	0000_0001h	35.4.23/ 1709
53FA_805C	SW_MUX_CTL_PAD_PWM2 (IOMUXC_SMUXC_PPWM2)	32	R/W	0000_0001h	35.4.24/ 1710
53FA_8060	SW_MUX_CTL_PAD_OWIRE (IOMUXC_SMUXC_POWIRE)	32	R/W	0000_0001h	35.4.25/ 1711
53FA_8064	SW_MUX_CTL_PAD_EPITO (IOMUXC_SMUXC_PEPITO)	32	R/W	0000_0001h	35.4.26/ 1712
53FA_8068	SW_MUX_CTL_PAD_WDOG (IOMUXC_SMUXC_PWDOG)	32	R/W	0000_0001h	35.4.27/ 1713
53FA_806C	SW_MUX_CTL_PAD_SSI_TXFS (IOMUXC_SMUXC_PSSI_TXFS)	32	R/W	0000_0001h	35.4.28/ 1714
53FA_8070	SW_MUX_CTL_PAD_SSI_TXC (IOMUXC_SMUXC_PSSI_TXC)	32	R/W	0000_0001h	35.4.29/ 1715
53FA_8074	SW_MUX_CTL_PAD_SSI_TXD (IOMUXC_SMUXC_PSSI_TXD)	32	R/W	0000_0001h	35.4.30/ 1716
53FA_8078	SW_MUX_CTL_PAD_SSI_RXD (IOMUXC_SMUXC_PSSI_RXD)	32	R/W	0000_0001h	35.4.31/ 1717
53FA_807C	SW_MUX_CTL_PAD_SSI_RXF (IOMUXC_SMUXC_PSSI_RXF)	32	R/W	0000_0001h	35.4.32/ 1718
53FA_8080	SW_MUX_CTL_PAD_SSI_RXC (IOMUXC_SMUXC_PSSI_RXC)	32	R/W	0000_0001h	35.4.33/ 1719
53FA_8084	SW_MUX_CTL_PAD_UART1_TXD (IOMUXC_SMUXC_PUART1_TXD)	32	R/W	0000_0001h	35.4.34/ 1720
53FA_8088	SW_MUX_CTL_PAD_UART1_RXD (IOMUXC_SMUXC_PUART1_RXD)	32	R/W	0000_0001h	35.4.35/ 1721

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_808C	SW_MUX_CTL_PAD_UART1_CTS (IOMUXC_SMUXC_PUART1_CTS)	32	R/W	0000_0001h	35.4.36/ 1722
53FA_8090	SW_MUX_CTL_PAD_UART1_RTS (IOMUXC_SMUXC_PUART1_RTS)	32	R/W	0000_0001h	35.4.37/ 1723
53FA_8094	SW_MUX_CTL_PAD_UART2_TXD (IOMUXC_SMUXC_PUART2_TXD)	32	R/W	0000_0001h	35.4.38/ 1724
53FA_8098	SW_MUX_CTL_PAD_UART2_RXD (IOMUXC_SMUXC_PUART2_RXD)	32	R/W	0000_0001h	35.4.39/ 1725
53FA_809C	SW_MUX_CTL_PAD_UART2_CTS (IOMUXC_SMUXC_PUART2_CTS)	32	R/W	0000_0001h	35.4.40/ 1726
53FA_80A0	SW_MUX_CTL_PAD_UART2_RTS (IOMUXC_SMUXC_PUART2_RTS)	32	R/W	0000_0001h	35.4.41/ 1727
53FA_80A4	SW_MUX_CTL_PAD_UART3_TXD (IOMUXC_SMUXC_PUART3_TXD)	32	R/W	0000_0001h	35.4.42/ 1728
53FA_80A8	SW_MUX_CTL_PAD_UART3_RXD (IOMUXC_SMUXC_PUART3_RXD)	32	R/W	0000_0001h	35.4.43/ 1730
53FA_80AC	SW_MUX_CTL_PAD_UART4_TXD (IOMUXC_SMUXC_PUART4_TXD)	32	R/W	0000_0001h	35.4.44/ 1731
53FA_80B0	SW_MUX_CTL_PAD_UART4_RXD (IOMUXC_SMUXC_PUART4_RXD)	32	R/W	0000_0001h	35.4.45/ 1732
53FA_80B4	SW_MUX_CTL_PAD_CSPI_SCLK (IOMUXC_SMUXC_PCSPI_SCLK)	32	R/W	0000_0001h	35.4.46/ 1733
53FA_80B8	SW_MUX_CTL_PAD_CSPI_MOSI (IOMUXC_SMUXC_PCSPI_MOSI)	32	R/W	0000_0001h	35.4.47/ 1734
53FA_80BC	SW_MUX_CTL_PAD_CSPI_MISO (IOMUXC_SMUXC_PCSPI_MISO)	32	R/W	0000_0001h	35.4.48/ 1735
53FA_80C0	SW_MUX_CTL_PAD_CSPI_SS0 (IOMUXC_SMUXC_PCSPI_SS0)	32	R/W	0000_0001h	35.4.49/ 1736
53FA_80C4	SW_MUX_CTL_PAD_ECSPI1_SCLK (IOMUXC_SMUXC_PECSPI1_SCLK)	32	R/W	0000_0001h	35.4.50/ 1737
53FA_80C8	SW_MUX_CTL_PAD_ECSPI1_MOSI (IOMUXC_SMUXC_PECSPI1_MOSI)	32	R/W	0000_0001h	35.4.51/ 1738
53FA_80CC	SW_MUX_CTL_PAD_ECSPI1_MISO (IOMUXC_SMUXC_PECSPI1_MISO)	32	R/W	0000_0001h	35.4.52/ 1739
53FA_80D0	SW_MUX_CTL_PAD_ECSPI1_SS0 (IOMUXC_SMUXC_PECSPI1_SS0)	32	R/W	0000_0001h	35.4.53/ 1740
53FA_80D4	SW_MUX_CTL_PAD_ECSPI2_SCLK (IOMUXC_SMUXC_PECSPI2_SCLK)	32	R/W	0000_0001h	35.4.54/ 1741
53FA_80D8	SW_MUX_CTL_PAD_ECSPI2_MOSI (IOMUXC_SMUXC_PECSPI2_MOSI)	32	R/W	0000_0001h	35.4.55/ 1742

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_80DC	SW_MUX_CTL_PAD_ECSPI2_MISO (IOMUXC_SMUXC_PECSPI2_MISO)	32	R/W	0000_0001h	35.4.56/ 1743
53FA_80E0	SW_MUX_CTL_PAD_ECSPI2_SS0 (IOMUXC_SMUXC_PECSPI2_SS0)	32	R/W	0000_0001h	35.4.57/ 1745
53FA_80E4	SW_MUX_CTL_PAD_SD1_CLK (IOMUXC_SMUXC_PSD1_CLK)	32	R/W	0000_0001h	35.4.58/ 1746
53FA_80E8	SW_MUX_CTL_PAD_SD1_CMD (IOMUXC_SMUXC_PSD1_CMD)	32	R/W	0000_0001h	35.4.59/ 1747
53FA_80EC	SW_MUX_CTL_PAD_SD1_D0 (IOMUXC_SMUXC_PSD1_D0)	32	R/W	0000_0001h	35.4.60/ 1748
53FA_80F0	SW_MUX_CTL_PAD_SD1_D1 (IOMUXC_SMUXC_PSD1_D1)	32	R/W	0000_0001h	35.4.61/ 1749
53FA_80F4	SW_MUX_CTL_PAD_SD1_D2 (IOMUXC_SMUXC_PSD1_D2)	32	R/W	0000_0001h	35.4.62/ 1750
53FA_80F8	SW_MUX_CTL_PAD_SD1_D3 (IOMUXC_SMUXC_PSD1_D3)	32	R/W	0000_0001h	35.4.63/ 1751
53FA_80FC	SW_MUX_CTL_PAD_SD2_CLK (IOMUXC_SMUXC_PSD2_CLK)	32	R/W	0000_0001h	35.4.64/ 1752
53FA_8100	SW_MUX_CTL_PAD_SD2_CMD (IOMUXC_SMUXC_PSD2_CMD)	32	R/W	0000_0001h	35.4.65/ 1753
53FA_8104	SW_MUX_CTL_PAD_SD2_D0 (IOMUXC_SMUXC_PSD2_D0)	32	R/W	0000_0001h	35.4.66/ 1754
53FA_8108	SW_MUX_CTL_PAD_SD2_D1 (IOMUXC_SMUXC_PSD2_D1)	32	R/W	0000_0001h	35.4.67/ 1755
53FA_810C	SW_MUX_CTL_PAD_SD2_D2 (IOMUXC_SMUXC_PSD2_D2)	32	R/W	0000_0001h	35.4.68/ 1756
53FA_8110	SW_MUX_CTL_PAD_SD2_D3 (IOMUXC_SMUXC_PSD2_D3)	32	R/W	0000_0001h	35.4.69/ 1757
53FA_8114	SW_MUX_CTL_PAD_SD2_D4 (IOMUXC_SMUXC_PSD2_D4)	32	R/W	0000_0001h	35.4.70/ 1758
53FA_8118	SW_MUX_CTL_PAD_SD2_D5 (IOMUXC_SMUXC_PSD2_D5)	32	R/W	0000_0001h	35.4.71/ 1759
53FA_811C	SW_MUX_CTL_PAD_SD2_D6 (IOMUXC_SMUXC_PSD2_D6)	32	R/W	0000_0001h	35.4.72/ 1760
53FA_8120	SW_MUX_CTL_PAD_SD2_D7 (IOMUXC_SMUXC_PSD2_D7)	32	R/W	0000_0001h	35.4.73/ 1761
53FA_8124	SW_MUX_CTL_PAD_SD2_WP (IOMUXC_SMUXC_PSD2_WP)	32	R/W	0000_0001h	35.4.74/ 1762
53FA_8128	SW_MUX_CTL_PAD_SD2_CD (IOMUXC_SMUXC_PSD2_CD)	32	R/W	0000_0001h	35.4.75/ 1763

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_812C	SW_MUX_CTL_PAD_DISP_D0 (IOMUXC_SMUXC_PDISP_D0)	32	R/W	0000_0003h	35.4.76/ 1764
53FA_8130	SW_MUX_CTL_PAD_DISP_D1 (IOMUXC_SMUXC_PDISP_D1)	32	R/W	0000_0003h	35.4.77/ 1765
53FA_8134	SW_MUX_CTL_PAD_DISP_D2 (IOMUXC_SMUXC_PDISP_D2)	32	R/W	0000_0003h	35.4.78/ 1766
53FA_8138	SW_MUX_CTL_PAD_DISP_D3 (IOMUXC_SMUXC_PDISP_D3)	32	R/W	0000_0003h	35.4.79/ 1767
53FA_813C	SW_MUX_CTL_PAD_DISP_D4 (IOMUXC_SMUXC_PDISP_D4)	32	R/W	0000_0003h	35.4.80/ 1768
53FA_8140	SW_MUX_CTL_PAD_DISP_D5 (IOMUXC_SMUXC_PDISP_D5)	32	R/W	0000_0003h	35.4.81/ 1769
53FA_8144	SW_MUX_CTL_PAD_DISP_D6 (IOMUXC_SMUXC_PDISP_D6)	32	R/W	0000_0003h	35.4.82/ 1770
53FA_8148	SW_MUX_CTL_PAD_DISP_D7 (IOMUXC_SMUXC_PDISP_D7)	32	R/W	0000_0003h	35.4.83/ 1771
53FA_814C	SW_MUX_CTL_PAD_DISP_WR (IOMUXC_SMUXC_PDISP_WR)	32	R/W	0000_0003h	35.4.84/ 1772
53FA_8150	SW_MUX_CTL_PAD_DISP_RD (IOMUXC_SMUXC_PDISP_RD)	32	R/W	0000_0003h	35.4.85/ 1773
53FA_8154	SW_MUX_CTL_PAD_DISP_RS (IOMUXC_SMUXC_PDISP_RS)	32	R/W	0000_0003h	35.4.86/ 1774
53FA_8158	SW_MUX_CTL_PAD_DISP_CS (IOMUXC_SMUXC_PDISP_CS)	32	R/W	0000_0003h	35.4.87/ 1775
53FA_815C	SW_MUX_CTL_PAD_DISP_BUSY (IOMUXC_SMUXC_PDISP_BUSY)	32	R/W	0000_0001h	35.4.88/ 1776
53FA_8160	SW_MUX_CTL_PAD_DISP_RESET (IOMUXC_SMUXC_PDISP_RESET)	32	R/W	0000_0001h	35.4.89/ 1777
53FA_8164	SW_MUX_CTL_PAD_SD3_CMD (IOMUXC_SMUXC_PSD3_CMD)	32	R/W	0000_0001h	35.4.90/ 1778
53FA_8168	SW_MUX_CTL_PAD_SD3_CLK (IOMUXC_SMUXC_PSD3_CLK)	32	R/W	0000_0001h	35.4.91/ 1779
53FA_816C	SW_MUX_CTL_PAD_SD3_D0 (IOMUXC_SMUXC_PSD3_D0)	32	R/W	0000_0001h	35.4.92/ 1780
53FA_8170	SW_MUX_CTL_PAD_SD3_D1 (IOMUXC_SMUXC_PSD3_D1)	32	R/W	0000_0001h	35.4.93/ 1781
53FA_8174	SW_MUX_CTL_PAD_SD3_D2 (IOMUXC_SMUXC_PSD3_D2)	32	R/W	0000_0001h	35.4.94/ 1782
53FA_8178	SW_MUX_CTL_PAD_SD3_D3 (IOMUXC_SMUXC_PSD3_D3)	32	R/W	0000_0001h	35.4.95/ 1783

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_817C	SW_MUX_CTL_PAD_SD3_D4 (IOMUXC_SMUXC_PSD3_D4)	32	R/W	0000_0001h	35.4.96/ 1784
53FA_8180	SW_MUX_CTL_PAD_SD3_D5 (IOMUXC_SMUXC_PSD3_D5)	32	R/W	0000_0001h	35.4.97/ 1785
53FA_8184	SW_MUX_CTL_PAD_SD3_D6 (IOMUXC_SMUXC_PSD3_D6)	32	R/W	0000_0001h	35.4.98/ 1786
53FA_8188	SW_MUX_CTL_PAD_SD3_D7 (IOMUXC_SMUXC_PSD3_D7)	32	R/W	0000_0001h	35.4.99/ 1787
53FA_818C	SW_MUX_CTL_PAD_SD3_WP (IOMUXC_SMUXC_PSD3_WP)	32	R/W	0000_0001h	35.4.100/ 1788
53FA_8190	SW_MUX_CTL_PAD_DISP_D8 (IOMUXC_SMUXC_PDISP_D8)	32	R/W	0000_0001h	35.4.101/ 1789
53FA_8194	SW_MUX_CTL_PAD_DISP_D9 (IOMUXC_SMUXC_PDISP_D9)	32	R/W	0000_0001h	35.4.102/ 1790
53FA_8198	SW_MUX_CTL_PAD_DISP_D10 (IOMUXC_SMUXC_PDISP_D10)	32	R/W	0000_0001h	35.4.103/ 1791
53FA_819C	SW_MUX_CTL_PAD_DISP_D11 (IOMUXC_SMUXC_PDISP_D11)	32	R/W	0000_0001h	35.4.104/ 1792
53FA_81A0	SW_MUX_CTL_PAD_DISP_D12 (IOMUXC_SMUXC_PDISP_D12)	32	R/W	0000_0001h	35.4.105/ 1794
53FA_81A4	SW_MUX_CTL_PAD_DISP_D13 (IOMUXC_SMUXC_PDISP_D13)	32	R/W	0000_0001h	35.4.106/ 1795
53FA_81A8	SW_MUX_CTL_PAD_DISP_D14 (IOMUXC_SMUXC_PDISP_D14)	32	R/W	0000_0001h	35.4.107/ 1796
53FA_81AC	SW_MUX_CTL_PAD_DISP_D15 (IOMUXC_SMUXC_PDISP_D15)	32	R/W	0000_0001h	35.4.108/ 1797
53FA_81B0	SW_MUX_CTL_PAD_EPDC_D0 (IOMUXC_SMUXC_PEPDC_D0)	32	R/W	0000_0001h	35.4.109/ 1799
53FA_81B4	SW_MUX_CTL_PAD_EPDC_D1 (IOMUXC_SMUXC_PEPDC_D1)	32	R/W	0000_0001h	35.4.110/ 1800
53FA_81B8	SW_MUX_CTL_PAD_EPDC_D2 (IOMUXC_SMUXC_PEPDC_D2)	32	R/W	0000_0001h	35.4.111/ 1801
53FA_81BC	SW_MUX_CTL_PAD_EPDC_D3 (IOMUXC_SMUXC_PEPDC_D3)	32	R/W	0000_0001h	35.4.112/ 1802
53FA_81C0	SW_MUX_CTL_PAD_EPDC_D4 (IOMUXC_SMUXC_PEPDC_D4)	32	R/W	0000_0001h	35.4.113/ 1803
53FA_81C4	SW_MUX_CTL_PAD_EPDC_D5 (IOMUXC_SMUXC_PEPDC_D5)	32	R/W	0000_0001h	35.4.114/ 1804
53FA_81C8	SW_MUX_CTL_PAD_EPDC_D6 (IOMUXC_SMUXC_PEPDC_D6)	32	R/W	0000_0001h	35.4.115/ 1805

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_81CC	SW_MUX_CTL_PAD_EPDC_D7 (IOMUXC_SMUXC_PEPDC_D7)	32	R/W	0000_0001h	35.4.116/ 1806
53FA_81D0	SW_MUX_CTL_PAD_EPDC_D8 (IOMUXC_SMUXC_PEPDC_D8)	32	R/W	0000_0001h	35.4.117/ 1807
53FA_81D4	SW_MUX_CTL_PAD_EPDC_D9 (IOMUXC_SMUXC_PEPDC_D9)	32	R/W	0000_0001h	35.4.118/ 1808
53FA_81D8	SW_MUX_CTL_PAD_EPDC_D10 (IOMUXC_SMUXC_PEPDC_D10)	32	R/W	0000_0001h	35.4.119/ 1809
53FA_81DC	SW_MUX_CTL_PAD_EPDC_D11 (IOMUXC_SMUXC_PEPDC_D11)	32	R/W	0000_0001h	35.4.120/ 1810
53FA_81E0	SW_MUX_CTL_PAD_EPDC_D12 (IOMUXC_SMUXC_PEPDC_D12)	32	R/W	0000_0001h	35.4.121/ 1811
53FA_81E4	SW_MUX_CTL_PAD_EPDC_D13 (IOMUXC_SMUXC_PEPDC_D13)	32	R/W	0000_0001h	35.4.122/ 1812
53FA_81E8	SW_MUX_CTL_PAD_EPDC_D14 (IOMUXC_SMUXC_PEPDC_D14)	32	R/W	0000_0001h	35.4.123/ 1813
53FA_81EC	SW_MUX_CTL_PAD_EPDC_D15 (IOMUXC_SMUXC_PEPDC_D15)	32	R/W	0000_0001h	35.4.124/ 1814
53FA_81F0	SW_MUX_CTL_PAD_EPDC_GDCLK (IOMUXC_SMUXC_PEPDC_GDCLK)	32	R/W	0000_0001h	35.4.125/ 1815
53FA_81F4	SW_MUX_CTL_PAD_EPDC_GDSP (IOMUXC_SMUXC_PEPDC_GDSP)	32	R/W	0000_0001h	35.4.126/ 1816
53FA_81F8	SW_MUX_CTL_PAD_EPDC_GDOE (IOMUXC_SMUXC_PEPDC_GDOE)	32	R/W	0000_0001h	35.4.127/ 1817
53FA_81FC	SW_MUX_CTL_PAD_EPDC_GDRL (IOMUXC_SMUXC_PEPDC_GDRL)	32	R/W	0000_0001h	35.4.128/ 1818
53FA_8200	SW_MUX_CTL_PAD_EPDC_SDCLK (IOMUXC_SMUXC_PEPDC_SDCLK)	32	R/W	0000_0001h	35.4.129/ 1819
53FA_8204	SW_MUX_CTL_PAD_EPDC_SDOEZ (IOMUXC_SMUXC_PEPDC_SDOEZ)	32	R/W	0000_0001h	35.4.130/ 1820
53FA_8208	SW_MUX_CTL_PAD_EPDC_SDOED (IOMUXC_SMUXC_PEPDC_SDOED)	32	R/W	0000_0001h	35.4.131/ 1821
53FA_820C	OMUXC_SW_MUX_CTL_PAD_EPDC_SDOE (IOMUXC_OMUXC_SMUXC_PEPDC_SDOE)	32	R/W	0000_0001h	35.4.132/ 1822
53FA_8210	SW_MUX_CTL_PAD_EPDC_SDLE (IOMUXC_SMUXC_PEPDC_SDLE)	32	R/W	0000_0001h	35.4.133/ 1823
53FA_8214	SW_MUX_CTL_PAD_EPDC_SDCLKN (IOMUXC_SMUXC_PEPDC_SDCLKN)	32	R/W	0000_0001h	35.4.134/ 1824
53FA_8218	SW_MUX_CTL_PAD_EPDC_SDSHR (IOMUXC_SMUXC_PEPDC_SDSHR)	32	R/W	0000_0001h	35.4.135/ 1825

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_821C	SW_MUX_CTL_PAD_EPDC_PWRCOM (IOMUXC_SMUXC_PEPDC_PWRCOM)	32	R/W	0000_0001h	35.4.136/ 1826
53FA_8220	SW_MUX_CTL_PAD_EPDC_PWRSTAT (IOMUXC_SMUXC_PEPDC_PWRSTAT)	32	R/W	0000_0001h	35.4.137/ 1827
53FA_8224	SW_MUX_CTL_PAD_EPDC_PWRCTRL0 (IOMUXC_SMUXC_PEPDC_PWRCTRL0)	32	R/W	0000_0001h	35.4.138/ 1828
53FA_8228	OMUXC_SW_MUX_CTL_PAD_EPDC_PWRCTRL1 (IOMUXC_OMUXC_SMUXC_PEPDC_PWRCTRL1)	32	R/W	0000_0001h	35.4.139/ 1829
53FA_822C	SW_MUX_CTL_PAD_EPDC_PWRCTRL2 (IOMUXC_SMUXC_PEPDC_PWRCTRL2)	32	R/W	0000_0001h	35.4.140/ 1830
53FA_8230	SW_MUX_CTL_PAD_EPDC_PWRCTRL3 (IOMUXC_SMUXC_PEPDC_PWRCTRL3)	32	R/W	0000_0001h	35.4.141/ 1831
53FA_8234	SW_MUX_CTL_PAD_EPDC_VCOM0 (IOMUXC_SMUXC_PEPDC_VCOM0)	32	R/W	0000_0001h	35.4.142/ 1832
53FA_8238	SW_MUX_CTL_PAD_EPDC_VCOM1 (IOMUXC_SMUXC_PEPDC_VCOM1)	32	R/W	0000_0001h	35.4.143/ 1833
53FA_823C	SW_MUX_CTL_PAD_EPDC_BDR0 (IOMUXC_SMUXC_PEPDC_BDR0)	32	R/W	0000_0001h	35.4.144/ 1834
53FA_8240	SW_MUX_CTL_PAD_EPDC_BDR1 (IOMUXC_SMUXC_PEPDC_BDR1)	32	R/W	0000_0001h	35.4.145/ 1835
53FA_8244	SW_MUX_CTL_PAD_EPDC_SDCE0 (IOMUXC_SMUXC_PEPDC_SDCE0)	32	R/W	0000_0001h	35.4.146/ 1836
53FA_8248	SW_MUX_CTL_PAD_EPDC_SDCE1 (IOMUXC_SMUXC_PEPDC_SDCE1)	32	R/W	0000_0001h	35.4.147/ 1837
53FA_824C	SW_MUX_CTL_PAD_EPDC_SDCE2 (IOMUXC_SMUXC_PEPDC_SDCE2)	32	R/W	0000_0001h	35.4.148/ 1838
53FA_8250	SW_MUX_CTL_PAD_EPDC_SDCE3 (IOMUXC_SMUXC_PEPDC_SDCE3)	32	R/W	0000_0001h	35.4.149/ 1839
53FA_8254	SW_MUX_CTL_PAD_EPDC_SDCE4 (IOMUXC_SMUXC_PEPDC_SDCE4)	32	R/W	0000_0001h	35.4.150/ 1840
53FA_8258	SW_MUX_CTL_PAD_EPDC_SDCE5 (IOMUXC_SMUXC_PEPDC_SDCE5)	32	R/W	0000_0001h	35.4.151/ 1841
53FA_825C	SW_MUX_CTL_PAD_EIM_DA0 (IOMUXC_SMUXC_PEIM_DA0)	32	R/W	0000_0000h	35.4.152/ 1842
53FA_8260	SW_MUX_CTL_PAD_EIM_DA1 (IOMUXC_SMUXC_PEIM_DA1)	32	R/W	0000_0000h	35.4.153/ 1843
53FA_8264	SW_MUX_CTL_PAD_EIM_DA2 (IOMUXC_SMUXC_PEIM_DA2)	32	R/W	0000_0000h	35.4.154/ 1844
53FA_8268	SW_MUX_CTL_PAD_EIM_DA3 (IOMUXC_SMUXC_PEIM_DA3)	32	R/W	0000_0000h	35.4.155/ 1845

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_826C	SW_MUX_CTL_PAD_EIM_DA4 (IOMUXC_SMUXC_PEIM_DA4)	32	R/W	0000_0000h	35.4.156/ 1846
53FA_8270	SW_MUX_CTL_PAD_EIM_DA5 (IOMUXC_SMUXC_PEIM_DA5)	32	R/W	0000_0000h	35.4.157/ 1847
53FA_8274	SW_MUX_CTL_PAD_EIM_DA6 (IOMUXC_SMUXC_PEIM_DA6)	32	R/W	0000_0000h	35.4.158/ 1848
53FA_8278	SW_MUX_CTL_PAD_EIM_DA7 (IOMUXC_SMUXC_PEIM_DA7)	32	R/W	0000_0000h	35.4.159/ 1849
53FA_827C	SW_MUX_CTL_PAD_EIM_DA8 (IOMUXC_SMUXC_PEIM_DA8)	32	R/W	0000_0000h	35.4.160/ 1850
53FA_8280	SW_MUX_CTL_PAD_EIM_DA9 (IOMUXC_SMUXC_PEIM_DA9)	32	R/W	0000_0000h	35.4.161/ 1851
53FA_8284	SW_MUX_CTL_PAD_EIM_DA10 (IOMUXC_SMUXC_PEIM_DA10)	32	R/W	0000_0000h	35.4.162/ 1852
53FA_8288	SW_MUX_CTL_PAD_EIM_DA11 (IOMUXC_SMUXC_PEIM_DA11)	32	R/W	0000_0000h	35.4.163/ 1853
53FA_828C	SW_MUX_CTL_PAD_EIM_DA12 (IOMUXC_SMUXC_PEIM_DA12)	32	R/W	0000_0000h	35.4.164/ 1854
53FA_8290	SW_MUX_CTL_PAD_EIM_DA13 (IOMUXC_SMUXC_PEIM_DA13)	32	R/W	0000_0000h	35.4.165/ 1855
53FA_8294	SW_MUX_CTL_PAD_EIM_DA14 (IOMUXC_SMUXC_PEIM_DA14)	32	R/W	0000_0000h	35.4.166/ 1856
53FA_8298	SW_MUX_CTL_PAD_EIM_DA15 (IOMUXC_SMUXC_PEIM_DA15)	32	R/W	0000_0000h	35.4.167/ 1857
53FA_829C	SW_MUX_CTL_PAD_EIM_CS2 (IOMUXC_SMUXC_PEIM_CS2)	32	R/W	0000_0000h	35.4.168/ 1858
53FA_82A0	SW_MUX_CTL_PAD_EIM_CS1 (IOMUXC_SMUXC_PEIM_CS1)	32	R/W	0000_0000h	35.4.169/ 1859
53FA_82A4	SW_MUX_CTL_PAD_EIM_CS0 (IOMUXC_SMUXC_PEIM_CS0)	32	R/W	0000_0000h	35.4.170/ 1860
53FA_82A8	SW_MUX_CTL_PAD_EIM_EB0 (IOMUXC_SMUXC_PEIM_EB0)	32	R/W	0000_0000h	35.4.171/ 1861
53FA_82AC	SW_MUX_CTL_PAD_EIM_EB1 (IOMUXC_SMUXC_PEIM_EB1)	32	R/W	0000_0000h	35.4.172/ 1862
53FA_82B0	SW_MUX_CTL_PAD_EIM_WAIT (IOMUXC_SMUXC_PEIM_WAIT)	32	R/W	0000_0000h	35.4.173/ 1863
53FA_82B4	SW_MUX_CTL_PAD_EIM_BCLK (IOMUXC_SMUXC_PEIM_BCLK)	32	R/W	0000_0000h	35.4.174/ 1864
53FA_82B8	SW_MUX_CTL_PAD_EIM_RDY (IOMUXC_SMUXC_PEIM_RDY)	32	R/W	0000_0000h	35.4.175/ 1865

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_82BC	SW_MUX_CTL_PAD_EIM_OE (IOMUXC_SMUXC_PEIM_OE)	32	R/W	0000_0000h	35.4.176/ 1866
53FA_82C0	SW_MUX_CTL_PAD_EIM_RW (IOMUXC_SMUXC_PEIM_RW)	32	R/W	0000_0000h	35.4.177/ 1867
53FA_82C4	SW_MUX_CTL_PAD_EIM_LBA (IOMUXC_SMUXC_PEIM_LBA)	32	R/W	0000_0000h	35.4.178/ 1868
53FA_82C8	SW_MUX_CTL_PAD_EIM_CRE (IOMUXC_SMUXC_PEIM_CRE)	32	R/W	0000_0000h	35.4.179/ 1869
53FA_82CC	SW_PAD_CTL_PAD_KEY_COL0 (IOMUXC_SPADC_PKC0)	32	R/W	0000_0084h	35.4.180/ 1870
53FA_82D0	SW_PAD_CTL_PAD_KEY_ROW0 (IOMUXC_SPADC_PKR0)	32	R/W	0000_0084h	35.4.181/ 1871
53FA_82D4	SW_PAD_CTL_PAD_KEY_COL1 (IOMUXC_SPADC_PKC1)	32	R/W	0000_0084h	35.4.182/ 1873
53FA_82D8	SW_PAD_CTL_PAD_KEY_ROW1 (IOMUXC_SPADC_PKR1)	32	R/W	0000_0084h	35.4.183/ 1874
53FA_82DC	SW_PAD_CTL_PAD_KEY_COL2 (IOMUXC_SPADC_PKC2)	32	R/W	0000_0084h	35.4.184/ 1876
53FA_82E0	SW_PAD_CTL_PAD_KEY_ROW2 (IOMUXC_SPADC_PKR2)	32	R/W	0000_0084h	35.4.185/ 1877
53FA_82E4	SW_PAD_CTL_PAD_KEY_COL3 (IOMUXC_SPADC_PKC3)	32	R/W	0000_0084h	35.4.186/ 1879
53FA_82E8	SW_PAD_CTL_PAD_KEY_ROW3 (IOMUXC_SPADC_PKR3)	32	R/W	0000_0084h	35.4.187/ 1880
53FA_82EC	SW_PAD_CTL_PAD_I2C1_SCL (IOMUXC_SPADC_PI2C1_SCL)	32	R/W	0000_0184h	35.4.188/ 1882
53FA_82F0	SW_PAD_CTL_PAD_I2C1_SDA (IOMUXC_SPADC_PI2C1_SDA)	32	R/W	0000_0184h	35.4.189/ 1883
53FA_82F4	SW_PAD_CTL_PAD_I2C2_SCL (IOMUXC_SPADC_PI2C2_SCL)	32	R/W	0000_0184h	35.4.190/ 1885
53FA_82F8	SW_PAD_CTL_PAD_I2C2_SDA (IOMUXC_SPADC_PI2C2_SDA)	32	R/W	0000_0184h	35.4.191/ 1886
53FA_82FC	SW_PAD_CTL_PAD_I2C3_SCL (IOMUXC_SPADC_PI2C3_SCL)	32	R/W	0000_0184h	35.4.192/ 1888
53FA_8300	SW_PAD_CTL_PAD_I2C3_SDA (IOMUXC_SPADC_PI2C3_SDA)	32	R/W	0000_0184h	35.4.193/ 1889
53FA_8304	SW_PAD_CTL_PAD_PWM1 (IOMUXC_SPADC_PPWM1)	32	R/W	0000_0084h	35.4.194/ 1891
53FA_8308	SW_PAD_CTL_PAD_PWM2 (IOMUXC_SPADC_PPWM2)	32	R/W	0000_0084h	35.4.195/ 1892

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_830C	SW_PAD_CTL_PAD_OWIRE (IOMUXC_SPADC_POWIRE)	32	R/W	0000_0084h	35.4.196/ 1894
53FA_8310	SW_PAD_CTL_PAD_EPITO (IOMUXC_SPADC_PEPITO)	32	R/W	0000_0084h	35.4.197/ 1895
53FA_8314	SW_PAD_CTL_PAD_WDOG (IOMUXC_SPADC_PWDOG)	32	R/W	0000_0084h	35.4.198/ 1897
53FA_8318	SW_PAD_CTL_PAD_SSI_TXFS (IOMUXC_SPADC_PSSI_TXFS)	32	R/W	0000_0084h	35.4.199/ 1898
53FA_831C	SW_PAD_CTL_PAD_SSI_TXC (IOMUXC_SPADC_PSSI_TXC)	32	R/W	0000_0084h	35.4.200/ 1900
53FA_8320	SW_PAD_CTL_PAD_SSI_TXD (IOMUXC_SPADC_PSSI_TXD)	32	R/W	0000_0084h	35.4.201/ 1901
53FA_8324	SW_PAD_CTL_PAD_SSI_RXD (IOMUXC_SPADC_PSSI_RXD)	32	R/W	0000_0084h	35.4.202/ 1903
53FA_8328	SW_PAD_CTL_PAD_SSI_RXFS (IOMUXC_SPADC_PSSI_RXFS)	32	R/W	0000_0084h	35.4.203/ 1904
53FA_832C	SW_PAD_CTL_PAD_SSI_RXC (IOMUXC_SPADC_PSSI_RXC)	32	R/W	0000_0084h	35.4.204/ 1906
53FA_8330	SW_PAD_CTL_PAD_UART1_TXD (IOMUXC_SPADC_PUART1_TXD)	32	R/W	0000_0084h	35.4.205/ 1907
53FA_8334	SW_PAD_CTL_PAD_UART1_RXD (IOMUXC_SPADC_PUART1_RXD)	32	R/W	0000_0084h	35.4.206/ 1909
53FA_8338	SW_PAD_CTL_PAD_UART1_CTS (IOMUXC_SPADC_PUART1_CTS)	32	R/W	0000_0084h	35.4.207/ 1910
53FA_833C	SW_PAD_CTL_PAD_UART1_RTS (IOMUXC_SPADC_PUART1_RTS)	32	R/W	0000_0084h	35.4.208/ 1912
53FA_8340	SW_PAD_CTL_PAD_UART2_TXD (IOMUXC_SPADC_PUART2_TXD)	32	R/W	0000_0084h	35.4.209/ 1913
53FA_8344	SW_PAD_CTL_PAD_UART2_RXD (IOMUXC_SPADC_PUART2_RXD)	32	R/W	0000_0084h	35.4.210/ 1915
53FA_8348	SW_PAD_CTL_PAD_UART2_CTS (IOMUXC_SPADC_PUART2_CTS)	32	R/W	0000_0084h	35.4.211/ 1916
53FA_834C	SW_PAD_CTL_PAD_UART2_RTS (IOMUXC_SPADC_PUART2_RTS)	32	R/W	0000_0084h	35.4.212/ 1918
53FA_8350	SW_PAD_CTL_PAD_UART3_TXD (IOMUXC_SPADC_PUART3_TXD)	32	R/W	0000_0084h	35.4.213/ 1919
53FA_8354	SW_PAD_CTL_PAD_UART3_RXD (IOMUXC_SPADC_PUART3_RXD)	32	R/W	0000_0084h	35.4.214/ 1921
53FA_8358	SW_PAD_CTL_PAD_UART4_TXD (IOMUXC_SPADC_PUART4_TXD)	32	R/W	0000_0084h	35.4.215/ 1922

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_835C	SW_PAD_CTL_PAD_UART4_RXD (IOMUXC_SPADC_PUART4_RXD)	32	R/W	0000_0084h	35.4.216/ 1924
53FA_8360	SW_PAD_CTL_PAD_CSPI_SCLK (IOMUXC_SPADC_PCSPI_SCLK)	32	R/W	0000_0084h	35.4.217/ 1925
53FA_8364	SW_PAD_CTL_PAD_CSPI_MOSI (IOMUXC_SPADC_PCSPI_MOSI)	32	R/W	0000_0084h	35.4.218/ 1927
53FA_8368	SW_PAD_CTL_PAD_CSPI_MISO (IOMUXC_SPADC_PCSPI_MISO)	32	R/W	0000_0084h	35.4.219/ 1928
53FA_836C	SW_PAD_CTL_PAD_CSPI_SS0 (IOMUXC_SPADC_PCSPI_SS0)	32	R/W	0000_0084h	35.4.220/ 1930
53FA_8370	SW_PAD_CTL_PAD_ECSPI1_SCLK (IOMUXC_SPADC_PECSPI1_SCLK)	32	R/W	0000_0084h	35.4.221/ 1931
53FA_8374	SW_PAD_CTL_PAD_ECSPI1_MOSI (IOMUXC_SPADC_PECSPI1_MOSI)	32	R/W	0000_0084h	35.4.222/ 1933
53FA_8378	SW_PAD_CTL_PAD_ECSPI1_MISO (IOMUXC_SPADC_PECSPI1_MISO)	32	R/W	0000_0084h	35.4.223/ 1934
53FA_837C	SW_PAD_CTL_PAD_ECSPI1_SS0 (IOMUXC_SPADC_PECSPI1_SS0)	32	R/W	0000_0084h	35.4.224/ 1936
53FA_8380	SW_PAD_CTL_PAD_ECSPI2_SCLK (IOMUXC_SPADC_PECSPI2_SCLK)	32	R/W	0000_0084h	35.4.225/ 1937
53FA_8384	SW_PAD_CTL_PAD_ECSPI2_MOSI (IOMUXC_SPADC_PECSPI2_MOSI)	32	R/W	0000_0084h	35.4.226/ 1939
53FA_8388	SW_PAD_CTL_PAD_ECSPI2_MISO (IOMUXC_SPADC_PECSPI2_MISO)	32	R/W	0000_0084h	35.4.227/ 1940
53FA_838C	SW_PAD_CTL_PAD_ECSPI2_SS0 (IOMUXC_SPADC_PECSPI2_SS0)	32	R/W	0000_0084h	35.4.228/ 1942
53FA_8390	SW_PAD_CTL_PAD_SD1_CLK (IOMUXC_SPADC_PSD1_CLK)	32	R/W	0000_0084h	35.4.229/ 1943
53FA_8394	SW_PAD_CTL_PAD_SD1_CMD (IOMUXC_SPADC_PSD1_CMD)	32	R/W	0000_0084h	35.4.230/ 1945
53FA_8398	SW_PAD_CTL_PAD_SD1_D0 (IOMUXC_SPADC_PSD1_D0)	32	R/W	0000_0084h	35.4.231/ 1946
53FA_839C	SW_PAD_CTL_PAD_SD1_D1 (IOMUXC_SPADC_PSD1_D1)	32	R/W	0000_0084h	35.4.232/ 1948
53FA_83A0	SW_PAD_CTL_PAD_SD1_D2 (IOMUXC_SPADC_PSD1_D2)	32	R/W	0000_0084h	35.4.233/ 1949
53FA_83A4	SW_PAD_CTL_PAD_SD1_D3 (IOMUXC_SPADC_PSD1_D3)	32	R/W	0000_0084h	35.4.234/ 1951
53FA_83A8	SW_PAD_CTL_PAD_SD2_CLK (IOMUXC_SPADC_PSD2_CLK)	32	R/W	0000_0084h	35.4.235/ 1952

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_83AC	SW_PAD_CTL_PAD_SD2_CMD (IOMUXC_SPADC_PSD2_CMD)	32	R/W	0000_0084h	35.4.236/ 1954
53FA_83B0	SW_PAD_CTL_PAD_SD2_D0 (IOMUXC_SPADC_PSD2_D0)	32	R/W	0000_0084h	35.4.237/ 1955
53FA_83B4	SW_PAD_CTL_PAD_SD2_D1 (IOMUXC_SPADC_PSD2_D1)	32	R/W	0000_0084h	35.4.238/ 1957
53FA_83B8	SW_PAD_CTL_PAD_SD2_D2 (IOMUXC_SPADC_PSD2_D2)	32	R/W	0000_0084h	35.4.239/ 1958
53FA_83BC	SW_PAD_CTL_PAD_SD2_D3 (IOMUXC_SPADC_PSD2_D3)	32	R/W	0000_0084h	35.4.240/ 1960
53FA_83C0	SW_PAD_CTL_PAD_SD2_D4 (IOMUXC_SPADC_PSD2_D4)	32	R/W	0000_0084h	35.4.241/ 1961
53FA_83C4	SW_PAD_CTL_PAD_SD2_D5 (IOMUXC_SPADC_PSD2_D5)	32	R/W	0000_0084h	35.4.242/ 1963
53FA_83C8	SW_PAD_CTL_PAD_SD2_D6 (IOMUXC_SPADC_PSD2_D6)	32	R/W	0000_0084h	35.4.243/ 1964
53FA_83CC	SW_PAD_CTL_PAD_SD2_D7 (IOMUXC_SPADC_PSD2_D7)	32	R/W	0000_0084h	35.4.244/ 1966
53FA_83D0	SW_PAD_CTL_PAD_SD2_WP (IOMUXC_SPADC_PSD2_WP)	32	R/W	0000_0084h	35.4.245/ 1967
53FA_83D4	SW_PAD_CTL_PAD_SD2_CD (IOMUXC_SPADC_PSD2_CD)	32	R/W	0000_0084h	35.4.246/ 1969
53FA_83D8	SW_PAD_CTL_PAD_PMIC_ON_REQ (IOMUXC_SPADC_PPMIC_ON_REQ)	32	R/W	0000_0405h	35.4.247/ 1970
53FA_83DC	SW_PAD_CTL_PAD_PMIC_STBY_REQ (IOMUXC_SPADC_PPMIC_STBY_REQ)	32	R/W	0000_0405h	35.4.248/ 1971
53FA_83E0	SW_PAD_CTL_PAD_POR_B (IOMUXC_SPADC_PPOR_B)	32	R/W	0000_05E0h	35.4.249/ 1972
53FA_83E4	SW_PAD_CTL_PAD_BOOT_MODE1 (IOMUXC_SPADC_PBOOT_MODE1)	32	R/W	0000_05E0h	35.4.250/ 1972
53FA_83E8	SW_PAD_CTL_PAD_RESET_IN_B (IOMUXC_SPADC_PRESET_IN_B)	32	R/W	0000_05E0h	35.4.251/ 1973
53FA_83EC	SW_PAD_CTL_PAD_BOOT_MODE0 (IOMUXC_SPADC_PBOOT_MODE0)	32	R/W	0000_05E0h	35.4.252/ 1974
53FA_83F0	SW_PAD_CTL_PAD_TEST_MODE (IOMUXC_SPADC_PTEST_MODE)	32	R/W	0000_04C0h	35.4.253/ 1974
53FA_83F4	SW_PAD_CTL_PAD_JTAG_TMS (IOMUXC_SPADC_PJTAG_TMS)	32	R/W	0000_05D0h	35.4.254/ 1975
53FA_83F8	SW_PAD_CTL_PAD_JTAG_MOD (IOMUXC_SPADC_PJTAG_MOD)	32	R/W	0000_04E0h	35.4.255/ 1976

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_83FC	SW_PAD_CTL_PAD_JTAG_TRSTB (IOMUXC_SPADC_PJTAG_TRSTB)	32	R/W	0000_04D0h	35.4.256/ 1976
53FA_8400	SW_PAD_CTL_PAD_JTAG_TDI (IOMUXC_SPADC_PJTAG_TDI)	32	R/W	0000_05D0h	35.4.257/ 1977
53FA_8404	SW_PAD_CTL_PAD_JTAG_TCK (IOMUXC_SPADC_PJTAG_TCK)	32	R/W	0000_05C0h	35.4.258/ 1978
53FA_8408	SW_PAD_CTL_PAD_JTAG_TDO (IOMUXC_SPADC_PJTAG_TDO)	32	R/W	0000_0485h	35.4.259/ 1978
53FA_840C	SW_PAD_CTL_PAD_DISP_D0 (IOMUXC_SPADC_PDISP_D0)	32	R/W	0000_00E4h	35.4.260/ 1980
53FA_8410	SW_PAD_CTL_PAD_DISP_D1 (IOMUXC_SPADC_PDISP_D1)	32	R/W	0000_00E4h	35.4.261/ 1981
53FA_8414	SW_PAD_CTL_PAD_DISP_D2 (IOMUXC_SPADC_PDISP_D2)	32	R/W	0000_00E4h	35.4.262/ 1983
53FA_8418	SW_PAD_CTL_PAD_DISP_D3 (IOMUXC_SPADC_PDISP_D3)	32	R/W	0000_00E4h	35.4.263/ 1984
53FA_841C	SW_PAD_CTL_PAD_DISP_D4 (IOMUXC_SPADC_PDISP_D4)	32	R/W	0000_00E4h	35.4.264/ 1986
53FA_8420	SW_PAD_CTL_PAD_DISP_D5 (IOMUXC_SPADC_PDISP_D5)	32	R/W	0000_00E4h	35.4.265/ 1987
53FA_8424	SW_PAD_CTL_PAD_DISP_D6 (IOMUXC_SPADC_PDISP_D6)	32	R/W	0000_00E4h	35.4.266/ 1989
53FA_8428	SW_PAD_CTL_PAD_DISP_D7 (IOMUXC_SPADC_PDISP_D7)	32	R/W	0000_00E4h	35.4.267/ 1990
53FA_842C	SW_PAD_CTL_PAD_DISP_WR (IOMUXC_SPADC_PDISP_WR)	32	R/W	0000_00E4h	35.4.268/ 1992
53FA_8430	SW_PAD_CTL_PAD_DISP_RD (IOMUXC_SPADC_PDISP_RD)	32	R/W	0000_00E4h	35.4.269/ 1993
53FA_8434	SW_PAD_CTL_PAD_DISP_RS (IOMUXC_SPADC_PDISP_RS)	32	R/W	0000_00E4h	35.4.270/ 1995
53FA_8438	SW_PAD_CTL_PAD_DISP_CS (IOMUXC_SPADC_PDISP_CS)	32	R/W	0000_00E4h	35.4.271/ 1996
53FA_843C	SW_PAD_CTL_PAD_DISP_BUSY (IOMUXC_SPADC_PDISP_BUSY)	32	R/W	0000_0084h	35.4.272/ 1998
53FA_8440	SW_PAD_CTL_PAD_DISP_RESET (IOMUXC_SPADC_PDISP_RESET)	32	R/W	0000_0084h	35.4.273/ 1999
53FA_8444	SW_PAD_CTL_PAD_SD3_CMD (IOMUXC_SPADC_PSD3_CMD)	32	R/W	0000_0084h	35.4.274/ 2001
53FA_8448	SW_PAD_CTL_PAD_SD3_CLK (IOMUXC_SPADC_PSD3_CLK)	32	R/W	0000_0084h	35.4.275/ 2002

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_844C	SW_PAD_CTL_PAD_SD3_D0 (IOMUXC_SPADC_PSD3_D0)	32	R/W	0000_0084h	35.4.276/ 2004
53FA_8450	SW_PAD_CTL_PAD_SD3_D1 (IOMUXC_SPADC_PSD3_D1)	32	R/W	0000_0084h	35.4.277/ 2005
53FA_8454	SW_PAD_CTL_PAD_SD3_D2 (IOMUXC_SPADC_PSD3_D2)	32	R/W	0000_0084h	35.4.278/ 2007
53FA_8458	SW_PAD_CTL_PAD_SD3_D3 (IOMUXC_SPADC_PSD3_D3)	32	R/W	0000_0084h	35.4.279/ 2008
53FA_845C	SW_PAD_CTL_PAD_SD3_D4 (IOMUXC_SPADC_PSD3_D4)	32	R/W	0000_0084h	35.4.280/ 2010
53FA_8460	SW_PAD_CTL_PAD_SD3_D5 (IOMUXC_SPADC_PSD3_D5)	32	R/W	0000_0084h	35.4.281/ 2011
53FA_8464	SW_PAD_CTL_PAD_SD3_D6 (IOMUXC_SPADC_PSD3_D6)	32	R/W	0000_0084h	35.4.282/ 2013
53FA_8468	SW_PAD_CTL_PAD_SD3_D7 (IOMUXC_SPADC_PSD3_D7)	32	R/W	0000_0084h	35.4.283/ 2014
53FA_846C	SW_PAD_CTL_PAD_SD3_WP (IOMUXC_SPADC_PSD3_WP)	32	R/W	0000_0084h	35.4.284/ 2016
53FA_8470	SW_PAD_CTL_PAD_DISP_D8 (IOMUXC_SPADC_PDISP_D8)	32	R/W	0000_0084h	35.4.285/ 2017
53FA_8474	SW_PAD_CTL_PAD_DISP_D9 (IOMUXC_SPADC_PDISP_D9)	32	R/W	0000_0084h	35.4.286/ 2019
53FA_8478	SW_PAD_CTL_PAD_DISP_D10 (IOMUXC_SPADC_PDISP_D10)	32	R/W	0000_0084h	35.4.287/ 2020
53FA_847C	SW_PAD_CTL_PAD_DISP_D11 (IOMUXC_SPADC_PDISP_D11)	32	R/W	0000_0084h	35.4.288/ 2022
53FA_8480	SW_PAD_CTL_PAD_DISP_D12 (IOMUXC_SPADC_PDISP_D12)	32	R/W	0000_0084h	35.4.289/ 2023
53FA_8484	SW_PAD_CTL_PAD_DISP_D13 (IOMUXC_SPADC_PDISP_D13)	32	R/W	0000_0084h	35.4.290/ 2025
53FA_8488	SW_PAD_CTL_PAD_DISP_D14 (IOMUXC_SPADC_PDISP_D14)	32	R/W	0000_0084h	35.4.291/ 2026
53FA_848C	SW_PAD_CTL_PAD_DISP_D15 (IOMUXC_SPADC_PDISP_D15)	32	R/W	0000_0084h	35.4.292/ 2028
53FA_8490	SW_PAD_CTL_PAD_DRAM_OPEN (IOMUXC_SPADC_PDRAM_OPEN)	32	R/W	0008_0000h	35.4.293/ 2029
53FA_8494	SW_PAD_CTL_PAD_DRAM_OPENFB (IOMUXC_SPADC_PDRAM_OPENFB)	32	R/W	0008_0000h	35.4.294/ 2030
53FA_8498	SW_PAD_CTL_PAD_DRAM_SDCLK_1 (IOMUXC_SPADC_PDRAM_SDCLK_1)	32	R/W	0008_0200h	35.4.295/ 2030

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_849C	SW_PAD_CTL_PAD_DRAM_SDCLK_0 (IOMUXC_SPADC_PDRAM_SDCLK_0)	32	R/W	0008_0200h	35.4.296/ 2031
53FA_84A0	SW_PAD_CTL_PAD_DRAM_SDCKE (IOMUXC_SPADC_PDRAM_SDCKE)	32	R/W	0000_0080h	35.4.297/ 2032
53FA_84A4	SW_PAD_CTL_PAD_DRAM_SDODT0 (IOMUXC_SPADC_PDRAM_SDODT0)	32	R/W	0008_0080h	35.4.298/ 2033
53FA_84A8	SW_PAD_CTL_PAD_DRAM_D16 (IOMUXC_SPADC_PDRAM_D16)	32	R/W	0000_0000h	35.4.299/ 2034
53FA_84AC	SW_PAD_CTL_PAD_DRAM_D17 (IOMUXC_SPADC_PDRAM_D17)	32	R/W	0000_0000h	35.4.300/ 2035
53FA_84B0	SW_PAD_CTL_PAD_DRAM_D18 (IOMUXC_SPADC_PDRAM_D18)	32	R/W	0000_0000h	35.4.301/ 2035
53FA_84B4	SW_PAD_CTL_PAD_DRAM_D19 (IOMUXC_SPADC_PDRAM_D19)	32	R/W	0000_0000h	35.4.302/ 2036
53FA_84B8	SW_PAD_CTL_PAD_DRAM_D20 (IOMUXC_SPADC_PDRAM_D20)	32	R/W	0000_0000h	35.4.303/ 2036
53FA_84BC	SW_PAD_CTL_PAD_DRAM_D21 (IOMUXC_SPADC_PDRAM_D21)	32	R/W	0000_0000h	35.4.304/ 2037
53FA_84C0	SW_PAD_CTL_PAD_DRAM_D22 (IOMUXC_SPADC_PDRAM_D22)	32	R/W	0000_0000h	35.4.305/ 2038
53FA_84C4	SW_PAD_CTL_PAD_DRAM_D23 (IOMUXC_SPADC_PDRAM_D23)	32	R/W	0000_0000h	35.4.306/ 2038
53FA_84C8	SW_PAD_CTL_PAD_DRAM_DQM2 (IOMUXC_SPADC_PDRAM_DQM2)	32	R/W	0008_0000h	35.4.307/ 2039
53FA_84CC	SW_PAD_CTL_PAD_DRAM_SDQS2 (IOMUXC_SPADC_PDRAM_SDQS2)	32	R/W	0008_0000h	35.4.308/ 2040
53FA_84D0	SW_PAD_CTL_PAD_DRAM_D0 (IOMUXC_SPADC_PDRAM_D0)	32	R/W	0000_0000h	35.4.309/ 2041
53FA_84D4	SW_PAD_CTL_PAD_DRAM_D1 (IOMUXC_SPADC_PDRAM_D1)	32	R/W	0000_0000h	35.4.310/ 2041
53FA_84D8	SW_PAD_CTL_PAD_DRAM_D2 (IOMUXC_SPADC_PDRAM_D2)	32	R/W	0000_0000h	35.4.311/ 2042
53FA_84DC	SW_PAD_CTL_PAD_DRAM_D3 (IOMUXC_SPADC_PDRAM_D3)	32	R/W	0000_0000h	35.4.312/ 2042
53FA_84E0	SW_PAD_CTL_PAD_DRAM_D4 (IOMUXC_SPADC_PDRAM_D4)	32	R/W	0000_0000h	35.4.313/ 2043
53FA_84E4	SW_PAD_CTL_PAD_DRAM_D5 (IOMUXC_SPADC_PDRAM_D5)	32	R/W	0000_0000h	35.4.314/ 2044
53FA_84E8	SW_PAD_CTL_PAD_DRAM_D6 (IOMUXC_SPADC_PDRAM_D6)	32	R/W	0000_0000h	35.4.315/ 2044

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_84EC	SW_PAD_CTL_PAD_DRAM_D7 (IOMUXC_SPADC_PDRAM_D7)	32	R/W	0000_0000h	35.4.316/ 2045
53FA_84F0	SW_PAD_CTL_PAD_DRAM_DQM0 (IOMUXC_SPADC_PDRAM_DQM0)	32	R/W	0008_0000h	35.4.317/ 2045
53FA_84F4	SW_PAD_CTL_PAD_DRAM_SDQS0 (IOMUXC_SPADC_PDRAM_SDQS0)	32	R/W	0008_0000h	35.4.318/ 2046
53FA_84F8	SW_PAD_CTL_PAD_DRAM_SDODT1 (IOMUXC_SPADC_PDRAM_SDODT1)	32	R/W	0008_0080h	35.4.319/ 2047
53FA_84FC	SW_PAD_CTL_PAD_DRAM_SDQS1 (IOMUXC_SPADC_PDRAM_SDQS1)	32	R/W	0008_0000h	35.4.320/ 2048
53FA_8500	SW_PAD_CTL_PAD_DRAM_DQM1 (IOMUXC_SPADC_PDRAM_DQM1)	32	R/W	0008_0000h	35.4.321/ 2049
53FA_8504	SW_PAD_CTL_PAD_DRAM_D8 (IOMUXC_SPADC_PDRAM_D8)	32	R/W	0000_0000h	35.4.322/ 2050
53FA_8508	SW_PAD_CTL_PAD_DRAM_D9 (IOMUXC_SPADC_PDRAM_D9)	32	R/W	0000_0000h	35.4.323/ 2051
53FA_850C	SW_PAD_CTL_PAD_DRAM_D10 (IOMUXC_SPADC_PDRAM_D10)	32	R/W	0000_0000h	35.4.324/ 2051
53FA_8510	SW_PAD_CTL_PAD_DRAM_D11 (IOMUXC_SPADC_PDRAM_D11)	32	R/W	0000_0000h	35.4.325/ 2052
53FA_8514	SW_PAD_CTL_PAD_DRAM_D12 (IOMUXC_SPADC_PDRAM_D12)	32	R/W	0000_0000h	35.4.326/ 2053
53FA_8518	SW_PAD_CTL_PAD_DRAM_D13 (IOMUXC_SPADC_PDRAM_D13)	32	R/W	0000_0000h	35.4.327/ 2053
53FA_851C	SW_PAD_CTL_PAD_DRAM_D14 (IOMUXC_SPADC_PDRAM_D14)	32	R/W	0000_0000h	35.4.328/ 2054
53FA_8520	SW_PAD_CTL_PAD_DRAM_D15 (IOMUXC_SPADC_PDRAM_D15)	32	R/W	0000_0000h	35.4.329/ 2054
53FA_8524	SW_PAD_CTL_PAD_DRAM_SDQS3 (IOMUXC_SPADC_PDRAM_SDQS3)	32	R/W	0008_0000h	35.4.330/ 2055
53FA_8528	SW_PAD_CTL_PAD_DRAM_DQM3 (IOMUXC_SPADC_PDRAM_DQM3)	32	R/W	0008_0000h	35.4.331/ 2056
53FA_852C	SW_PAD_CTL_PAD_DRAM_D24 (IOMUXC_SPADC_PDRAM_D24)	32	R/W	0000_0000h	35.4.332/ 2057
53FA_8530	SW_PAD_CTL_PAD_DRAM_D25 (IOMUXC_SPADC_PDRAM_D25)	32	R/W	0000_0000h	35.4.333/ 2057
53FA_8534	SW_PAD_CTL_PAD_DRAM_D26 (IOMUXC_SPADC_PDRAM_D26)	32	R/W	0000_0000h	35.4.334/ 2058
53FA_8538	SW_PAD_CTL_PAD_DRAM_D27 (IOMUXC_SPADC_PDRAM_D27)	32	R/W	0000_0000h	35.4.335/ 2059

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_853C	SW_PAD_CTL_PAD_DRAM_D28 (IOMUXC_SPADC_PDRAM_D28)	32	R/W	0000_0000h	35.4.336/ 2059
53FA_8540	SW_PAD_CTL_PAD_DRAM_D29 (IOMUXC_SPADC_PDRAM_D29)	32	R/W	0000_0000h	35.4.337/ 2060
53FA_8544	SW_PAD_CTL_PAD_DRAM_D30 (IOMUXC_SPADC_PDRAM_D30)	32	R/W	0000_0000h	35.4.338/ 2060
53FA_8548	SW_PAD_CTL_PAD_DRAM_D31 (IOMUXC_SPADC_PDRAM_D31)	32	R/W	0000_0000h	35.4.339/ 2061
53FA_854C	SW_PAD_CTL_PAD_EPDC_D0 (IOMUXC_SPADC_PEPDC_D0)	32	R/W	0000_0084h	35.4.340/ 2062
53FA_8550	SW_PAD_CTL_PAD_EPDC_D1 (IOMUXC_SPADC_PEPDC_D1)	32	R/W	0000_0084h	35.4.341/ 2063
53FA_8554	SW_PAD_CTL_PAD_EPDC_D2 (IOMUXC_SPADC_PEPDC_D2)	32	R/W	0000_0084h	35.4.342/ 2065
53FA_8558	SW_PAD_CTL_PAD_EPDC_D3 (IOMUXC_SPADC_PEPDC_D3)	32	R/W	0000_0084h	35.4.343/ 2066
53FA_855C	SW_PAD_CTL_PAD_EPDC_D4 (IOMUXC_SPADC_PEPDC_D4)	32	R/W	0000_0084h	35.4.344/ 2068
53FA_8560	SW_PAD_CTL_PAD_EPDC_D5 (IOMUXC_SPADC_PEPDC_D5)	32	R/W	0000_0084h	35.4.345/ 2069
53FA_8564	SW_PAD_CTL_PAD_EPDC_D6 (IOMUXC_SPADC_PEPDC_D6)	32	R/W	0000_0084h	35.4.346/ 2071
53FA_8568	SW_PAD_CTL_PAD_EPDC_D7 (IOMUXC_SPADC_PEPDC_D7)	32	R/W	0000_0084h	35.4.347/ 2072
53FA_856C	SW_PAD_CTL_PAD_EPDC_D8 (IOMUXC_SPADC_PEPDC_D8)	32	R/W	0000_0084h	35.4.348/ 2074
53FA_8570	SW_PAD_CTL_PAD_EPDC_D9 (IOMUXC_SPADC_PEPDC_D9)	32	R/W	0000_0084h	35.4.349/ 2075
53FA_8574	SW_PAD_CTL_PAD_EPDC_D10 (IOMUXC_SPADC_PEPDC_D10)	32	R/W	0000_0084h	35.4.350/ 2077
53FA_8578	SW_PAD_CTL_PAD_EPDC_D11 (IOMUXC_SPADC_PEPDC_D11)	32	R/W	0000_0084h	35.4.351/ 2078
53FA_857C	SW_PAD_CTL_PAD_EPDC_D12 (IOMUXC_SPADC_PEPDC_D12)	32	R/W	0000_0084h	35.4.352/ 2080
53FA_8580	SW_PAD_CTL_PAD_EPDC_D13 (IOMUXC_SPADC_PEPDC_D13)	32	R/W	0000_0084h	35.4.353/ 2081
53FA_8584	SW_PAD_CTL_PAD_EPDC_D14 (IOMUXC_SPADC_PEPDC_D14)	32	R/W	0000_0084h	35.4.354/ 2083
53FA_8588	SW_PAD_CTL_PAD_EPDC_D15 (IOMUXC_SPADC_PEPDC_D15)	32	R/W	0000_0084h	35.4.355/ 2084

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_858C	SW_PAD_CTL_PAD_EPDC_GDCLK (IOMUXC_SPADC_PEPDC_GDCLK)	32	R/W	0000_0084h	35.4.356/ 2086
53FA_8590	SW_PAD_CTL_PAD_EPDC_GDSP (IOMUXC_SPADC_PEPDC_GDSP)	32	R/W	0000_0084h	35.4.357/ 2087
53FA_8594	SW_PAD_CTL_PAD_EPDC_GDOE (IOMUXC_SPADC_PEPDC_GDOE)	32	R/W	0000_0084h	35.4.358/ 2089
53FA_8598	SW_PAD_CTL_PAD_EPDC_GDRL (IOMUXC_SPADC_PEPDC_GDRL)	32	R/W	0000_0084h	35.4.359/ 2090
53FA_859C	SW_PAD_CTL_PAD_EPDC_SDCLK (IOMUXC_SPADC_PEPDC_SDCLK)	32	R/W	0000_0084h	35.4.360/ 2092
53FA_85A0	SW_PAD_CTL_PAD_EPDC_SDOEZ (IOMUXC_SPADC_PEPDC_SDOEZ)	32	R/W	0000_0084h	35.4.361/ 2093
53FA_85A4	SW_PAD_CTL_PAD_EPDC_SDOED (IOMUXC_SPADC_PEPDC_SDOED)	32	R/W	0000_0084h	35.4.362/ 2095
53FA_85A8	SW_PAD_CTL_PAD_EPDC_SDOE (IOMUXC_SPADC_PEPDC_SDOE)	32	R/W	0000_0084h	35.4.363/ 2096
53FA_85AC	SW_PAD_CTL_PAD_EPDC_SDLE (IOMUXC_SPADC_PEPDC_SDLE)	32	R/W	0000_0084h	35.4.364/ 2098
53FA_85B0	SW_PAD_CTL_PAD_EPDC_SDCLKN (IOMUXC_SPADC_PEPDC_SDCLKN)	32	R/W	0000_0084h	35.4.365/ 2099
53FA_85B4	SW_PAD_CTL_PAD_EPDC_SDSHR (IOMUXC_SPADC_PEPDC_SDSHR)	32	R/W	0000_0084h	35.4.366/ 2101
53FA_85B8	SW_PAD_CTL_PAD_EPDC_PWRCOM (IOMUXC_SPADC_PEPDC_PWRCOM)	32	R/W	0000_0084h	35.4.367/ 2102
53FA_85BC	SW_PAD_CTL_PAD_EPDC_PWRSTAT (IOMUXC_SPADC_PEPDC_PWRSTAT)	32	R/W	0000_0084h	35.4.368/ 2104
53FA_85C0	SW_PAD_CTL_PAD_EPDC_PWRCTRL0 (IOMUXC_SPADC_PEPDC_PWRCTRL0)	32	R/W	0000_0084h	35.4.369/ 2105
53FA_85C4	SW_PAD_CTL_PAD_EPDC_PWRCTRL1 (IOMUXC_SPADC_PEPDC_PWRCTRL1)	32	R/W	0000_0084h	35.4.370/ 2107
53FA_85C8	SW_PAD_CTL_PAD_EPDC_PWRCTRL2 (IOMUXC_SPADC_PEPDC_PWRCTRL2)	32	R/W	0000_0084h	35.4.371/ 2108
53FA_85CC	SW_PAD_CTL_PAD_EPDC_PWRCTRL3 (IOMUXC_SPADC_PEPDC_PWRCTRL3)	32	R/W	0000_0084h	35.4.372/ 2110
53FA_85D0	SW_PAD_CTL_PAD_EPDC_VCOM0 (IOMUXC_SPADC_PEPDC_VCOM0)	32	R/W	0000_0084h	35.4.373/ 2111
53FA_85D4	SW_PAD_CTL_PAD_EPDC_VCOM1 (IOMUXC_SPADC_PEPDC_VCOM1)	32	R/W	0000_0084h	35.4.374/ 2113
53FA_85D8	SW_PAD_CTL_PAD_EPDC_BDR0 (IOMUXC_SPADC_PEPDC_BDR0)	32	R/W	0000_0084h	35.4.375/ 2114

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_85DC	SW_PAD_CTL_PAD_EPDC_BDR1 (IOMUXC_SPADC_PEPDC_BDR1)	32	R/W	0000_0084h	35.4.376/ 2116
53FA_85E0	SW_PAD_CTL_PAD_EPDC_SDCE0 (IOMUXC_SPADC_PEPDC_SDCE0)	32	R/W	0000_0084h	35.4.377/ 2117
53FA_85E4	SW_PAD_CTL_PAD_EPDC_SDCE1 (IOMUXC_SPADC_PEPDC_SDCE1)	32	R/W	0000_0084h	35.4.378/ 2119
53FA_85E8	SW_PAD_CTL_PAD_EPDC_SDCE2 (IOMUXC_SPADC_PEPDC_SDCE2)	32	R/W	0000_0084h	35.4.379/ 2120
53FA_85EC	SW_PAD_CTL_PAD_EPDC_SDCE3 (IOMUXC_SPADC_PEPDC_SDCE3)	32	R/W	0000_0084h	35.4.380/ 2122
53FA_85F0	SW_PAD_CTL_PAD_EPDC_SDCE4 (IOMUXC_SPADC_PEPDC_SDCE4)	32	R/W	0000_0084h	35.4.381/ 2123
53FA_85F4	SW_PAD_CTL_PAD_EPDC_SDCE5 (IOMUXC_SPADC_PEPDC_SDCE5)	32	R/W	0000_0084h	35.4.382/ 2125
53FA_85F8	SW_PAD_CTL_PAD_EIM_DA0 (IOMUXC_SPADC_PEIM_DA0)	32	R/W	0000_00E4h	35.4.383/ 2126
53FA_85FC	SW_PAD_CTL_PAD_EIM_DA1 (IOMUXC_SPADC_PEIM_DA1)	32	R/W	0000_00E4h	35.4.384/ 2128
53FA_8600	SW_PAD_CTL_PAD_EIM_DA2 (IOMUXC_SPADC_PEIM_DA2)	32	R/W	0000_00E4h	35.4.385/ 2129
53FA_8604	SW_PAD_CTL_PAD_EIM_DA3 (IOMUXC_SPADC_PEIM_DA3)	32	R/W	0000_00E4h	35.4.386/ 2131
53FA_8608	SW_PAD_CTL_PAD_EIM_DA4 (IOMUXC_SPADC_PEIM_DA4)	32	R/W	0000_00E4h	35.4.387/ 2132
53FA_860C	SW_PAD_CTL_PAD_EIM_DA5 (IOMUXC_SPADC_PEIM_DA5)	32	R/W	0000_00E4h	35.4.388/ 2134
53FA_8610	SW_PAD_CTL_PAD_EIM_DA6 (IOMUXC_SPADC_PEIM_DA6)	32	R/W	0000_00E4h	35.4.389/ 2135
53FA_8614	SW_PAD_CTL_PAD_EIM_DA7 (IOMUXC_SPADC_PEIM_DA7)	32	R/W	0000_00E4h	35.4.390/ 2137
53FA_8618	SW_PAD_CTL_PAD_EIM_DA8 (IOMUXC_SPADC_PEIM_DA8)	32	R/W	0000_00E4h	35.4.391/ 2138
53FA_861C	SW_PAD_CTL_PAD_EIM_DA9 (IOMUXC_SPADC_PEIM_DA9)	32	R/W	0000_00E4h	35.4.392/ 2140
53FA_8620	SW_PAD_CTL_PAD_EIM_DA10 (IOMUXC_SPADC_PEIM_DA10)	32	R/W	0000_00E4h	35.4.393/ 2141
53FA_8624	SW_PAD_CTL_PAD_EIM_DA11 (IOMUXC_SPADC_PEIM_DA11)	32	R/W	0000_00E4h	35.4.394/ 2143
53FA_8628	SW_PAD_CTL_PAD_EIM_DA12 (IOMUXC_SPADC_PEIM_DA12)	32	R/W	0000_00E4h	35.4.395/ 2144

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_862C	SW_PAD_CTL_PAD_EIM_DA13 (IOMUXC_SPADC_PEIM_DA13)	32	R/W	0000_00E4h	35.4.396/ 2146
53FA_8630	SW_PAD_CTL_PAD_EIM_DA14 (IOMUXC_SPADC_PEIM_DA14)	32	R/W	0000_00E4h	35.4.397/ 2147
53FA_8634	SW_PAD_CTL_PAD_EIM_DA15 (IOMUXC_SPADC_PEIM_DA15)	32	R/W	0000_00E4h	35.4.398/ 2149
53FA_8638	SW_PAD_CTL_PAD_EIM_CS2 (IOMUXC_SPADC_PEIM_CS2)	32	R/W	0000_00E4h	35.4.399/ 2150
53FA_863C	SW_PAD_CTL_PAD_EIM_CS1 (IOMUXC_SPADC_PEIM_CS1)	32	R/W	0000_00E4h	35.4.400/ 2152
53FA_8640	SW_PAD_CTL_PAD_EIM_CS0 (IOMUXC_SPADC_PEIM_CS0)	32	R/W	0000_00E4h	35.4.401/ 2153
53FA_8644	SW_PAD_CTL_PAD_EIM_EB0 (IOMUXC_SPADC_PEIM_EB0)	32	R/W	0000_00E4h	35.4.402/ 2155
53FA_8648	SW_PAD_CTL_PAD_EIM_EB1 (IOMUXC_SPADC_PEIM_EB1)	32	R/W	0000_00E4h	35.4.403/ 2156
53FA_864C	SW_PAD_CTL_PAD_EIM_WAIT (IOMUXC_SPADC_PEIM_WAIT)	32	R/W	0000_00E0h	35.4.404/ 2158
53FA_8650	SW_PAD_CTL_PAD_EIM_BCLK (IOMUXC_SPADC_PEIM_BCLK)	32	R/W	0000_00E4h	35.4.405/ 2159
53FA_8654	SW_PAD_CTL_PAD_EIM_RDY (IOMUXC_SPADC_PEIM_RDY)	32	R/W	0000_00E4h	35.4.406/ 2161
53FA_8658	SW_PAD_CTL_PAD_EIM_OE (IOMUXC_SPADC_PEIM_OE)	32	R/W	0000_00E4h	35.4.407/ 2162
53FA_865C	SW_PAD_CTL_PAD_EIM_RW (IOMUXC_SPADC_PEIM_RW)	32	R/W	0000_00E4h	35.4.408/ 2164
53FA_8660	SW_PAD_CTL_PAD_EIM_LBA (IOMUXC_SPADC_PEIM_LBA)	32	R/W	0000_00E4h	35.4.409/ 2165
53FA_8664	SW_PAD_CTL_PAD_EIM_CRE (IOMUXC_SPADC_PEIM_CRE)	32	R/W	0000_00E4h	35.4.410/ 2167
53FA_8668	SW_PAD_CTL_GRP_ADDDS (IOMUXC_SPAD_GADDDS)	32	R/W	0008_0000h	35.4.411/ 2168
53FA_866C	SW_PAD_CTL_GRP_DDRMODE_CTL (IOMUXC_SPAD_GDDRMODE_CTL)	32	R/W	0000_0200h	35.4.412/ 2169
53FA_8670	SW_PAD_CTL_GRP_DDRPKE (IOMUXC_SPAD_GDDRPKE)	32	R/W	0000_0080h	35.4.413/ 2169
53FA_8674	SW_PAD_CTL_GRP_EIM (IOMUXC_SPAD_GEIM)	32	R/W	0000_0000h	35.4.414/ 2170
53FA_8678	SW_PAD_CTL_GRP_EPDC (IOMUXC_SPAD_GEPDC)	32	R/W	0000_0000h	35.4.415/ 2171

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_867C	SW_PAD_CTL_GRP_UART (IOMUXC_SPAD_GUART)	32	R/W	0000_0000h	35.4.416/ 2171
53FA_8680	SW_PAD_CTL_GRP_DDRPK (IOMUXC_SPAD_GDDRPK)	32	R/W	0000_0000h	35.4.417/ 2172
53FA_8684	SW_PAD_CTL_GRP_DDRHYS (IOMUXC_SPAD_GDDRHYS)	32	R/W	0000_0000h	35.4.418/ 2173
53FA_8688	SW_PAD_CTL_GRP_KEYPAD (IOMUXC_SPAD_GKEYPAD)	32	R/W	0000_0000h	35.4.419/ 2173
53FA_868C	SW_PAD_CTL_GRP_DDRMODE (IOMUXC_SPAD_GDDRMODE)	32	R/W	0000_0000h	35.4.420/ 2174
53FA_8690	SW_PAD_CTL_GRP_SSI (IOMUXC_SPAD_GSSI)	32	R/W	0000_0000h	35.4.421/ 2175
53FA_8694	SW_PAD_CTL_GRP_SD1 (IOMUXC_SPAD_GSD1)	32	R/W	0000_0000h	35.4.422/ 2175
53FA_8698	SW_PAD_CTL_GRP_B0DS (IOMUXC_SPAD_GB0DS)	32	R/W	0008_0000h	35.4.423/ 2176
53FA_869C	SW_PAD_CTL_GRP_SD2 (IOMUXC_SPAD_GSD2)	32	R/W	0000_0000h	35.4.424/ 2176
53FA_86A0	SW_PAD_CTL_GRP_B1DS (IOMUXC_SPAD_GB1DS)	32	R/W	0008_0000h	35.4.425/ 2177
53FA_86A4	SW_PAD_CTL_GRP_CTLDS (IOMUXC_SPAD_GCTLDS)	32	R/W	0008_0000h	35.4.426/ 2178
53FA_86A8	SW_PAD_CTL_GRP_B2DS (IOMUXC_SPAD_GB2DS)	32	R/W	0008_0000h	35.4.427/ 2178
53FA_86AC	SW_PAD_CTL_GRP_DDR_TYPE (IOMUXC_SPAD_GDDR_TYPE)	32	R/W	0400_0000h	35.4.428/ 2179
53FA_86B0	SW_PAD_CTL_GRP_LCD (IOMUXC_SPAD_GLCD)	32	R/W	0000_0000h	35.4.429/ 2180
53FA_86B4	SW_PAD_CTL_GRP_B3DS (IOMUXC_SPAD_GB3DS)	32	R/W	0008_0000h	35.4.430/ 2180
53FA_86B8	SW_PAD_CTL_GRP_MISC (IOMUXC_SPAD_GMISC)	32	R/W	0000_0000h	35.4.431/ 2181
53FA_86BC	SW_PAD_CTL_GRP_SPI (IOMUXC_SPAD_GSPI)	32	R/W	0000_0000h	35.4.432/ 2182
53FA_86C0	SW_PAD_CTL_GRP_NANDF (IOMUXC_SPAD_GNANDF)	32	R/W	0000_0000h	35.4.433/ 2182
53FA_86C4	AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT (IOMUXC_API_DA_AMX_SI)	32	R/W	0000_0000h	35.4.434/ 2183
53FA_86C8	AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT (IOMUXC_API_DB_AMX_SI)	32	R/W	0000_0000h	35.4.435/ 2183

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_86CC	AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT (IOMUXC_API_RXCLK_AMX_SI)	32	R/W	0000_0000h	35.4.436/ 2184
53FA_86D0	AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT (IOMUXC_API_RXFS_AMX_SI)	32	R/W	0000_0000h	35.4.437/ 2184
53FA_86D4	AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT (IOMUXC_API_TXCLK_AMX_SI)	32	R/W	0000_0000h	35.4.438/ 2185
53FA_86D8	AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT (IOMUXC_API_TXFS_AMX_SI)	32	R/W	0000_0000h	35.4.439/ 2185
53FA_86DC	CCM_PLL1_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL1_BYPASS_CLK_SI)	32	R/W	0000_0000h	35.4.440/ 2186
53FA_86E0	CCM_PLL2_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL2_BYPASS_CLK_SI)	32	R/W	0000_0000h	35.4.441/ 2186
53FA_86E4	CCM_PLL3_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL3_BYPASS_CLK_SI)	32	R/W	0000_0000h	35.4.442/ 2187
53FA_86E8	CSPI_IPP_IND_DATAREADY_B_SELECT_INPUT (IOMUXC_CSPI_DRDY_SI)	32	R/W	0000_0000h	35.4.443/ 2187
53FA_86EC	CSPI_IPP_IND_SS1_B_SELECT_INPUT (IOMUXC_CSPI_SS1_SI)	32	R/W	0000_0000h	35.4.444/ 2188
53FA_86F0	CSPI_IPP_IND_SS2_B_SELECT_INPUT (IOMUXC_CSPI_SS2_SI)	32	R/W	0000_0000h	35.4.445/ 2188
53FA_86F4	CSPI_IPP_IND_SS3_B_SELECT_INPUT (IOMUXC_CSPI_SS3_SI)	32	R/W	0000_0000h	35.4.446/ 2189
53FA_86F8	ELCDIF_LCDIF_BUSY_SELECT_INPUT (IOMUXC_ELCDIFL_BUSY_SI)	32	R/W	0000_0000h	35.4.447/ 2189
53FA_86FC	ELCDIF_LCDIF_RXDATA_0_SELECT_INPUT (IOMUXC_ELCDIFL_R0_SI)	32	R/W	0000_0000h	35.4.448/ 2190
53FA_8700	ELCDIF_LCDIF_RXDATA_1_SELECT_INPUT (IOMUXC_ELCDIFL_R1_SI)	32	R/W	0000_0000h	35.4.449/ 2190
53FA_8704	ELCDIF_LCDIF_RXDATA_2_SELECT_INPUT (IOMUXC_ELCDIFL_R2_SI)	32	R/W	0000_0000h	35.4.450/ 2191
53FA_8708	ELCDIF_LCDIF_RXDATA_3_SELECT_INPUT (IOMUXC_ELCDIFL_R3_SI)	32	R/W	0000_0000h	35.4.451/ 2191
53FA_870C	ELCDIF_LCDIF_RXDATA_4_SELECT_INPUT (IOMUXC_ELCDIFL_R4_SI)	32	R/W	0000_0000h	35.4.452/ 2192
53FA_8710	ELCDIF_LCDIF_RXDATA_5_SELECT_INPUT (IOMUXC_ELCDIFL_R5_SI)	32	R/W	0000_0000h	35.4.453/ 2192
53FA_8714	ELCDIF_LCDIF_RXDATA_6_SELECT_INPUT (IOMUXC_ELCDIFL_R6_SI)	32	R/W	0000_0000h	35.4.454/ 2193
53FA_8718	ELCDIF_LCDIF_RXDATA_7_SELECT_INPUT (IOMUXC_ELCDIFL_R7_SI)	32	R/W	0000_0000h	35.4.455/ 2193

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_871C	ELCDIF_LCDIF_RXDATA_8_SELECT_INPUT (IOMUXC_ELCDIFL_R8_SI)	32	R/W	0000_0000h	35.4.456/ 2194
53FA_8720	C_ELCDIF_LCDIF_RXDATA_9_SELECT_INPUT (IOMUXC_C_ELCDIFL_R9_SI)	32	R/W	0000_0000h	35.4.457/ 2194
53FA_8724	ELCDIF_LCDIF_RXDATA_10_SELECT_INPUT (IOMUXC_ELCDIFL_R10_SI)	32	R/W	0000_0000h	35.4.458/ 2195
53FA_8728	ELCDIF_LCDIF_RXDATA_11_SELECT_INPUT (IOMUXC_ELCDIFL_R11_SI)	32	R/W	0000_0000h	35.4.459/ 2195
53FA_872C	ELCDIF_LCDIF_RXDATA_12_SELECT_INPUT (IOMUXC_ELCDIFL_R12_SI)	32	R/W	0000_0000h	35.4.460/ 2196
53FA_8730	ELCDIF_LCDIF_RXDATA_13_SELECT_INPUT (IOMUXC_ELCDIFL_R13_SI)	32	R/W	0000_0000h	35.4.461/ 2196
53FA_8734	ELCDIF_LCDIF_RXDATA_14_SELECT_INPUT (IOMUXC_ELCDIFL_R14_SI)	32	R/W	0000_0000h	35.4.462/ 2197
53FA_8738	ELCDIF_LCDIF_RXDATA_15_SELECT_INPUT (IOMUXC_ELCDIFL_R15_SI)	32	R/W	0000_0000h	35.4.463/ 2197
53FA_873C	ELCDIF_VSYNC_I_SELECT_INPUT (IOMUXC_ELCDIF_VSYNC_I_SI)	32	R/W	0000_0000h	35.4.464/ 2198
53FA_8740	ESDHC2_IPP_CARD_DET_SELECT_INPUT (IOMUXC_ESDHC2_CDET_SI)	32	R/W	0000_0000h	35.4.465/ 2198
53FA_8744	ESDHC2_IPP_WP_ON_SELECT_INPUT (IOMUXC_ESDHC2_WP_ON_SI)	32	R/W	0000_0000h	35.4.466/ 2199
53FA_8748	ESDHC4_IPP_CARD_CLK_IN_SELECT_INPUT (IOMUXC_ESDHC4_CCLK_IN_SI)	32	R/W	0000_0000h	35.4.467/ 2199
53FA_874C	ESDHC4_IPP_CMD_IN_SELECT_INPUT (IOMUXC_ESDHC4_CMD_IN_SI)	32	R/W	0000_0000h	35.4.468/ 2200
53FA_8750	ESDHC4_IPP_DAT0_IN_SELECT_INPUT (IOMUXC_ESDHC4_DAT0_IN_SI)	32	R/W	0000_0000h	35.4.469/ 2200
53FA_8754	XC_ESDHC4_IPP_DAT1_IN_SELECT_INPUT (IOMUXC_XC_ESDHC4_1_IN_SI)	32	R/W	0000_0000h	35.4.470/ 2201
53FA_8758	ESDHC4_IPP_DAT2_IN_SELECT_INPUT (IOMUXC_ESDHC4_2_IN_SI)	32	R/W	0000_0000h	35.4.471/ 2201
53FA_875C	ESDHC4_IPP_DAT3_IN_SELECT_INPUT (IOMUXC_ESDHC4_3_IN_SI)	32	R/W	0000_0000h	35.4.472/ 2202
53FA_8760	ESDHC4_IPP_DAT4_IN_SELECT_INPUT (IOMUXC_ESDHC4_4_IN_SI)	32	R/W	0000_0000h	35.4.473/ 2202
53FA_8764	ESDHC4_IPP_DAT5_IN_SELECT_INPUT (IOMUXC_ESDHC4_5_IN_SI)	32	R/W	0000_0000h	35.4.474/ 2203
53FA_8768	ESDHC4_IPP_DAT6_IN_SELECT_INPUT (IOMUXC_ESDHC4_6_IN_SI)	32	R/W	0000_0000h	35.4.475/ 2203

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_876C	ESDHC4_IPP_DAT7_IN_SELECT_INPUT (IOMUXC_ESDHC4_7_IN_SI)	32	R/W	0000_0000h	35.4.476/ 2204
53FA_8770	FEC_FEC_COL_SELECT_INPUT (IOMUXC_FEC_COL_SI)	32	R/W	0000_0000h	35.4.477/ 2204
53FA_8774	FEC_FEC_MDI_SELECT_INPUT (IOMUXC_FEC_MDI_SI)	32	R/W	0000_0000h	35.4.478/ 2205
53FA_8778	FEC_FEC_RDATA_0_SELECT_INPUT (IOMUXC_FEC_RD0_SI)	32	R/W	0000_0000h	35.4.479/ 2205
53FA_877C	FEC_FEC_RDATA_1_SELECT_INPUT (IOMUXC_FEC_RD1_SI)	32	R/W	0000_0000h	35.4.480/ 2206
53FA_8780	FEC_FEC_RX_CLK_SELECT_INPUT (IOMUXC_FEC_RX_CLK_SI)	32	R/W	0000_0000h	35.4.481/ 2206
53FA_8784	FEC_FEC_RX_DV_SELECT_INPUT (IOMUXC_FEC_RX_DV_SI)	32	R/W	0000_0000h	35.4.482/ 2207
53FA_8788	FEC_FEC_RX_ER_SELECT_INPUT (IOMUXC_FEC_RX_ER_SI)	32	R/W	0000_0000h	35.4.483/ 2207
53FA_878C	FEC_FEC_TX_CLK_SELECT_INPUT (IOMUXC_FEC_TX_CLK_SI)	32	R/W	0000_0000h	35.4.484/ 2208
53FA_8790	KPP_IPP_IND_COL_4_SELECT_INPUT (IOMUXC_KPP_FC4_SI)	32	R/W	0000_0000h	35.4.485/ 2208
53FA_8794	KPP_IPP_IND_COL_5_SELECT_INPUT (IOMUXC_KPP_FC5_SI)	32	R/W	0000_0000h	35.4.486/ 2209
53FA_8798	KPP_IPP_IND_COL_6_SELECT_INPUT (IOMUXC_KPP_FC6_SI)	32	R/W	0000_0000h	35.4.487/ 2209
53FA_879C	KPP_IPP_IND_COL_7_SELECT_INPUT (IOMUXC_KPP_FC7_SI)	32	R/W	0000_0000h	35.4.488/ 2210
53FA_87A0	KPP_IPP_IND_ROW_4_SELECT_INPUT (IOMUXC_KPP_FR4_SI)	32	R/W	0000_0000h	35.4.489/ 2210
53FA_87A4	KPP_IPP_IND_ROW_5_SELECT_INPUT (IOMUXC_KPP_FR5_SI)	32	R/W	0000_0000h	35.4.490/ 2211
53FA_87A8	KPP_IPP_IND_ROW_6_SELECT_INPUT (IOMUXC_KPP_FR6_SI)	32	R/W	0000_0000h	35.4.491/ 2211
53FA_87AC	KPP_IPP_IND_ROW_7_SELECT_INPUT (IOMUXC_KPP_FR7_SI)	32	R/W	0000_0000h	35.4.492/ 2212
53FA_87B0	RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SELECT_INPUT (IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SI)	32	R/W	0000_0000h	35.4.493/ 2212
53FA_87B4	RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SELECT_INPUT (IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SI)	32	R/W	0000_0000h	35.4.494/ 2213

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_87B8	SDMA_EVENTS_14_SELECT_INPUT (IOMUXC_SDMA_EVENTS_14_SI)	32	R/W	0000_0000h	35.4.495/ 2213
53FA_87BC	SDMA_EVENTS_15_SELECT_INPUT (IOMUXC_SDMA_EVENTS_15_SI)	32	R/W	0000_0000h	35.4.496/ 2214
53FA_87C0	UART1_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U1U_RTS_B_SI)	32	R/W	0000_0000h	35.4.497/ 2214
53FA_87C4	UART1_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U1U_RXD_MUX_SI)	32	R/W	0000_0000h	35.4.498/ 2215
53FA_87C8	UART2_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U2U_RTS_B_SI)	32	R/W	0000_0000h	35.4.499/ 2215
53FA_87CC	UART2_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U2U_RXD_MUX_SI)	32	R/W	0000_0000h	35.4.500/ 2216
53FA_87D0	UART3_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U3U_RTS_B_SI)	32	R/W	0000_0000h	35.4.501/ 2217
53FA_87D4	UART3_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U3U_RXD_MUX_SI)	32	R/W	0000_0000h	35.4.502/ 2217
53FA_87D8	UART4_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U4U_RTS_B_SI)	32	R/W	0000_0000h	35.4.503/ 2218
53FA_87DC	UART4_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U4U_RXD_MUX_SI)	32	R/W	0000_0000h	35.4.504/ 2218
53FA_87E0	UART5_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U5U_RTS_B_SI)	32	R/W	0000_0000h	35.4.505/ 2219
53FA_87E4	UART5_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U5U_RXD_MUX_SI)	32	R/W	0000_0000h	35.4.506/ 2219
53FA_87E8	USBOH1_IPP_IND_OTG_OC_SELECT_INPUT (IOMUXC_USBOH1_OTG_OC_SI)	32	R/W	0000_0000h	35.4.507/ 2220
53FA_87EC	WEIMV2_IPP_IND_READ_DATA_0_SELECT_INPUT (IOMUXC_WEIMV2_RD0_SI)	32	R/W	0000_0000h	35.4.508/ 2220
53FA_87F0	WEIMV2_IPP_IND_READ_DATA_1_SELECT_INPUT (IOMUXC_WEIMV2_RD1_SI)	32	R/W	0000_0000h	35.4.509/ 2221
53FA_87F4	WEIMV2_IPP_IND_READ_DATA_2_SELECT_INPUT (IOMUXC_WEIMV2_RD2_SI)	32	R/W	0000_0000h	35.4.510/ 2221
53FA_87F8	WEIMV2_IPP_IND_READ_DATA_3_SELECT_INPUT (IOMUXC_WEIMV2_RD3_SI)	32	R/W	0000_0000h	35.4.511/ 2222
53FA_87FC	WEIMV2_IPP_IND_READ_DATA_4_SELECT_INPUT (IOMUXC_WEIMV2_RD4_SI)	32	R/W	0000_0000h	35.4.512/ 2222
53FA_8800	WEIMV2_IPP_IND_READ_DATA_5_SELECT_INPUT (IOMUXC_WEIMV2_RD5_SI)	32	R/W	0000_0000h	35.4.513/ 2223
53FA_8804	WEIMV2_IPP_IND_READ_DATA_6_SELECT_INPUT (IOMUXC_WEIMV2_RD6_SI)	32	R/W	0000_0000h	35.4.514/ 2223

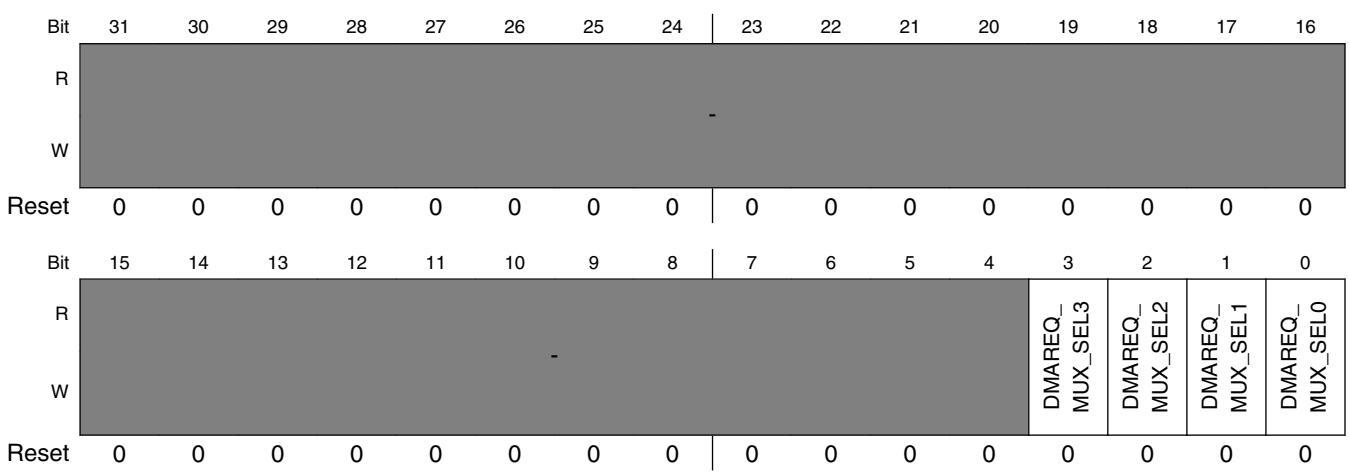
Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_8808	WEIMV2_IPP_IND_READ_DATA_7_SELECT_INPUT (IOMUXC_WEIMV2_RD7_SI)	32	R/W	0000_0000h	35.4.515/ 2224
53FA_880C	WEIMV2_IPP_IND_READ_DATA_8_SELECT_INPUT (IOMUXC_WEIMV2_RD8_SI)	32	R/W	0000_0000h	35.4.516/ 2224
53FA_8810	WEIMV2_IPP_IND_READ_DATA_9_SELECT_INPUT (IOMUXC_WEIMV2_RD9_SI)	32	R/W	0000_0000h	35.4.517/ 2225
53FA_8814	WEIMV2_IPP_IND_READ_DATA_10_SELECT_INPUT (IOMUXC_WEIMV2_RD10_SI)	32	R/W	0000_0000h	35.4.518/ 2225
53FA_8818	WEIMV2_IPP_IND_READ_DATA_11_SELECT_INPUT (IOMUXC_WEIMV2_RD11_SI)	32	R/W	0000_0000h	35.4.519/ 2226
53FA_881C	WEIMV2_IPP_IND_READ_DATA_12_SELECT_INPUT (IOMUXC_WEIMV2_RD12_SI)	32	R/W	0000_0000h	35.4.520/ 2226
53FA_8820	WEIMV2_IPP_IND_READ_DATA_13_SELECT_INPUT (IOMUXC_WEIMV2_RD13_SI)	32	R/W	0000_0000h	35.4.521/ 2227
53FA_8824	WEIMV2_IPP_IND_READ_DATA_14_SELECT_INPUT (IOMUXC_WEIMV2_RD14_SI)	32	R/W	0000_0000h	35.4.522/ 2227
53FA_8828	WEIMV2_IPP_IND_READ_DATA_15_SELECT_INPUT (IOMUXC_WEIMV2_RD15_SI)	32	R/W	0000_0000h	35.4.523/ 2228

35.4.1 GPR0 (IOMUXC_GPR0)

Address: IOMUXC_GPR0 is 53FA_8000h base + 0h offset = 53FA_8000h



IOMUXC_GPR0 field descriptions

Field	Description
31–4 -	Reserved

Table continues on the next page...

IOMUXC_GPR0 field descriptions (continued)

Field	Description
3 DMAREQ_ MUX_SEL3	Selects between two possible sources for SDMA_EVENT[11]: 0 esdhc4.ipd_esdhcv2_dreq_b 1 cti2.CTITRIGOUT[0]
2 DMAREQ_ MUX_SEL2	Selects between two possible sources for SDMA_EVENT[10]: 0 i2c3.ipi_int_b 1 esdhc3.ipd_esdhcv3_dreq_b
1 DMAREQ_ MUX_SEL1	Selects between two possible sources for SDMA_EVENT[21]: 0 i2c2.ipi_int_b 1 esdhc2.ipd_esdhcv2_dreq_b
0 DMAREQ_ MUX_SEL0	Selects between two possible sources for SDMA_EVENT[20]: 0 i2c1.ipi_int_b 1 esdhc1.ipd_esdhcv2_dreq_b

35.4.2 GPR1 (IOMUXC_GPR1)

Address: IOMUXC_GPR1 is 53FA_8000h base + 4h offset = 53FA_8004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-				ADDRS3[1:0]		ACT_CS3		ADDRS2[1:0]		ACT_CS2		ADDRS1[1:0]		ACT_CS1		ADDRS0[1:0]		ACT_CS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_GPR1 field descriptions

Field	Description
31–12 -	Reserved
11–10 ADDRS3[1:0]	Active Chip Select and Address Space. Each of the ACT_CSx represents one of the four chip selects of the WEIM. When ACT_CSx=1'b1, the corresponding chip select is active and has a valid address space according to its address space configuration determined by ADDRSx[1:0] bits

Table continues on the next page...

IOMUXC_GPR1 field descriptions (continued)

Field	Description
	ADDRSx[1:0] is setting the space for each chip select which is active. The address space of the first active chip select must be the biggest one, the following active chip select address spaces may be equal or lower. Total address space size is 512 Mbyte. Address Space Configuration options (ADDRSx[1:0]): 00 32 Mbyte 01 64 Mbyte 10 128 Mbyte 11 256 Mbyte
9 ACT_CS3	See ADDRS3 for description
8–7 ADDRS2[1:0]	See ADDRS3 for description
6 ACT_CS2	See ADDRS3 for description
5–4 ADDRS1[1:0]	See ADDRS3 for description
3 ACT_CS1	See ADDRS3 for description
2–1 ADDRS0[1:0]	See ADDRS3 for description
0 ACT_CS0	See ADDRS3 for description

35.4.3 GPR2 (IOMUXC_GPR2)

Address: IOMUXC_GPR2 is 53FA_8000h base + 8h offset = 53FA_8008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																												DRAM_DQ_INPUTON[3:0]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR2 field descriptions

Field	Description
31–4 -	Reserved
3–0 DRAM_DQ_INPUTON[3:0]	set 1 to override input path enable on DRAM DQ/DQS/DQM pads, to make it always turn on for loop-back test GPR2[3] control dram_dq[31:24], dqm[3], sdqs[3]

Table continues on the next page...

IOMUXC_GPR2 field descriptions (continued)

Field	Description
	GPR2[2] control dram_dq[23:16], dqm[2], sdqs[2] GPR2[1] control dram_dq[15:8], dqm[1], sdqs[1] GPR2[0] control dram_dq[7:0], dqm[0], sdqs[0]

35.4.4 OBSERVE_MUX_0 (IOMUXC_OBSMUX0)

Address: IOMUXC_OBSMUX0 is 53FA_8000h base + Ch offset = 53FA_800Ch

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

IOMUXC_OBSMUX0 field descriptions

Field	Description
31–6 -	Reserved
5–0 OBSRV	Select Instance Pin for Observability IOMUXC_OBSERVE_MUX_0 000000 Select Instance ahbmax, Pin max_halted 000001 Select Instance ccm, Pin ccm_clk_switch_ack 000010 Select Instance ccm, Pin ccm_ipg_stop 000011 Select Instance ccm, Pin ccm_ipg_wait 000100 Select Instance ccm, Pin ccm_pdn_4all_req 000101 Select Instance ccm, Pin hndsk_current_state[0] 000110 Select Instance ccm, Pin ipi_int_1 000111 Select Instance ccm, Pin ipi_int_2 001000 Select Instance ccm, Pin lpm_current_state[0] 001001 Select Instance ccm, Pin shd_current_state[0] 001010 Select Instance cspi, Pin ~ipi_int_cspi_b 001011 Select Instance dpllip1, Pin dpllip_cpen 001100 Select Instance ecspi1, Pin ipd_req_cspi_rdma_b 001101 Select Instance ecspi1, Pin ipd_req_cspi_tdma_b 001110 Select Instance ecspi1, Pin ~ipi_int_cspi_b 001111 Select Instance ecspi2, Pin ipd_req_cspi_rdma_b 010000 Select Instance ecspi2, Pin ~ipi_int_cspi_b 010001 Select Instance epit1, Pin ipi_int_epit_oc 010010 Select Instance esdhc1, Pin ~ipi_esdhcv2_irq_b 010011 Select Instance esdhc2, Pin ~ipi_esdhcv2_irq_b 010100 Select Instance esdhc3, Pin ~ipi_esdhcv3_irq_b 010101 Select Instance esdhc4, Pin ~ipi_esdhcv2_irq_b 010110 Select Instance fec, Pin fec_ipi_int 010111 Select Instance gpc, Pin gpc_cta8_pg[0]

Table continues on the next page...

IOMUXC_OBSMUX0 field descriptions (continued)

Field	Description
	011000 Select Instance gpc, Pin gpc_cta8_pg[1] 011001 Select Instance gpc, Pin gpc_cta8_pg[2] 011010 Select Instance gpc, Pin gpc_cta8_pg[3] 011011 Select Instance gpc, Pin gpc_cta8_pg[4] 011100 Select Instance gpc, Pin gpc_emi_pg[0] 011101 Select Instance gpc, Pin gpc_emi_short_b 011110 Select Instance gpc, Pin gpc_event 011111 Select Instance gpc, Pin gpc_int 100000 Select Instance gpc, Pin gpc_int2 100001 Select Instance gpc, Pin gpc_neon_pg[0] 100010 Select Instance src, Pin arm_por_rst 100011 Select Instance src, Pin warm_reset 100100 Select Instance tigerp_platform_ne_32k_256k, Pin dsm_request 100101 Select Instance tigerp_platform_ne_32k_256k, Pin ~nm_irq_b 100110 Select Instance tzic, Pin tzic_fiq_b 100111 Select Instance tzic, Pin tzic_irq_b

35.4.5 OBSERVE_MUX_1 (IOMUXC_OBSMUX1)

Address: IOMUXC_OBSMUX1 is 53FA_8000h base + 10h offset = 53FA_8010h

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

IOMUXC_OBSMUX1 field descriptions

Field	Description
31–6 -	Reserved
5–0 OBSRV	Select Instance Pin for Observability IOMUXC_OBSERVE_MUX_1 000000 Select Instance ccm, Pin ahbmax_halt_req 000001 Select Instance ccm, Pin ccm_pdn_4arm_req 000010 Select Instance ccm, Pin ccm_pup_req 000011 Select Instance ccm, Pin ccm_system_in_stop_mode 000100 Select Instance ccm, Pin ccm_system_in_wait_mode 000101 Select Instance ccm, Pin dpll_en_dpllip 000110 Select Instance ccm, Pin weim_lpmd 000111 Select Instance ccm, Pin hndsk_current_state[1] 001000 Select Instance ccm, Pin lpm_current_state[1] 001001 Select Instance ccm, Pin shd_current_state[1] 001010 Select Instance dpllip2, Pin dpllip_cpen

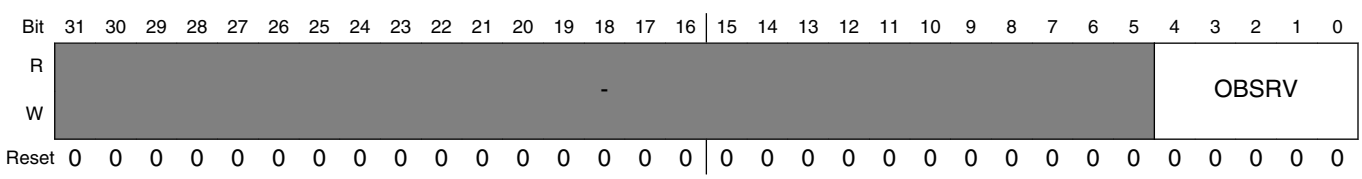
Table continues on the next page...

IOMUXC_OBSMUX1 field descriptions (continued)

Field	Description
	001011 Select Instance ecspi2, Pin ipd_req_cspi_tdma_b 001100 Select Instance gpc, Pin gpc_cta8_clk_upd_req 001101 Select Instance gpc, Pin gpc_cta8_pg[5] 001110 Select Instance gpc, Pin gpc_cta8_pg[6] 001111 Select Instance gpc, Pin gpc_cta8_pg[7] 010000 Select Instance gpc, Pin gpc_cta8_pg[8] 010001 Select Instance gpc, Pin gpc_cta8_pg[9] 010010 Select Instance gpc, Pin gpc_emi_pg[1] 010011 Select Instance gpc, Pin gpc_ipu_switch_b 010100 Select Instance gpc, Pin gpc_l1bits_pwrdwn 010101 Select Instance gpc, Pin gpc_l2bits_pwrdwn 010110 Select Instance gpc, Pin gpc_neon_pg[1] 010111 Select Instance gpio1, Pin ipi_gpio_int15_0 011000 Select Instance gpio1, Pin ipi_gpio_int31_16 011001 Select Instance gpio1, Pin ipi_gpio_int32[0] 011010 Select Instance gpio1, Pin ipi_gpio_int32[1] 011011 Select Instance gpio1, Pin ipi_gpio_int32[2] 011100 Select Instance gpio1, Pin ipi_gpio_int32[3] 011101 Select Instance gpio1, Pin ipi_gpio_int32[4] 011110 Select Instance gpio1, Pin ipi_gpio_int32[5] 011111 Select Instance gpio1, Pin ipi_gpio_int32[6] 100000 Select Instance gpio1, Pin ipi_gpio_int32[7] 100001 Select Instance gpio2, Pin ipi_gpio_int15_0 100010 Select Instance gpio2, Pin ipi_gpio_int31_16 100011 Select Instance gpio3, Pin ipi_gpio_int15_0 100100 Select Instance gpio3, Pin ipi_gpio_int31_16 100101 Select Instance gpio4, Pin ipi_gpio_int15_0 100110 Select Instance gpio4, Pin ipi_gpio_int31_16 100111 Select Instance gpt, Pin ipi_int_gpt 101000 Select Instance src, Pin arm_soc_rst_b 101001 Select Instance tigerp_platform_ne_32k_256k, Pin dsm_request 101010 Select Instance uart1, Pin ipd_uart_rx_dmareq_b 101011 Select Instance uart1, Pin ipd_uart_tx_dmareq_b 101100 Select Instance uart2, Pin ipd_uart_rx_dmareq_b 101101 Select Instance uart2, Pin ipd_uart_tx_dmareq_b 101110 Select Instance wdog1, Pin wdog_rst_b

35.4.6 OBSERVE_MUX_2 (IOMUXC_OBSMUX2)

Address: IOMUXC_OBSMUX2 is 53FA_8000h base + 14h offset = 53FA_8014h

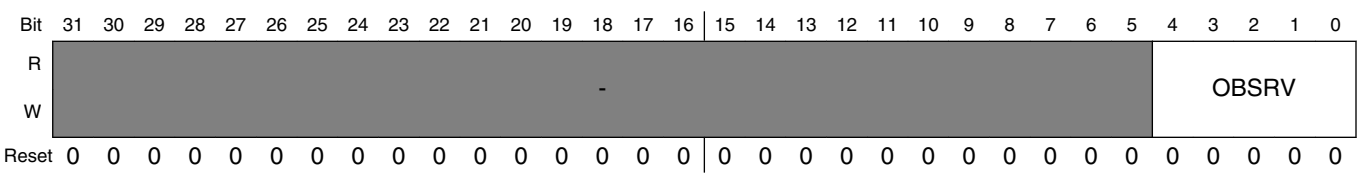


IOMUXC_OBSMUX2 field descriptions

Field	Description
31–5 -	Reserved
4–0 OBSRV	Select Instance Pin for Observability IOMUXC_OBSERVE_MUX_2 00000 Select Instance ccm, Pin hndsk_current_state[2] 00001 Select Instance ccm, Pin lpm_current_state[2] 00010 Select Instance dpllip3, Pin dpllip_cpen 00011 Select Instance gpc, Pin gpc_core_pwrdwn 00100 Select Instance gpc, Pin gpc_cta8_clk_dvfs_operation 00101 Select Instance gpc, Pin gpc_cta8_pg[10] 00110 Select Instance gpc, Pin gpc_cta8_pg[11] 00111 Select Instance gpc, Pin gpc_cta8_pg[12] 01000 Select Instance gpc, Pin gpc_cta8_pg[13] 01001 Select Instance gpc, Pin gpc_cta8_pg[14] 01010 Select Instance gpc, Pin gpc_emi_pg[2] 01011 Select Instance gpc, Pin gpc_neon_pg[2] 01100 Select Instance gpc, Pin gpc_neon_pwrdwn 01101 Select Instance gpc, Pin gpc_pdn_ack 01110 Select Instance gpc, Pin gpc_pup_ack 01111 Select Instance i2c1, Pin ~ipi_int_b 10000 Select Instance i2c2, Pin ~ipi_int_b 10001 Select Instance kpp, Pin ~ipi_int_kpp_b 10010 Select Instance owire, Pin ipi_owire_int 10011 Select Instance pwm1, Pin ipi_int_pwm 10100 Select Instance pwm2, Pin ipi_int_pwm 10101 Select Instance src, Pin sjc_por_rst_b 10110 Select Instance src, Pin system_rst_b 10111 Select Instance ssi1, Pin ipd_ssi_rx0_dmareq_b 11000 Select Instance ssi1, Pin ipd_ssi_rx1_dmareq_b 11001 Select Instance ssi1, Pin ipd_ssi_tx1_dmareq_b 11010 Select Instance ssi2, Pin ipd_ssi_rx1_dmareq_b

35.4.7 OBSERVE_MUX_3 (IOMUXC_OBSMUX3)

Address: IOMUXC_OBSMUX3 is 53FA_8000h base + 18h offset = 53FA_8018h

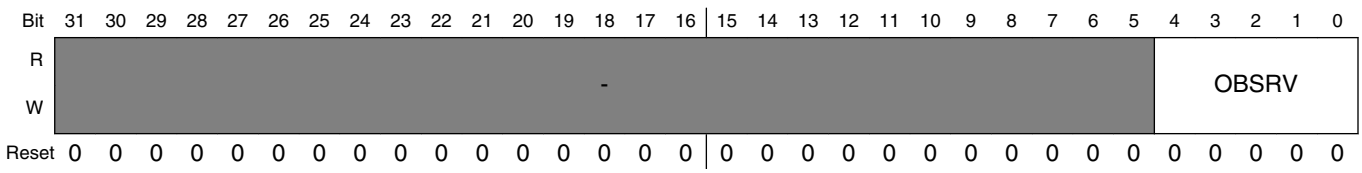


IOMUXC_OBSMUX3 field descriptions

Field	Description
31–5 -	Reserved
4–0 OBSRV	Select Instance Pin for Observability IOMUXC_OBSERVE_MUX_3 00000 Select Instance cspi, Pin ipd_req_cspi_rdma_b 00001 Select Instance cspi, Pin ipd_req_cspi_tdma_b 00010 Select Instance dpllip1, Pin dpllip_lrf_sticky 00011 Select Instance dpllip2, Pin dpllip_lrf_sticky 00100 Select Instance dpllip3, Pin dpllip_lrf_sticky 00101 Select Instance gpc, Pin gpc_cta8_pg[15] 00110 Select Instance gpc, Pin gpc_cta8_pg[16] 00111 Select Instance gpc, Pin gpc_cta8_pg[17] 01000 Select Instance gpc, Pin gpc_cta8_pg[18] 01001 Select Instance gpc, Pin gpc_cta8_pg[19] 01010 Select Instance gpc, Pin gpc_emi_pg[3] 01011 Select Instance gpc, Pin gpc_freq_change_mode 01100 Select Instance gpc, Pin gpc_neon_pg[3] 01101 Select Instance gpc, Pin gpc_neon_short_b 01110 Select Instance sdma, Pin ipg_stop_ack 01111 Select Instance sdma, Pin ~ipi_host_intr_b 10000 Select Instance src, Pin any_pu_rst_b 10001 Select Instance src, Pin emi_rst_b 10010 Select Instance src, Pin ipi_int_1 10011 Select Instance src, Pin system_early_rst_b 10100 Select Instance srtc, Pin ~ipi_srtc_int_b 10101 Select Instance srtc, Pin ~ipi_srtc_sec_int_b 10110 Select Instance ssi1, Pin ipd_ssi_tx0_dmareq_b 10111 Select Instance ssi1, Pin ~ipi_int_b 11000 Select Instance ssi2, Pin ~ipi_int_b 11001 Select Instance tigerp_platform_ne_32k_256k, Pin nm_irq_b

35.4.8 OBSERVE_MUX_4 (IOMUXC_OBSMUX4)

Address: IOMUXC_OBSMUX4 is 53FA_8000h base + 1Ch offset = 53FA_801Ch

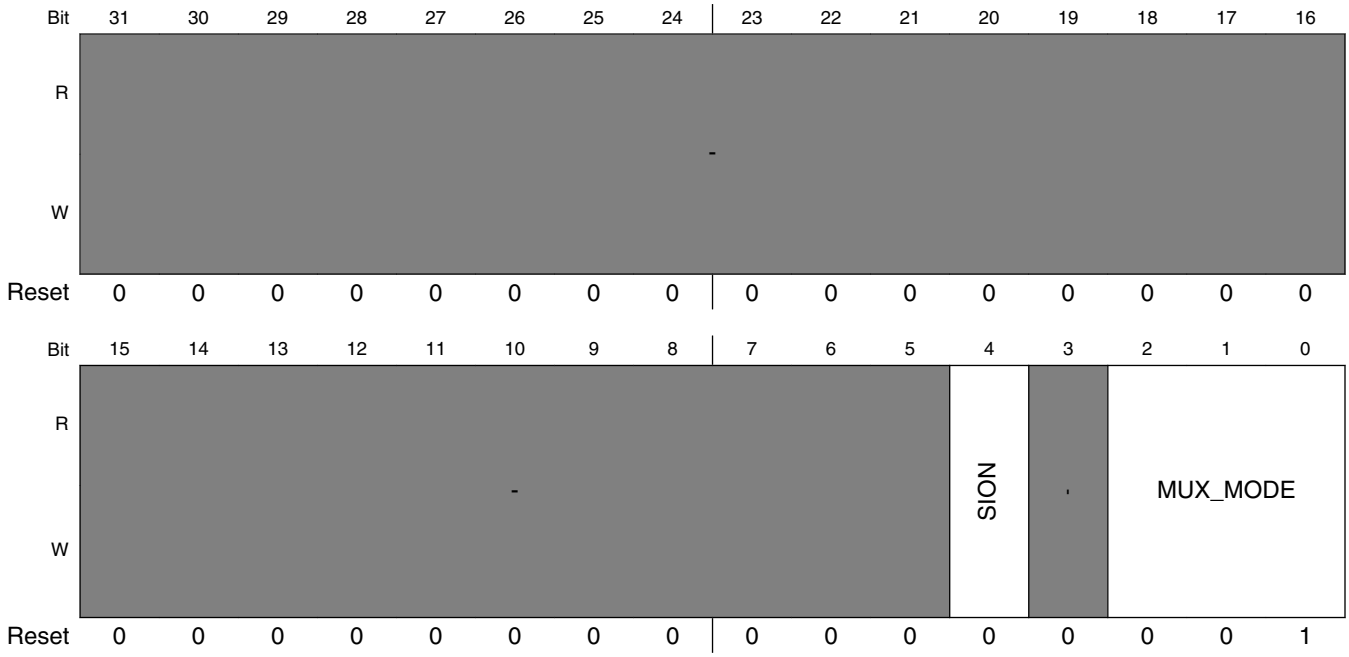


IOMUXC_OBSMUX4 field descriptions

Field	Description
31–5 -	Reserved
4–0 OBSRV	Select Instance Pin for Observability IOMUXC_OBSERVE_MUX_4 00000 Select Instance ccm, Pin pll_lvs 00001 Select Instance ccm, Pin sdma_ipg_stop_req 00010 Select Instance ccm, Pin src_clock_ready 00011 Select Instance esdhc3, Pin ipd_esdhcv3_dreq_b 00100 Select Instance esdhc4, Pin ipd_esdhcv2_dreq_b 00101 Select Instance gpc, Pin gpc_cta8_iso 00110 Select Instance gpc, Pin gpc_cta8_pg[20] 00111 Select Instance gpc, Pin gpc_emi_pg[4] 01000 Select Instance gpc, Pin gpc_ipu_pg_event 01001 Select Instance gpc, Pin gpc_ipu_stat_pg 01010 Select Instance gpc, Pin volt_chng 01011 Select Instance sfp, Pin sfp_ready 01100 Select Instance src, Pin emi_dvfs_req 01101 Select Instance src, Pin en_system_clk 01110 Select Instance src, Pin memory_repair_mode 01111 Select Instance src, Pin power_gating_reset_done 10000 Select Instance tigerp_platform_ne_32k_256k, Pin cti1_trigout0 10001 Select Instance tigerp_platform_ne_32k_256k, Pin cti1_trigout1 10010 Select Instance tigerp_platform_ne_32k_256k, Pin cti1_trigout2 10011 Select Instance tigerp_platform_ne_32k_256k, Pin cti1_trigout3 10100 Select Instance tigerp_platform_ne_32k_256k, Pin ~cti_irq_b 10101 Select Instance tigerp_platform_ne_32k_256k, Pin dbgack 10110 Select Instance tigerp_platform_ne_32k_256k, Pin ~pmu_irq_b 10111 Select Instance tzic, Pin tzic_wakeup_request 11000 Select Instance uart1, Pin ~ipi_uart_anded_b 11001 Select Instance uart2, Pin ~ipi_uart_anded_b 11010 Select Instance uart3, Pin ipd_uart_rx_dmareq_b 11011 Select Instance uart3, Pin ipd_uart_tx_dmareq_b 11100 Select Instance uart3, Pin ~ipi_uart_anded_b 11101 Select Instance usboh1, Pin ipi_int_uh1 11110 Select Instance usboh1, Pin ipi_int_uotg 11111 Select Instance wdog1, Pin ~ipi_wdog_int_b

35.4.9 SW_MUX_CTL_PAD_KEY_COL0 (IOMUXC_SMUXC_PKC0)

Address: IOMUXC_SMUXC_PKC0 is 53FA_8000h base + 20h offset = 53FA_8020h

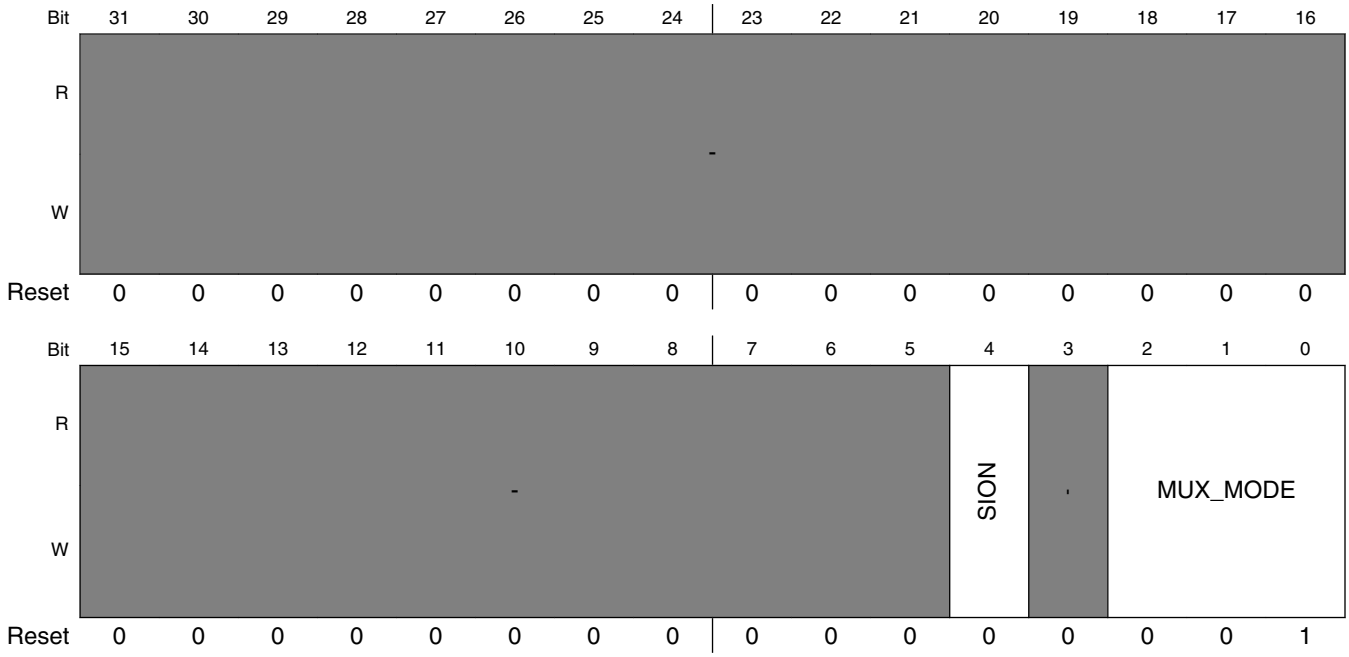


IOMUXC_SMUXC_PKC0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad KEY_COL0 0 Input Path is determined by functionality of the selected mux mode (regular)
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KEY_COL0. 000 Select mux mode: ALT0 mux port: COL[0] of instance: kpp 001 Select mux mode: ALT1 mux port: GPIO[0] of instance: gpio4 010 Select mux mode: ALT2 mux port: CLE of instance: rawnand 110 Select mux mode: ALT6 mux port: CTI_TRIGIN7 of instance: tigerp_platform_ne_32k_256k 111 Select mux mode: ALT7 mux port: TXREADY of instance: usbphy1.

35.4.10 SW_MUX_CTL_PAD_KEY_ROW0 (IOMUXC_SMUXC_PKR0)

Address: IOMUXC_SMUXC_PKR0 is 53FA_8000h base + 24h offset = 53FA_8024h

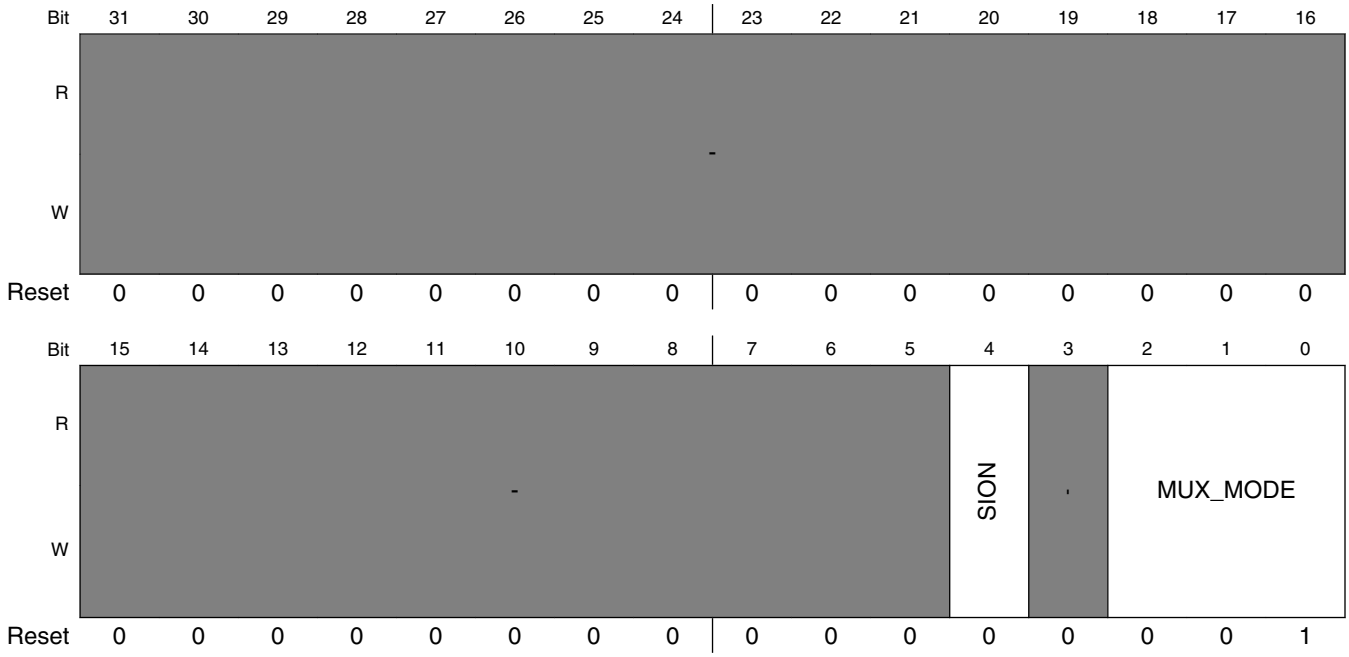


IOMUXC_SMUXC_PKR0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad KEY_ROW0 0 Input Path is determined by functionality of the selected mux mode (regular)
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KEY_ROW0. 000 Select mux mode: ALT0 mux port: ROW[0] of instance: kpp 001 Select mux mode: ALT1 mux port: GPIO[1] of instance: gpio4 010 Select mux mode: ALT2 mux port: ALE of instance: rawnand 110 Select mux mode: ALT6 mux port: CTI_TRIGIN_ACK7 of instance: tigerp_platform_ne_32k_256k 111 Select mux mode: ALT7 mux port: RXVALID of instance: usbphy1

35.4.11 SW_MUX_CTL_PAD_KEY_COL1 (IOMUXC_SMUXC_PKC1)

Address: IOMUXC_SMUXC_PKC1 is 53FA_8000h base + 28h offset = 53FA_8028h

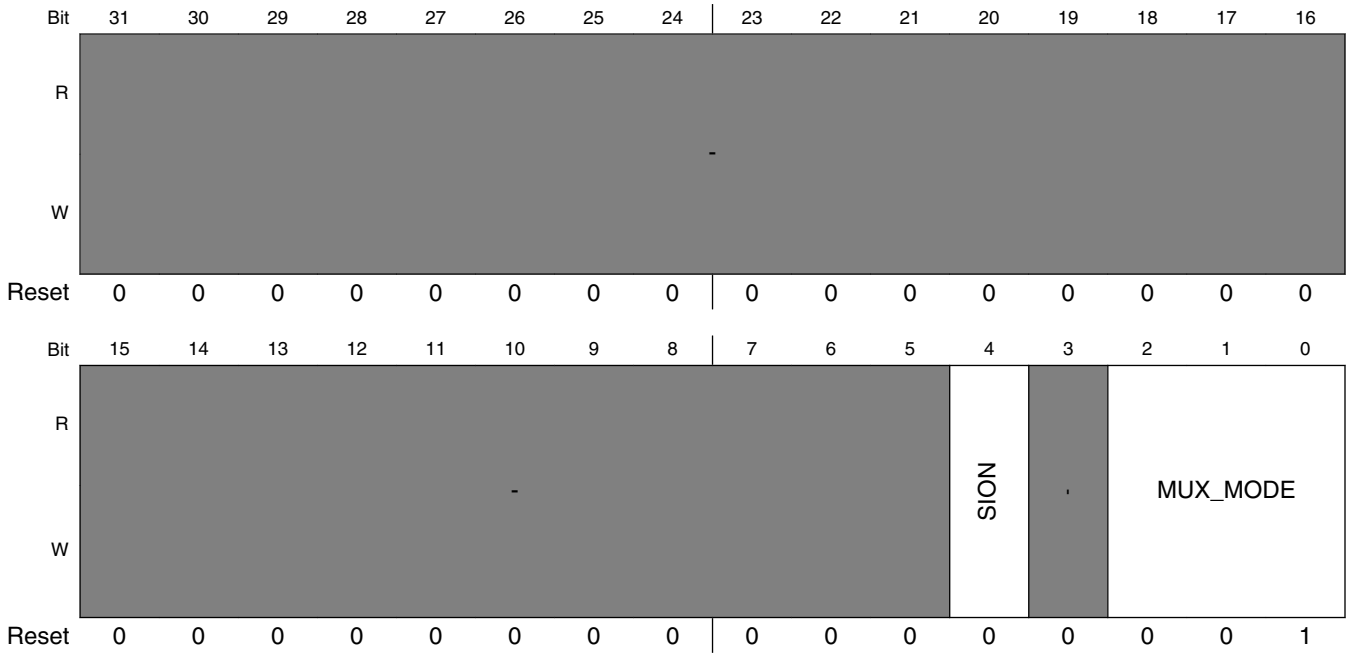


IOMUXC_SMUXC_PKC1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad KEY_COL1 0 Input Path is determined by functionality of the selected mux mode (regular)
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KEY_COL1. 000 Select mux mode: ALT0 mux port: COL[1] of instance: kpp 001 Select mux mode: ALT1 mux port: GPIO[2] of instance: gpio4 010 Select mux mode: ALT2 mux port: CEN[0] of instance: rawnand 110 Select mux mode: ALT6 mux port: CTI_TRIGOUT_ACK6 of instance: tigerp_platform_ne_32k_256k 111 Select mux mode: ALT7 mux port: RXACTIVE of instance: usbphy1

35.4.12 SW_MUX_CTL_PAD_KEY_ROW1 (IOMUXC_SMUXC_PKR1)

Address: IOMUXC_SMUXC_PKR1 is 53FA_8000h base + 2Ch offset = 53FA_802Ch

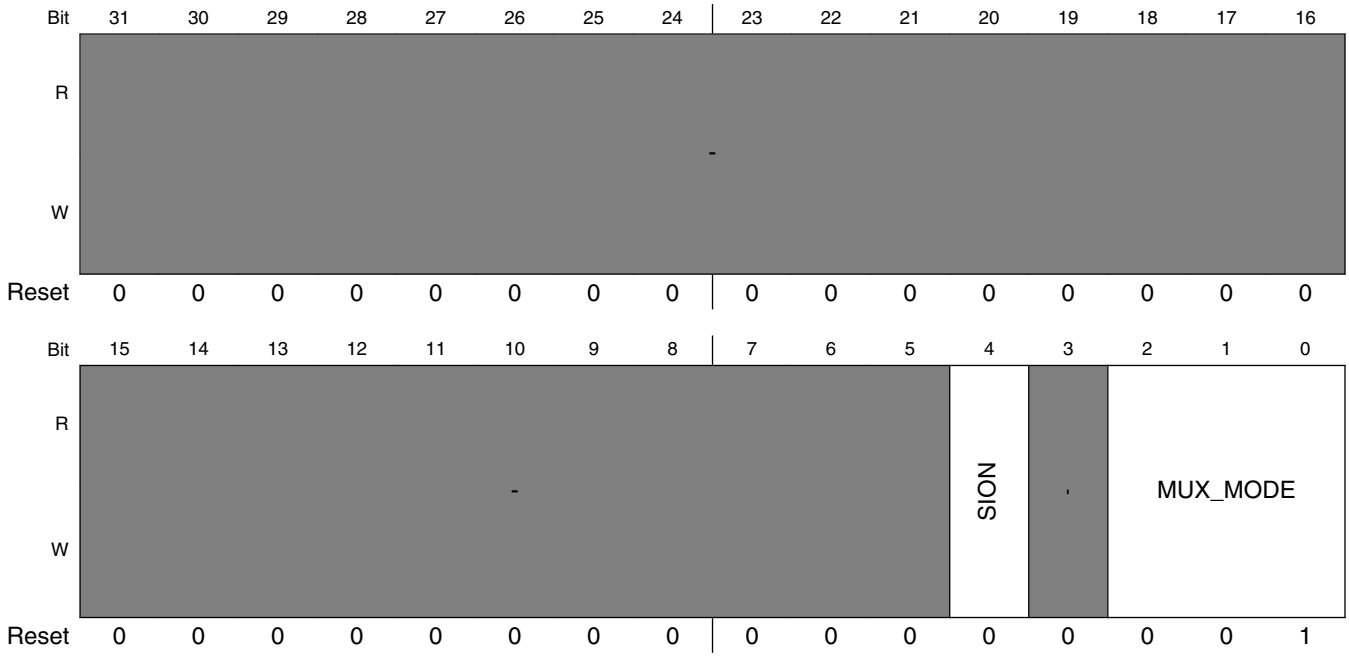


IOMUXC_SMUXC_PKR1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad KEY_ROW1 0 Input Path is determined by functionality of the selected mux mode (regular)
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KEY_ROW1. 000 Select mux mode: ALT0 mux port: ROW[1] of instance: kpp 001 Select mux mode: ALT1 mux port: GPIO[3] of instance: gpio4 010 Select mux mode: ALT2 mux port: CEN[1] of instance: rawnand 110 Select mux mode: ALT6 mux port: CTI_TRIGOUT_ACK7 of instance: tigerp_platform_ne_32k_256k 111 Select mux mode: ALT7 mux port: RXERROR of instance: usbphy1

35.4.13 SW_MUX_CTL_PAD_KEY_COL2 (IOMUXC_SMUXC_PKC2)

Address: IOMUXC_SMUXC_PKC2 is 53FA_8000h base + 30h offset = 53FA_8030h

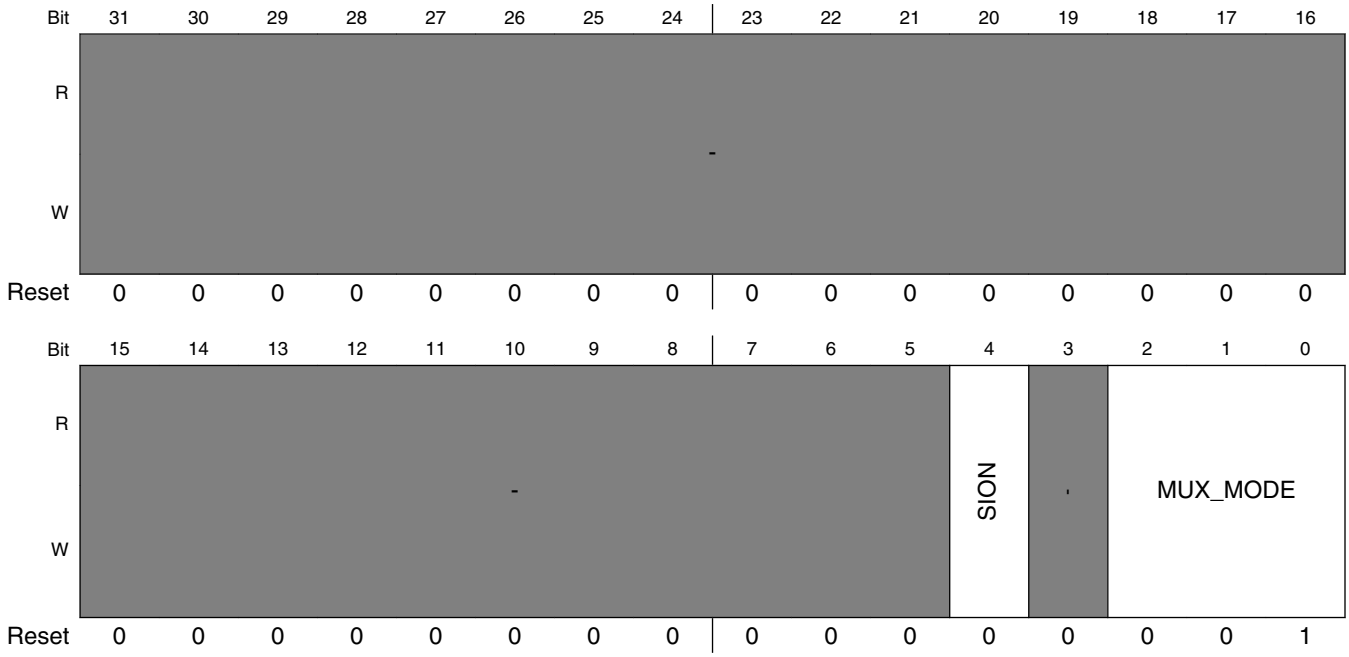


IOMUXC_SMUXC_PKC2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad KEY_COL2 0 Input Path is determined by functionality of the selected mux mode (regular)
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KEY_COL2. 000 Select mux mode: ALT0 mux port: COL[2] of instance: kpp. 001 Select mux mode: ALT1 mux port: GPIO[4] of instance: gpio4. 010 Select mux mode: ALT2 mux port: CEN[2] of instance: rawnand. 110 Select mux mode: ALT6 mux port: CTI_TRIGOUT6 of instance: tigerp_platform_ne_32k_256k. 111 Select mux mode: ALT7 mux port: SIECLOCK of instance: usbphy1.

35.4.14 SW_MUX_CTL_PAD_KEY_ROW2 (IOMUXC_SMUXC_PKR2)

Address: IOMUXC_SMUXC_PKR2 is 53FA_8000h base + 34h offset = 53FA_8034h

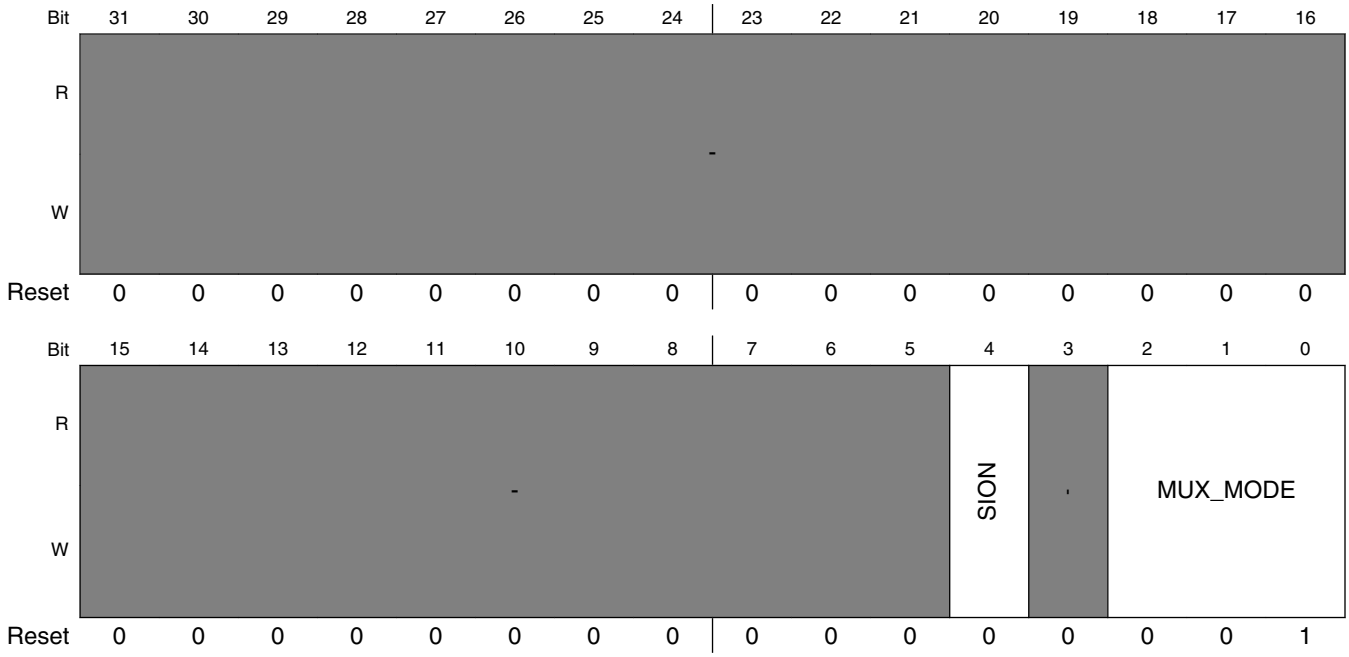


IOMUXC_SMUXC_PKR2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad KEY_ROW2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KEY_ROW2. 000 Select mux mode: ALT0 mux port: ROW[2] of instance: kpp. 001 Select mux mode: ALT1 mux port: GPIO[5] of instance: gpio4. 010 Select mux mode: ALT2 mux port: CEN[3] of instance: rawnand. 110 Select mux mode: ALT6 mux port: CTI_TRIGOUT7 of instance: tigerp_platform_ne_32k_256k. 111 Select mux mode: ALT7 mux port: LINESTATE[0] of instance: usbphy1.

35.4.15 SW_MUX_CTL_PAD_KEY_COL3 (IOMUXC_SMUXC_PKC3)

Address: IOMUXC_SMUXC_PKC3 is 53FA_8000h base + 38h offset = 53FA_8038h

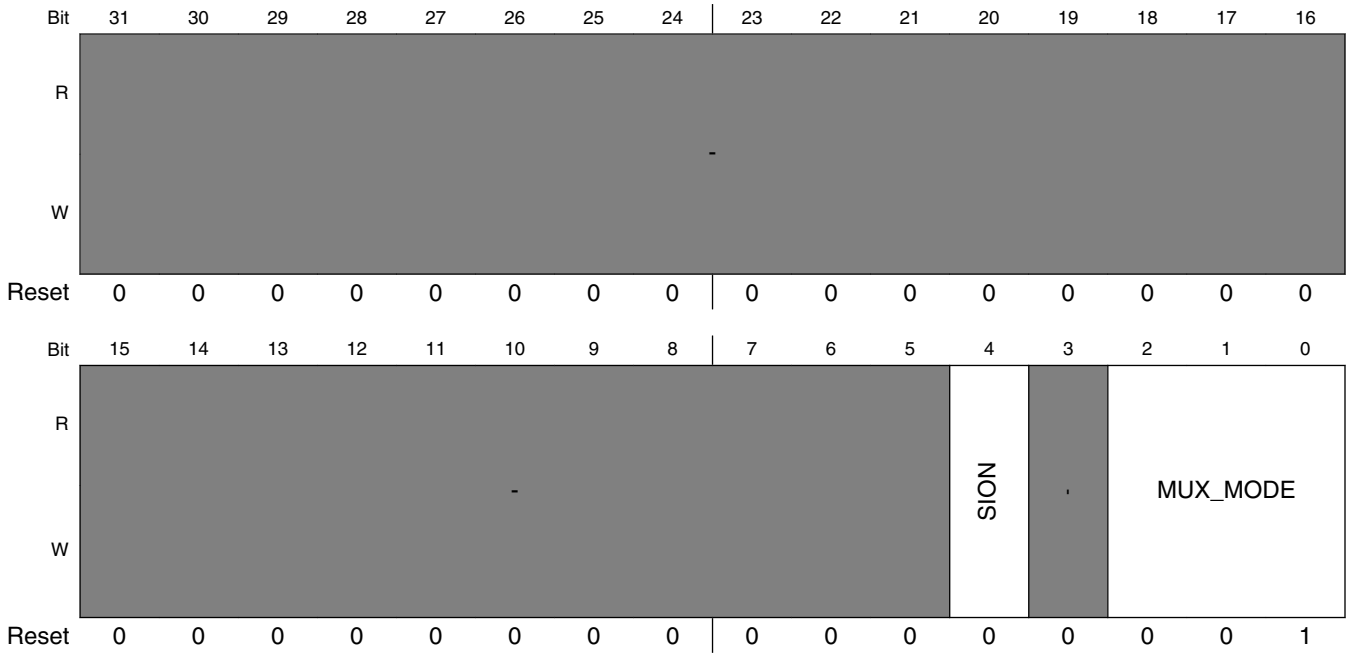


IOMUXC_SMUXC_PKC3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad KEY_COL3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KEY_COL3. **NOTE:** Pad KEY_COL3 is involved in Daisy Chain. • Config Register IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_SDMA_EVENTS_14_SELECT_INPUT for mode ALT6. 000 Select mux mode: ALT0 mux port: COL[3] of instance: kpp. 001 Select mux mode: ALT1 mux port: GPIO[6] of instance: gpio4. 010 Select mux mode: ALT2 mux port: READY0 of instance: rawnand. 110 Select mux mode: ALT6 mux port: SDMA_EXT_EVENT[0] of instance: sdma. 111 Select mux mode: ALT7 mux port: LINESTATE[1] of instance: usbphy1.

35.4.16 SW_MUX_CTL_PAD_KEY_ROW3 (IOMUXC_SMUXC_PKR3)

Address: IOMUXC_SMUXC_PKR3 is 53FA_8000h base + 3Ch offset = 53FA_803Ch



IOMUXC_SMUXC_PKR3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad KEY_ROW3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KEY_ROW3. **NOTE:** Pad KEY_ROW3 is involved in Daisy Chain. • Config Register IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_SDMA_EVENTS_15_SELECT_INPUT for mode ALT6. 000 Select mux mode: ALT0 mux port: ROW[3] of instance: kpp. 001 Select mux mode: ALT1 mux port: GPIO[7] of instance: gpio4. 010 Select mux mode: ALT2 mux port: DQS of instance: rawnand. 110 Select mux mode: ALT6 mux port: SDMA_EXT_EVENT[1] of instance: sdma. 111 Select mux mode: ALT7 mux port: VBUSVALID of instance: usbphy1.

35.4.17 SW_MUX_CTL_PAD_I2C1_SCL (IOMUXC_SMUXC_PI2C1_SCL)

Address: IOMUXC_SMUXC_PI2C1_SCL is 53FA_8000h base + 40h offset = 53FA_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

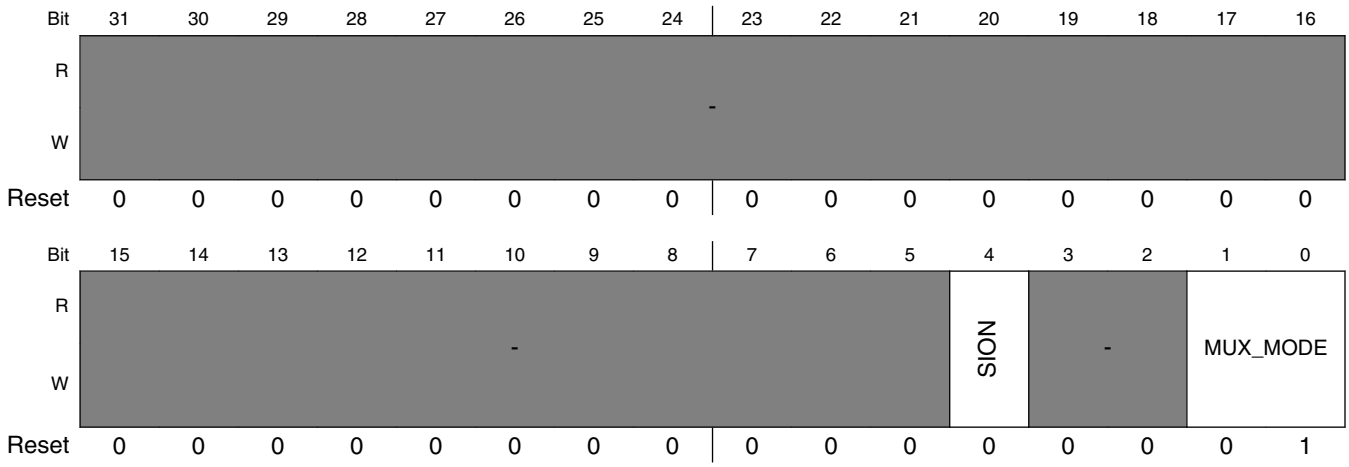
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PI2C1_SCL field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad I2C1_SCL. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: I2C1_SCL. **NOTE:** Pad I2C1_SCL is involved in Daisy Chain. • Config Register IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT2. 00 Select mux mode: ALT0 mux port: SCL of instance: i2c1. 01 Select mux mode: ALT1 mux port: GPIO[18] of instance: gpio6. 10 Select mux mode: ALT2 mux port: TXD_MUX of instance: uart2.

35.4.18 SW_MUX_CTL_PAD_I2C1_SDA (IOMUXC_SMUXC_PI2C1_SDA)

Address: IOMUXC_SMUXC_PI2C1_SDA is 53FA_8000h base + 44h offset = 53FA_8044h



IOMUXC_SMUXC_PI2C1_SDA field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad I2C1_SDA. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: I2C1_SDA. **NOTE:** Pad I2C1_SDA is involved in Daisy Chain. • Config Register IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT2. 00 Select mux mode: ALT0 mux port: SDA of instance: i2c1. 01 Select mux mode: ALT1 mux port: GPIO[19] of instance: gpio6. 10 Select mux mode: ALT2 mux port: RXD_MUX of instance: uart2.

35.4.19 SW_MUX_CTL_PAD_I2C2_SCL (IOMUXC_SMUXC_PI2C2_SCL)

Address: IOMUXC_SMUXC_PI2C2_SCL is 53FA_8000h base + 48h offset = 53FA_8048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

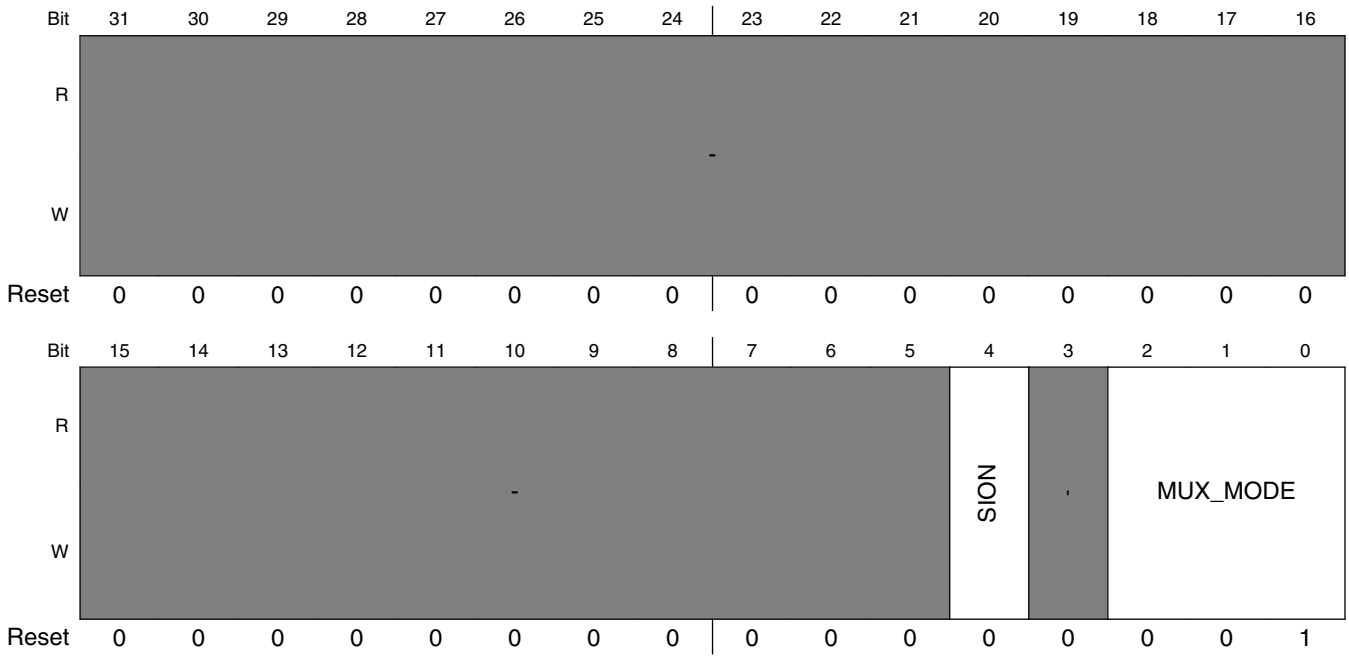
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PI2C2_SCL field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad I2C2_SCL. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: I2C2_SCL. **NOTE:** Pad I2C2_SCL is involved in Daisy Chain. • Config Register IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SCL of instance: i2c2. 001 Select mux mode: ALT1 mux port: GPIO[20] of instance: gpio6. 010 Select mux mode: ALT2 mux port: CTS of instance: uart2.

35.4.20 SW_MUX_CTL_PAD_I2C2_SDA (IOMUXC_SMUXC_PI2C2_SDA)

Address: IOMUXC_SMUXC_PI2C2_SDA is 53FA_8000h base + 4Ch offset = 53FA_804Ch

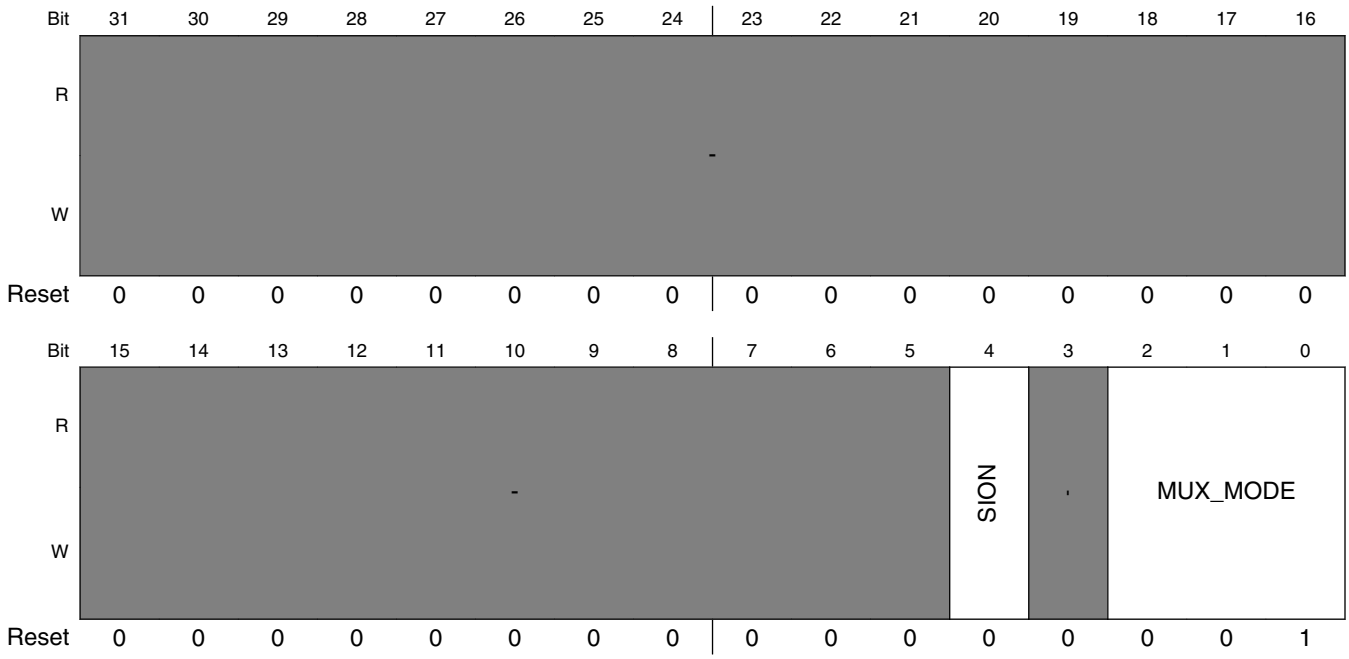


IOMUXC_SMUXC_PI2C2_SDA field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad I2C2_SDA. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: I2C2_SDA. **NOTE:** Pad I2C2_SDA is involved in Daisy Chain. • Config Register IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDA of instance: i2c2. 001 Select mux mode: ALT1 mux port: GPIO[21] of instance: gpio6. 010 Select mux mode: ALT2 mux port: RTS of instance: uart2.

35.4.21 SW_MUX_CTL_PAD_I2C3_SCL (IOMUXC_SMUXC_PI2C3_SCL)

Address: IOMUXC_SMUXC_PI2C3_SCL is 53FA_8000h base + 50h offset = 53FA_8050h

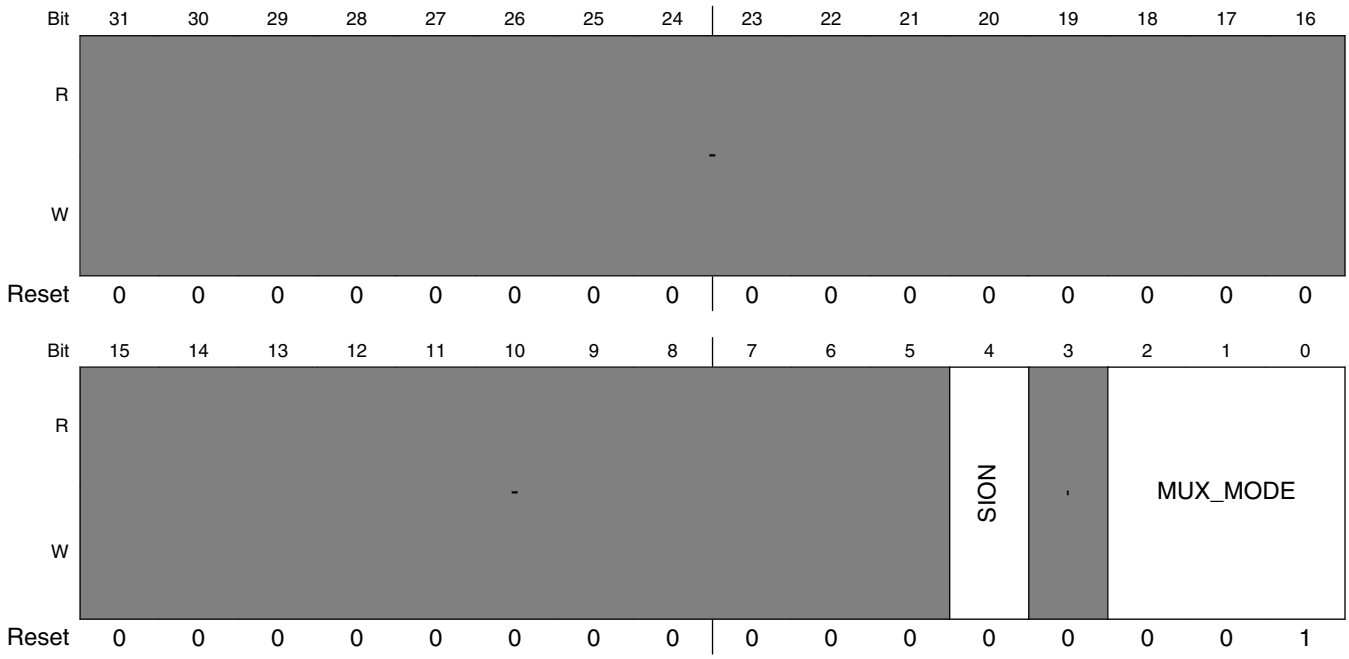


IOMUXC_SMUXC_PI2C3_SCL field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad I2C3_SCL. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: I2C3_SCL. **NOTE:** Pad I2C3_SCL is involved in Daisy Chain. • Config Register IOMUXC_USBOH1_IPP_IND_OTG_OC_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: SCL of instance: i2c3. 001 Select mux mode: ALT1 mux port: GPIO[22] of instance: gpio6. 010 Select mux mode: ALT2 mux port: MDC of instance: fec. 011 Select mux mode: ALT3 mux port: PMIC_RDY of instance: gpc. 101 Select mux mode: ALT5 mux port: CAPIN1 of instance: gpt. 110 Select mux mode: ALT6 mux port: OBSRV_INT_OUT0 of instance: observe_mux. 111 Select mux mode: ALT7 mux port: USBOTG_OC of instance: usboh1.

35.4.22 SW_MUX_CTL_PAD_I2C3_SDA (IOMUXC_SMUXC_PI2C3_SDA)

Address: IOMUXC_SMUXC_PI2C3_SDA is 53FA_8000h base + 54h offset = 53FA_8054h

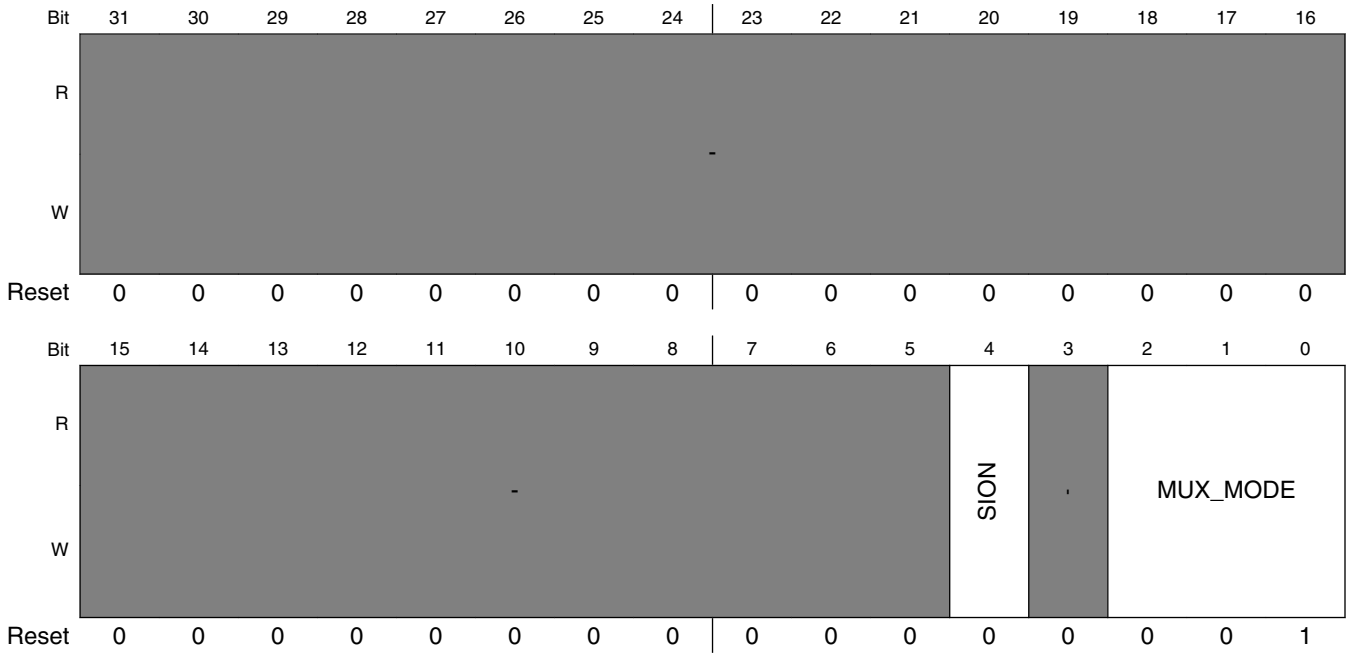


IOMUXC_SMUXC_PI2C3_SDA field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad I2C3_SDA. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: I2C3_SDA. **NOTE:** Pad I2C3_SDA is involved in Daisy Chain. • Config Register IOMUXC_FEC_FEC_MDI_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDA of instance: i2c3. 001 Select mux mode: ALT1 mux port: GPIO[23] of instance: gpio6. 010 Select mux mode: ALT2 mux port: MDIO of instance: fec. 011 Select mux mode: ALT3 mux port: PWRFAIL_INT of instance: tzic. 100 Select mux mode: ALT4 mux port: ALARM_DEB of instance: srtc. 101 Select mux mode: ALT5 mux port: CAPIN2 of instance: gpt. 110 Select mux mode: ALT6 mux port: OBSRV_INT_OUT1 of instance: observe_mux. 111 Select mux mode: ALT7 mux port: USBOTG_PWR of instance: usboh1.

35.4.23 SW_MUX_CTL_PAD_PWM1 (IOMUXC_SMUXC_PPWM1)

Address: IOMUXC_SMUXC_PPWM1 is 53FA_8000h base + 58h offset = 53FA_8058h

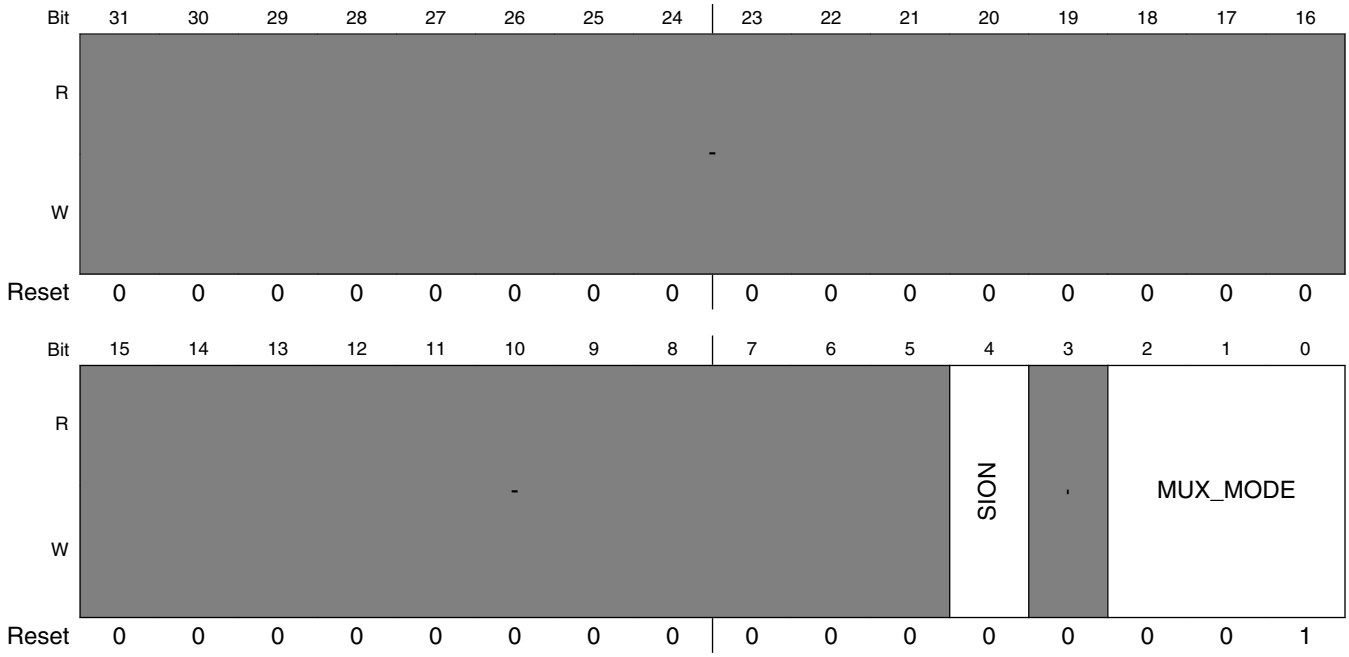


IOMUXC_SMUXC_PPWM1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad PWM1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: PWM1. **NOTE:** Pad PWM1 is involved in Daisy Chain. • Config Register IOMUXC_USBOH1_IPP_IND_OTG_OC_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: PWMO of instance: pwm1. 001 Select mux mode: ALT1 mux port: GPIO[24] of instance: gpio6. 010 Select mux mode: ALT2 mux port: USBOTG_OC of instance: usboh1. 101 Select mux mode: ALT5 mux port: CMPOUT1 of instance: gpt. 110 Select mux mode: ALT6 mux port: OBSRV_INT_OUT2 of instance: observe_mux. 111 Select mux mode: ALT7 mux port: FAIL of instance: sjc.

35.4.24 SW_MUX_CTL_PAD_PWM2 (IOMUXC_SMUXC_PPWM2)

Address: IOMUXC_SMUXC_PPWM2 is 53FA_8000h base + 5Ch offset = 53FA_805Ch

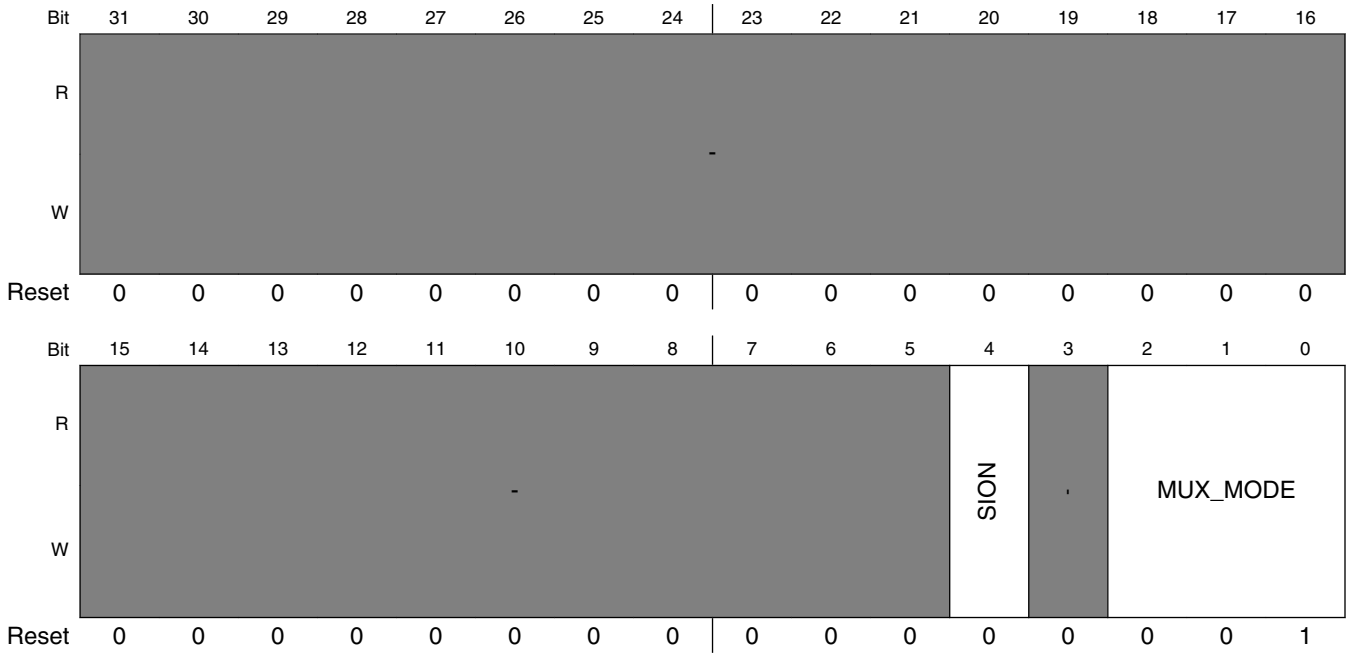


IOMUXC_SMUXC_PPWM2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad PWM2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: PWM2. 000 Select mux mode: ALT0 mux port: PWMO of instance: pwm2. 001 Select mux mode: ALT1 mux port: GPIO[25] of instance: gpio6. 010 Select mux mode: ALT2 mux port: USBOTG_PWR of instance: usboh1. 101 Select mux mode: ALT5 mux port: CMPOUT2 of instance: gpt. 110 Select mux mode: ALT6 mux port: OBSRV_INT_OUT3 of instance: observe_mux. 111 Select mux mode: ALT7 mux port: ANY_PU_RST of instance: src.

35.4.25 SW_MUX_CTL_PAD_OWIRE (IOMUXC_SMUXC_POWIRE)

Address: IOMUXC_SMUXC_POWIRE is 53FA_8000h base + 60h offset = 53FA_8060h

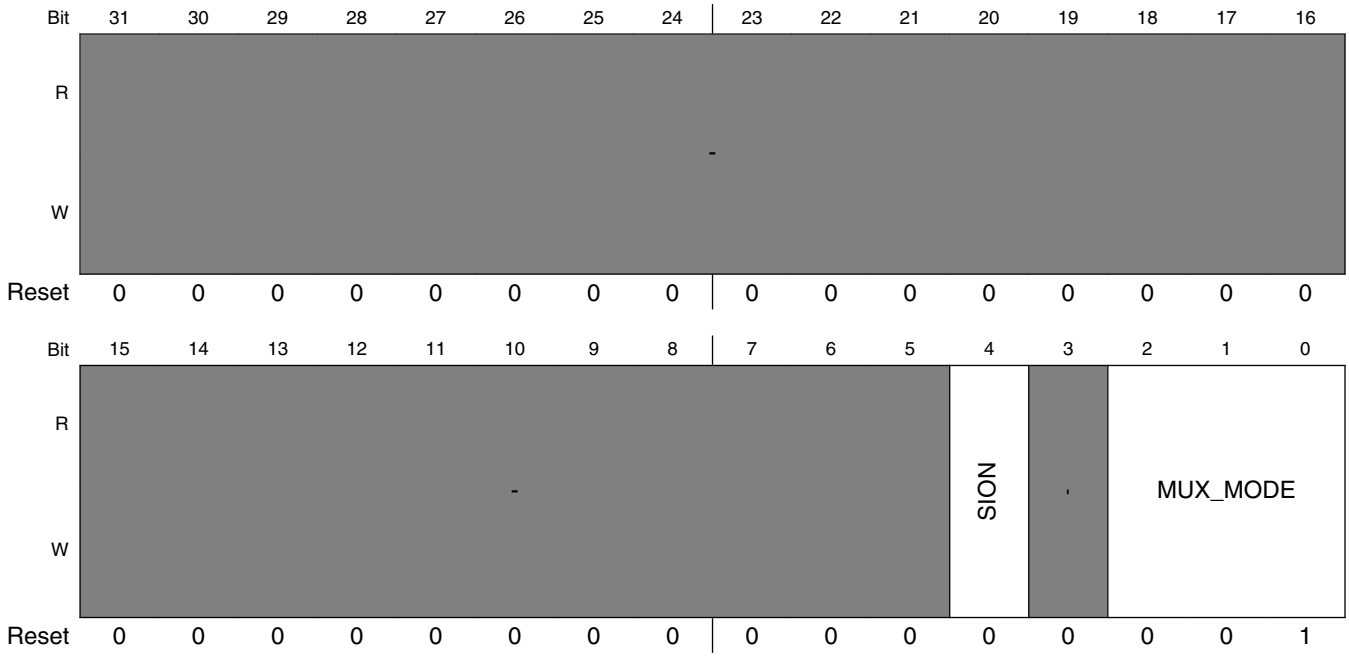


IOMUXC_SMUXC_POWIRE field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad OWIRE. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: OWIRE. 000 Select mux mode: ALT0 mux port: LINE of instance: owire. 001 Select mux mode: ALT1 mux port: GPIO[26] of instance: gpio6. 010 Select mux mode: ALT2 mux port: USBH1_OC of instance: usboh1. 011 Select mux mode: ALT3 mux port: SSI_EXT1_CLK of instance: ccm. 100 Select mux mode: ALT4 mux port: PWRIRQ of instance: epdc. 101 Select mux mode: ALT5 mux port: CMPOUT3 of instance: gpt. 110 Select mux mode: ALT6 mux port: OBSRV_INT_OUT4 of instance: observe_mux. 111 Select mux mode: ALT7 mux port: JTAG_ACT of instance: sjc.

35.4.26 SW_MUX_CTL_PAD_EPITO (IOMUXC_SMUXC_PEPITO)

Address: IOMUXC_SMUXC_PEPITO is 53FA_8000h base + 64h offset = 53FA_8064h

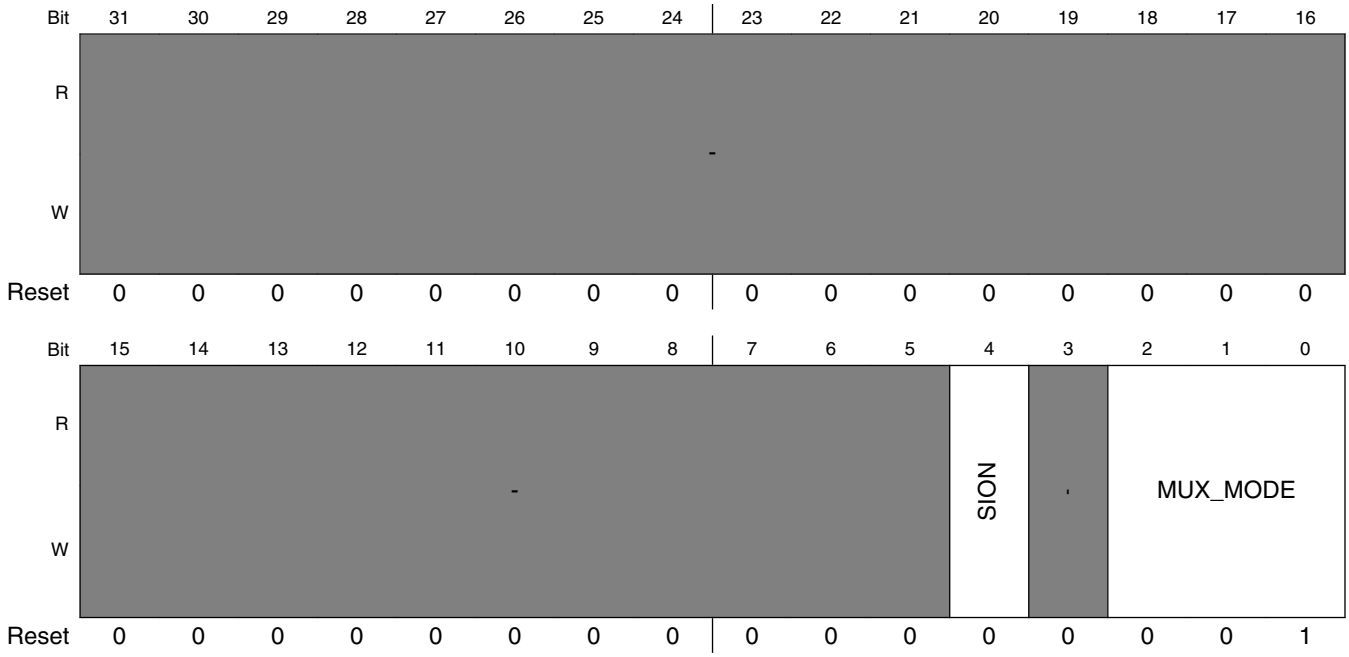


IOMUXC_SMUXC_PEPITO field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPITO. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: EPITO. 000 Select mux mode: ALT0 mux port: EPITO of instance: epit1. 001 Select mux mode: ALT1 mux port: GPIO[27] of instance: gpio6. 010 Select mux mode: ALT2 mux port: USBH1_PWR of instance: usboh1. 011 Select mux mode: ALT3 mux port: SSI_EXT2_CLK of instance: ccm. 100 Select mux mode: ALT4 mux port: TOG_EN of instance: dpllip1. 101 Select mux mode: ALT5 mux port: CLKIN of instance: gpt. 110 Select mux mode: ALT6 mux port: PMU_IRQ_B of instance: tigerp_platform_ne_32k_256k. 111 Select mux mode: ALT7 mux port: DE_B of instance: sjc.

35.4.27 SW_MUX_CTL_PAD_WDOG (IOMUXC_SMUXC_PWDOG)

Address: IOMUXC_SMUXC_PWDOG is 53FA_8000h base + 68h offset = 53FA_8068h

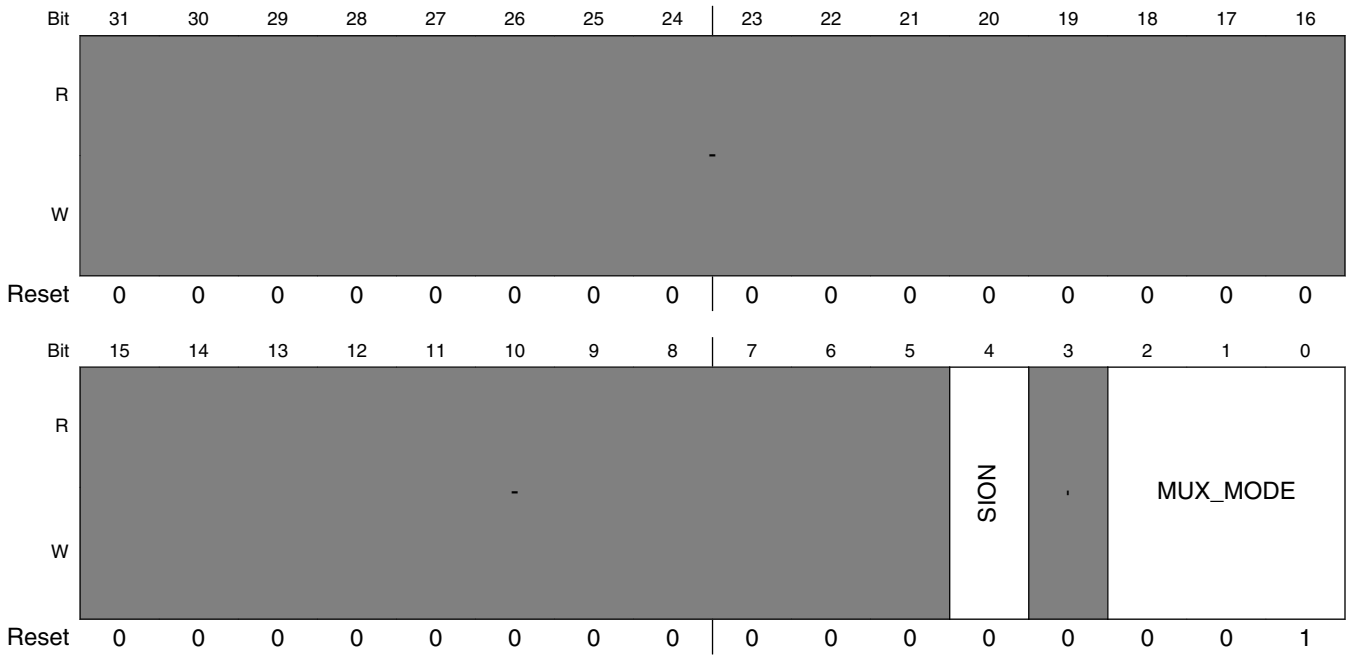


IOMUXC_SMUXC_PWDOG field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad WDOG. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: WDOG. 000 Select mux mode: ALT0 mux port: WDOG_B of instance: wdog1. 001 Select mux mode: ALT1 mux port: GPIO[28] of instance: gpio6. 010 Select mux mode: ALT2 mux port: WDOG_RST_B_DEB of instance: wdog1. 110 Select mux mode: ALT6 mux port: XTAL32K of instance: ccm. 111 Select mux mode: ALT7 mux port: DONE of instance: sjc.

35.4.28 SW_MUX_CTL_PAD_SSI_TXFS (IOMUXC_SMUXC_PSSI_TXFS)

Address: IOMUXC_SMUXC_PSSI_TXFS is 53FA_8000h base + 6Ch offset = 53FA_806Ch

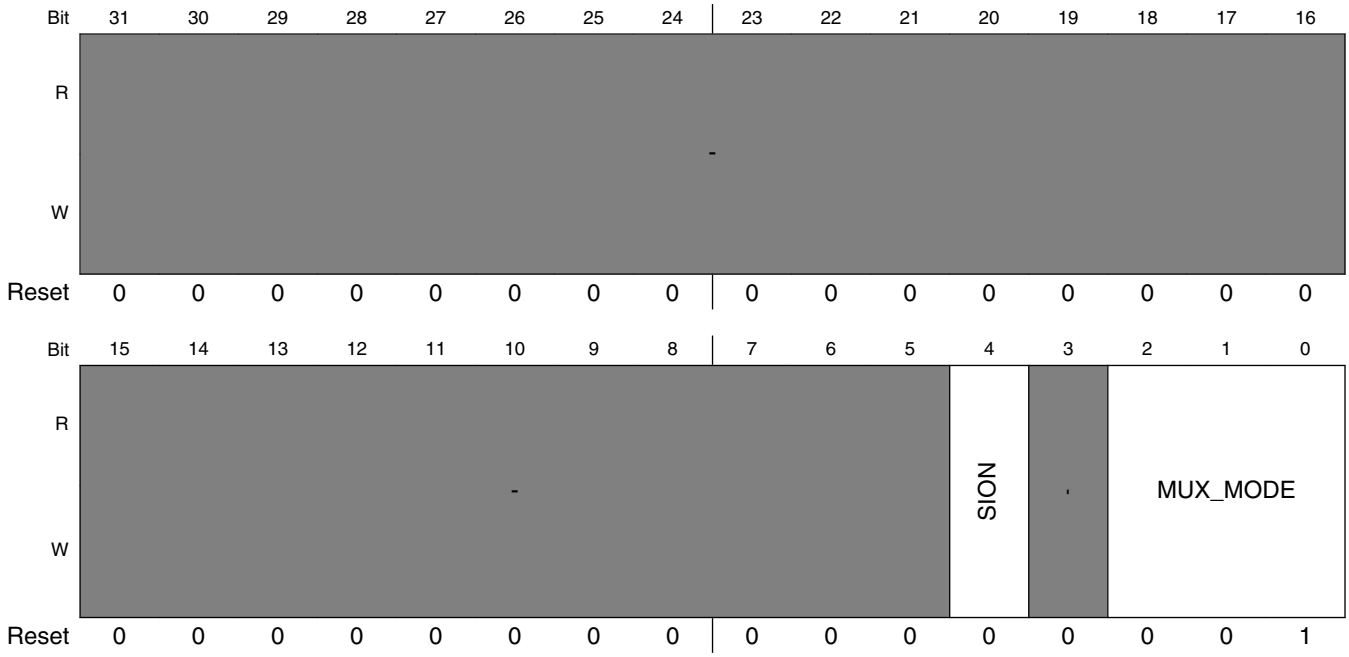


IOMUXC_SMUXC_PSSI_TXFS field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SSI_TXFS. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SSI_TXFS. 000 Select mux mode: ALT0 mux port: AUD3_TXFS of instance: audmux. 001 Select mux mode: ALT1 mux port: GPIO[0] of instance: gpio6. 110 Select mux mode: ALT6 mux port: BT_FUSE_RSV[1] of instance: src. 111 Select mux mode: ALT7 mux port: DATAOUT[8] of instance: usbphy1.

35.4.29 SW_MUX_CTL_PAD_SSI_TXC (IOMUXC_SMUXC_PSSI_TXC)

Address: IOMUXC_SMUXC_PSSI_TXC is 53FA_8000h base + 70h offset = 53FA_8070h

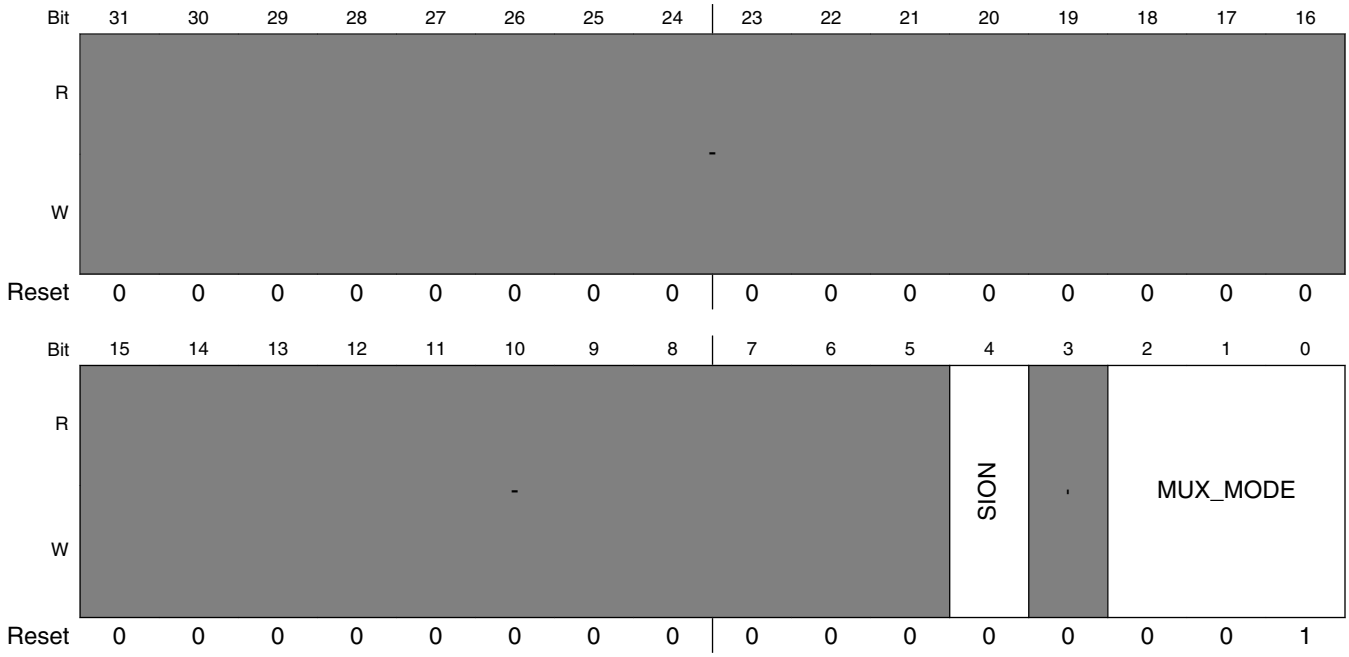


IOMUXC_SMUXC_PSSI_TXC field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SSI_TXC. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SSI_TXC. 000 Select mux mode: ALT0 mux port: AUD3_TXC of instance: audmux. 001 Select mux mode: ALT1 mux port: GPIO[1] of instance: gpio6. 110 Select mux mode: ALT6 mux port: BT_FUSE_RSV[0] of instance: src. 111 Select mux mode: ALT7 mux port: DATAOUT[9] of instance: usbphy1.

35.4.30 SW_MUX_CTL_PAD_SSI_TXD (IOMUXC_SMUXC_PSSI_TXD)

Address: IOMUXC_SMUXC_PSSI_TXD is 53FA_8000h base + 74h offset = 53FA_8074h



IOMUXC_SMUXC_PSSI_TXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SSI_TXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SSI_TXD. **NOTE:** Pad SSI_TXD is involved in Daisy Chain. • Config Register IOMUXC_CSPI_IPP_IND_DATAREADY_B_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: AUD3_TXD of instance: audmux. 001 Select mux mode: ALT1 mux port: GPIO[2] of instance: gpio6. 100 Select mux mode: ALT4 mux port: RDY of instance: cspi. 111 Select mux mode: ALT7 mux port: DATAOUT[10] of instance: usbphy1.

35.4.31 SW_MUX_CTL_PAD_SSI_RXD (IOMUXC_SMUXC_PSSI_RXD)

Address: IOMUXC_SMUXC_PSSI_RXD is 53FA_8000h base + 78h offset = 53FA_8078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

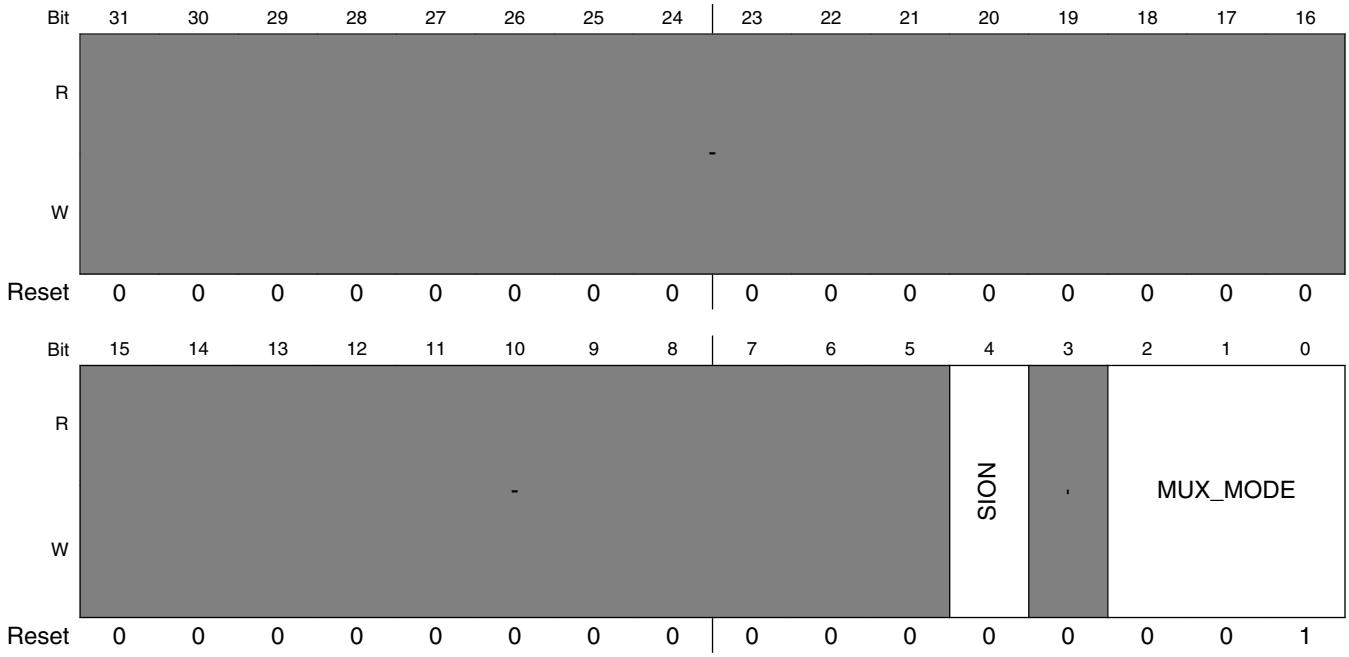
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSSI_RXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SSI_RXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SSI_RXD. **NOTE:** Pad SSI_RXD is involved in Daisy Chain. • Config Register IOMUXC_CSPI_IPP_IND_SS3_B_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: AUD3_RXD of instance: audmux. 001 Select mux mode: ALT1 mux port: GPIO[3] of instance: gpio6. 100 Select mux mode: ALT4 mux port: SS3 of instance: cspi. 111 Select mux mode: ALT7 mux port: DATAOUT[11] of instance: usbphy1.

35.4.32 SW_MUX_CTL_PAD_SSI_RXF (IOMUXC_SMUXC_PSSI_RXF)

Address: IOMUXC_SMUXC_PSSI_RXF is 53FA_8000h base + 7Ch offset = 53FA_807Ch



IOMUXC_SMUXC_PSSI_RXF field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SSI_RXFS. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SSI_RXFS. **NOTE:** Pad SSI_RXFS is involved in Daisy Chain. • Config Register IOMUXC_CSPI_IPP_IND_SS2_B_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_FEC_FEC_COL_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_6_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: AUD3_RXFS of instance: audmux. 001 Select mux mode: ALT1 mux port: GPIO[4] of instance: gpio6. 010 Select mux mode: ALT2 mux port: TXD_MUX of instance: uart5. 011 Select mux mode: ALT3 mux port: D[6] of instance: weimv2. 100 Select mux mode: ALT4 mux port: SS2 of instance: cspi. 101 Select mux mode: ALT5 mux port: COL of instance: fec.

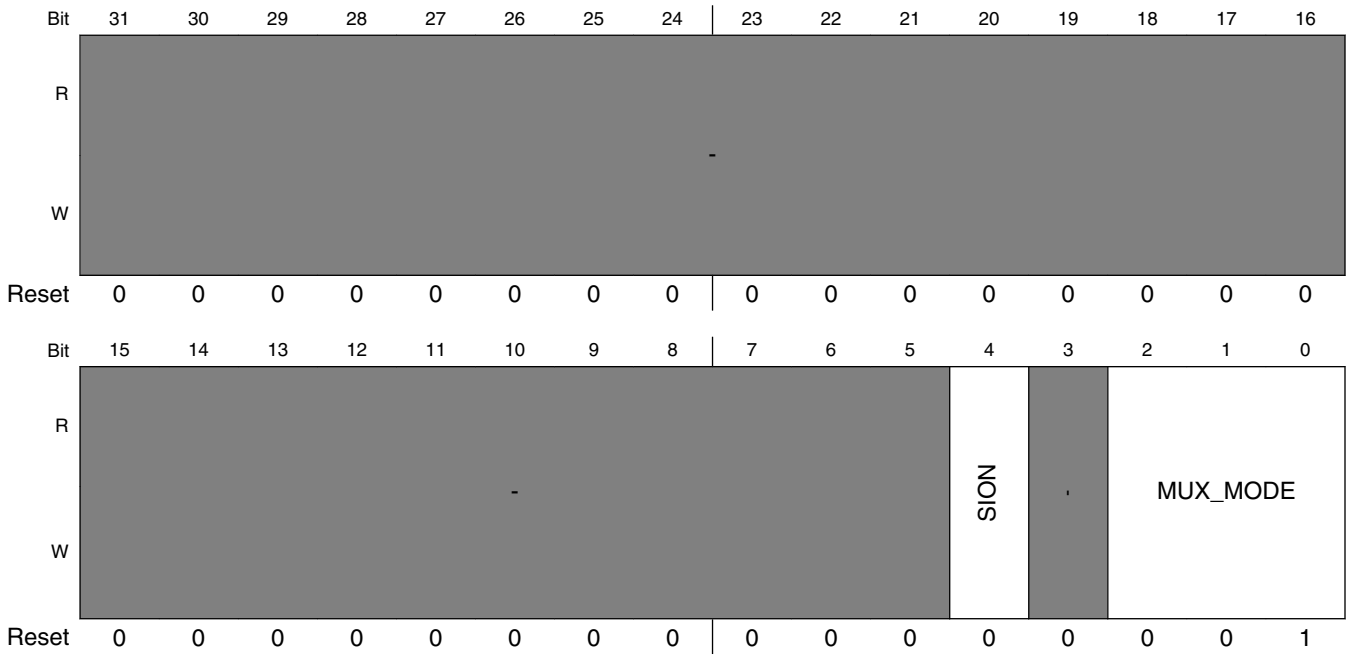
Table continues on the next page...

IOMUXC_SMUXC_PSSI_RXF field descriptions (continued)

Field	Description
	110 Select mux mode: ALT6 mux port: MDC of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[12] of instance: usbphy1.

35.4.33 SW_MUX_CTL_PAD_SSI_RXC (IOMUXC_SMUXC_PSSI_RXC)

Address: IOMUXC_SMUXC_PSSI_RXC is 53FA_8000h base + 80h offset = 53FA_8080h



IOMUXC_SMUXC_PSSI_RXC field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SSI_RXC. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SSI_RXC. **NOTE:** Pad SSI_RXC is involved in Daisy Chain. • Config Register IOMUXC_CSPI_IPP_IND_SS1_B_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_FEC_FEC_MDI_SELECT_INPUT for mode ALT6. • Config Register IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_7_SELECT_INPUT for mode ALT3.

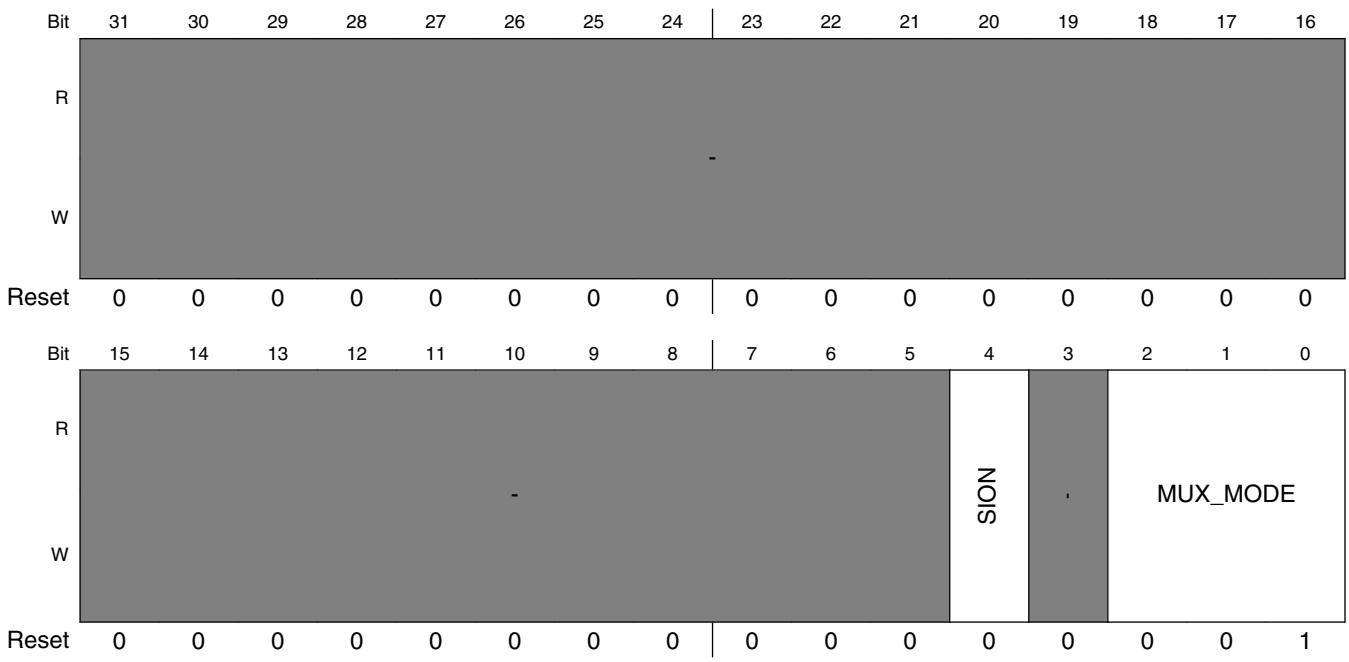
Table continues on the next page...

IOMUXC_SMUXC_PSSI_RXC field descriptions (continued)

Field	Description
	000 Select mux mode: ALT0 mux port: AUD3_RXC of instance: audmux. 001 Select mux mode: ALT1 mux port: GPIO[5] of instance: gpio6. 010 Select mux mode: ALT2 mux port: RXD_MUX of instance: uart5. 011 Select mux mode: ALT3 mux port: D[7] of instance: weimv2. 100 Select mux mode: ALT4 mux port: SS1 of instance: cspi. 101 Select mux mode: ALT5 mux port: RX_CLK of instance: fec. 110 Select mux mode: ALT6 mux port: MDIO of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[13] of instance: usbphy1.

35.4.34 SW_MUX_CTL_PAD_UART1_TXD (IOMUXC_SMUXC_PUART1_TXD)

Address: IOMUXC_SMUXC_PUART1_TXD is 53FA_8000h base + 84h offset = 53FA_8084h



IOMUXC_SMUXC_PUART1_TXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART1_TXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved

Table continues on the next page...

IOMUXC_SMUXC_PUART1_TXD field descriptions (continued)

Field	Description
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: UART1_TXD. **NOTE:** Pad UART1_TXD is involved in Daisy Chain. • Config Register IOMUXC_UART1_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: TXD_MUX of instance: uart1. 001 Select mux mode: ALT1 mux port: GPIO[6] of instance: gpio6. 111 Select mux mode: ALT7 mux port: DATAOUT[14] of instance: usbphy1.

35.4.35 SW_MUX_CTL_PAD_UART1_RXD (IOMUXC_SMUXC_PUART1_RXD)

Address: IOMUXC_SMUXC_PUART1_RXD is 53FA_8000h base + 88h offset = 53FA_8088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PUART1_RXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART1_RXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved

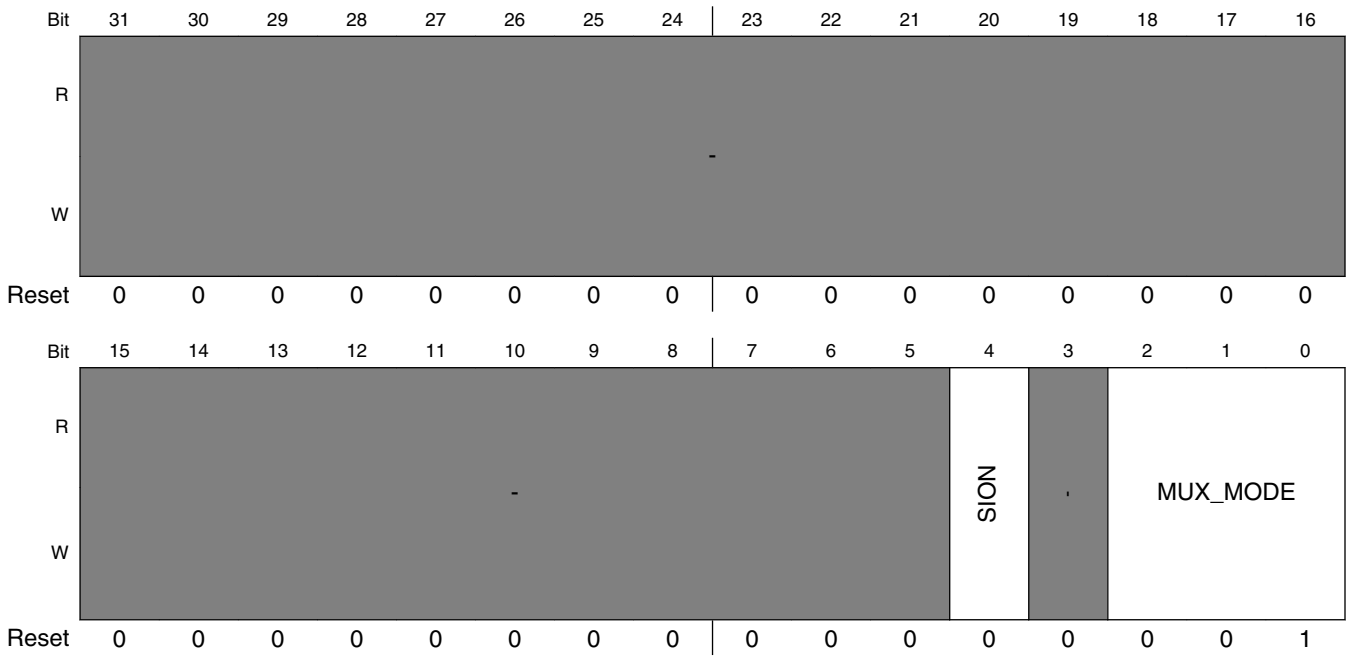
Table continues on the next page...

IOMUXC_SMUXC_PUART1_RXD field descriptions (continued)

Field	Description
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: UART1_RXD. **NOTE:** Pad UART1_RXD is involved in Daisy Chain. • Config Register IOMUXC_UART1_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: RXD_MUX of instance: uart1. 001 Select mux mode: ALT1 mux port: GPIO[7] of instance: gpio6. 111 Select mux mode: ALT7 mux port: DATAOUT[15] of instance: usbphy1.

35.4.36 SW_MUX_CTL_PAD_UART1_CTS (IOMUXC_SMUXC_PUART1_CTS)

Address: IOMUXC_SMUXC_PUART1_CTS is 53FA_8000h base + 8Ch offset = 53FA_808Ch



IOMUXC_SMUXC_PUART1_CTS field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART1_CTS. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved

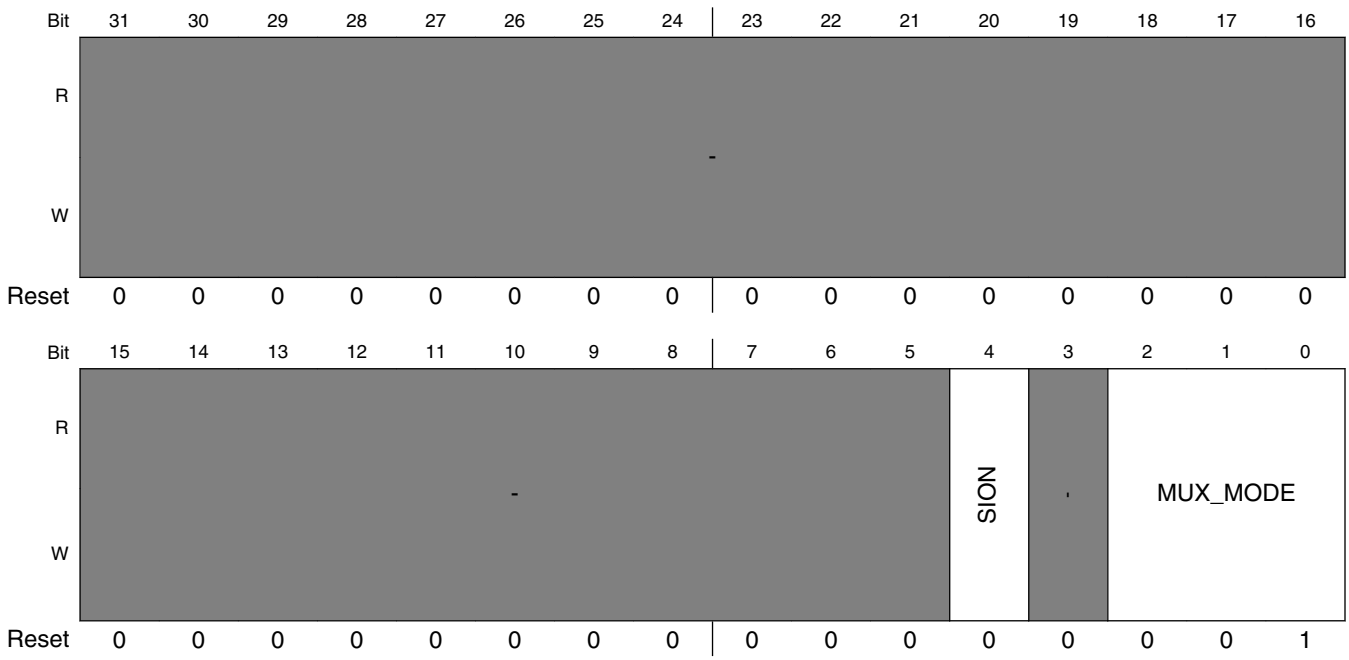
Table continues on the next page...

IOMUXC_SMUXC_PUART1_CTS field descriptions (continued)

Field	Description
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: UART1_CTS. **NOTE:** Pad UART1_CTS is involved in Daisy Chain. • Config Register IOMUXC_ESDHC4_IPP_CMD_IN_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_ESDHC4_IPP_DAT4_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_UART1_IPP_UART_RTS_B_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: CTS of instance: uart1. 001 Select mux mode: ALT1 mux port: GPIO[8] of instance: gpio6. 010 Select mux mode: ALT2 mux port: TXD_MUX of instance: uart5. 100 Select mux mode: ALT4 mux port: DAT4 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: CMD of instance: esdhc4. 111 Select mux mode: ALT7 mux port: DATAOUT[8] of instance: usbphy2.

35.4.37 SW_MUX_CTL_PAD_UART1_RTS (IOMUXC_SMUXC_PUART1_RTS)

Address: IOMUXC_SMUXC_PUART1_RTS is 53FA_8000h base + 90h offset = 53FA_8090h



IOMUXC_SMUXC_PUART1_RTS field descriptions

Field	Description
31–5 -	Reserved

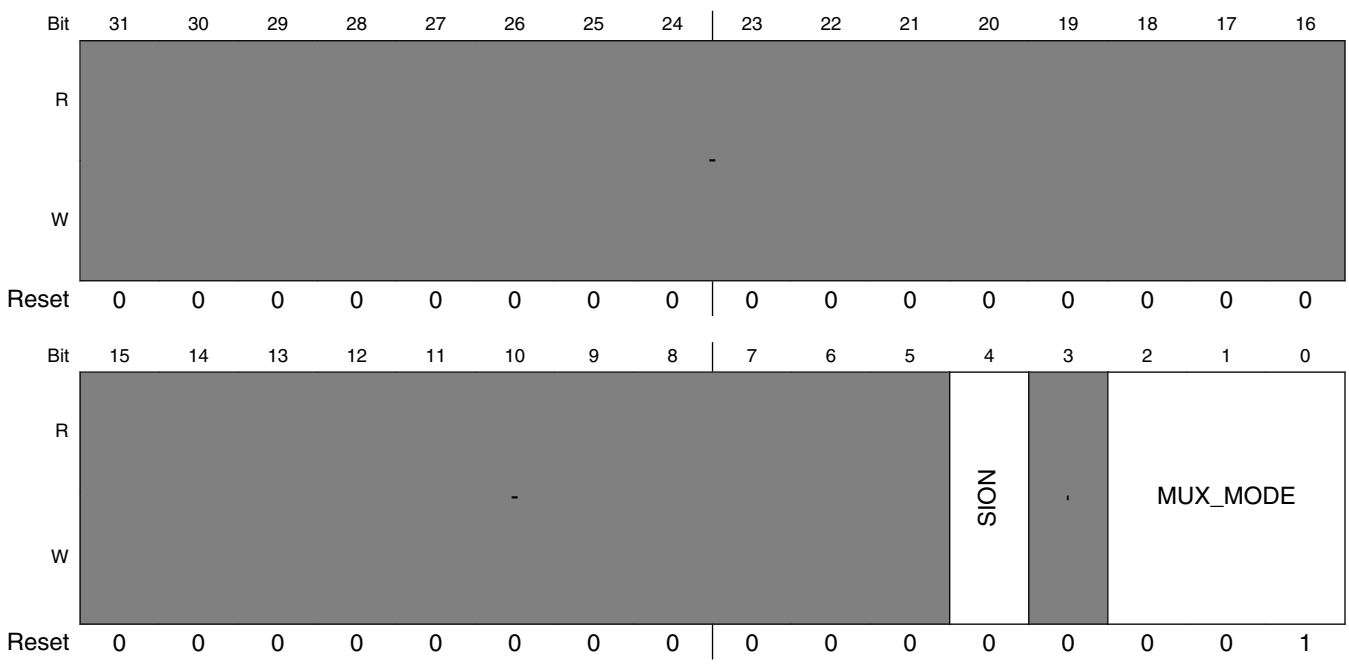
Table continues on the next page...

IOMUXC_SMUXC_PUART1_RTS field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART1_RTS. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: UART1_RTS. **NOTE:** Pad UART1_RTS is involved in Daisy Chain. • Config Register IOMUXC_ESDHC4_IPP_CARD_CLK_IN_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_ESDHC4_IPP_DAT5_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_UART1_IPP_UART_RTS_B_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: RTS of instance: uart1. 001 Select mux mode: ALT1 mux port: GPIO[9] of instance: gpio6. 010 Select mux mode: ALT2 mux port: RXD_MUX of instance: uart5. 100 Select mux mode: ALT4 mux port: DAT5 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: CLK of instance: esdhc4. 111 Select mux mode: ALT7 mux port: DATAOUT[9] of instance: usbphy2.

35.4.38 SW_MUX_CTL_PAD_UART2_TXD (IOMUXC_SMUXC_PUART2_TXD)

Address: IOMUXC_SMUXC_PUART2_TXD is 53FA_8000h base + 94h offset = 53FA_8094h

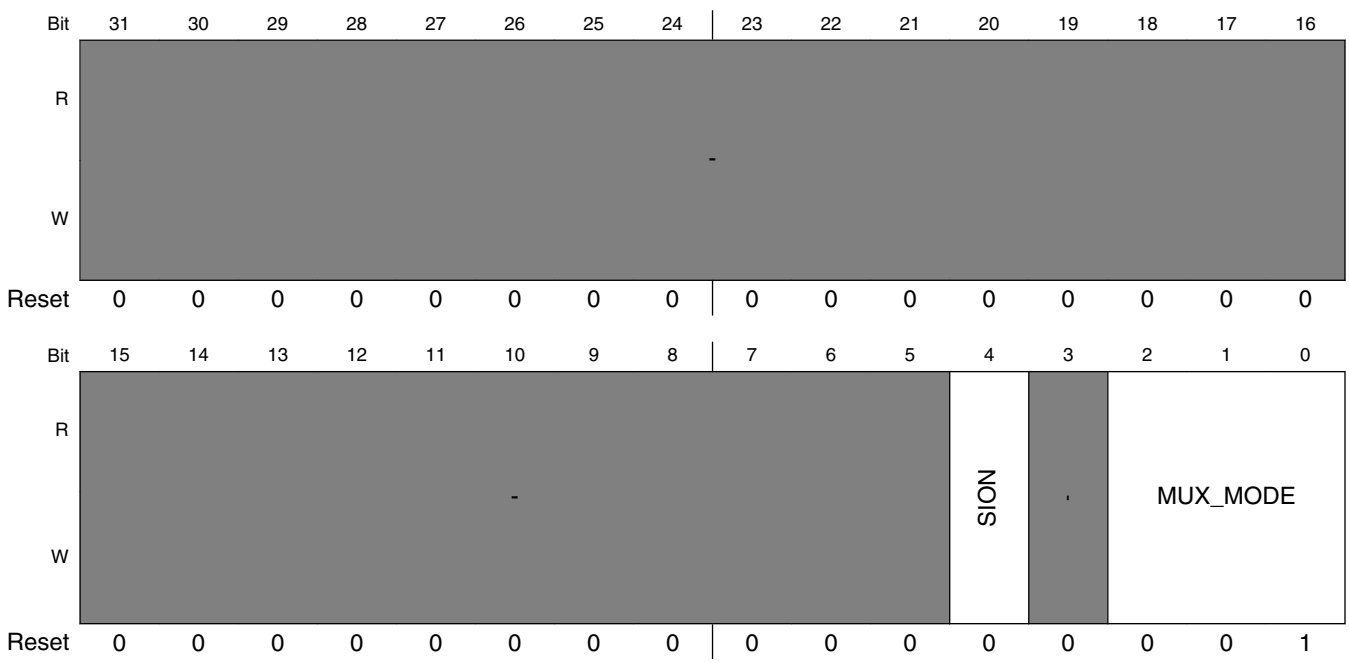


IOMUXC_SMUXC_PUART2_TXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART2_TXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: UART2_TXD. **NOTE:** Pad UART2_TXD is involved in Daisy Chain. • Config Register IOMUXC_ESDHC4_IPP_DAT4_IN_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_ESDHC4_IPP_DAT6_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: TXD_MUX of instance: uart2. 001 Select mux mode: ALT1 mux port: GPIO[10] of instance: gpio6. 100 Select mux mode: ALT4 mux port: DAT6 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: DAT4 of instance: esdhc4. 111 Select mux mode: ALT7 mux port: DATAOUT[10] of instance: usbphy2.

35.4.39 SW_MUX_CTL_PAD_UART2_RXD (IOMUXC_SMUXC_PUART2_RXD)

Address: IOMUXC_SMUXC_PUART2_RXD is 53FA_8000h base + 98h offset = 53FA_8098h



IOMUXC_SMUXC_PUART2_RXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART2_RXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: UART2_RXD. **NOTE:** Pad UART2_RXD is involved in Daisy Chain. • Config Register IOMUXC_ESDHC4_IPP_DAT5_IN_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_ESDHC4_IPP_DAT7_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: RXD_MUX of instance: uart2. 001 Select mux mode: ALT1 mux port: GPIO[11] of instance: gpio6. 100 Select mux mode: ALT4 mux port: DAT7 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: DAT5 of instance: esdhc4. 111 Select mux mode: ALT7 mux port: DATAOUT[11] of instance: usbphy2.

35.4.40 SW_MUX_CTL_PAD_UART2_CTS (IOMUXC_SMUXC_PUART2_CTS)

Address: IOMUXC_SMUXC_PUART2_CTS is 53FA_8000h base + 9Ch offset = 53FA_809Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

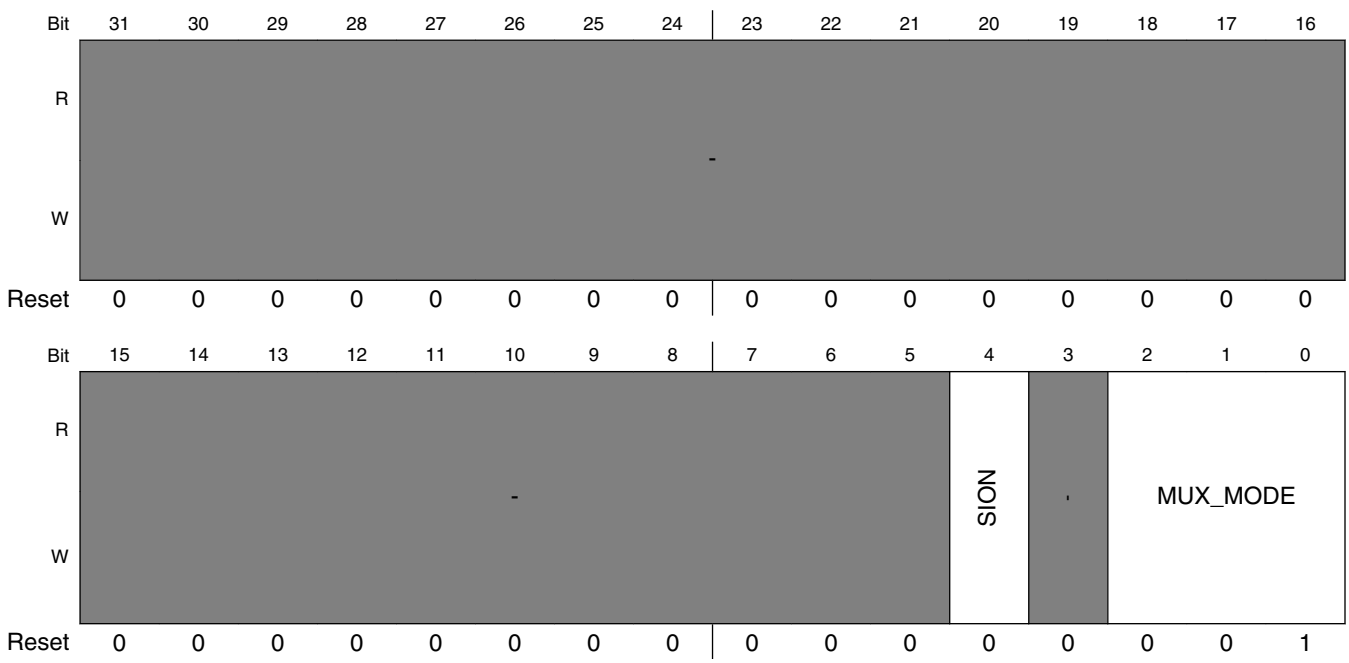
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PUART2_CTS field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART2_CTS. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: UART2_CTS. **NOTE:** Pad UART2_CTS is involved in Daisy Chain. • Config Register IOMUXC_ESDHC4_IPP_CMD_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_ESDHC4_IPP_DAT6_IN_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: CTS of instance: uart2. 001 Select mux mode: ALT1 mux port: GPIO[12] of instance: gpio6. 100 Select mux mode: ALT4 mux port: CMD of instance: esdhc4. 101 Select mux mode: ALT5 mux port: DAT6 of instance: esdhc4. 111 Select mux mode: ALT7 mux port: DATAOUT[12] of instance: usbphy2.

35.4.41 SW_MUX_CTL_PAD_UART2_RTS (IOMUXC_SMUXC_PUART2_RTS)

Address: IOMUXC_SMUXC_PUART2_RTS is 53FA_8000h base + A0h offset = 53FA_80A0h

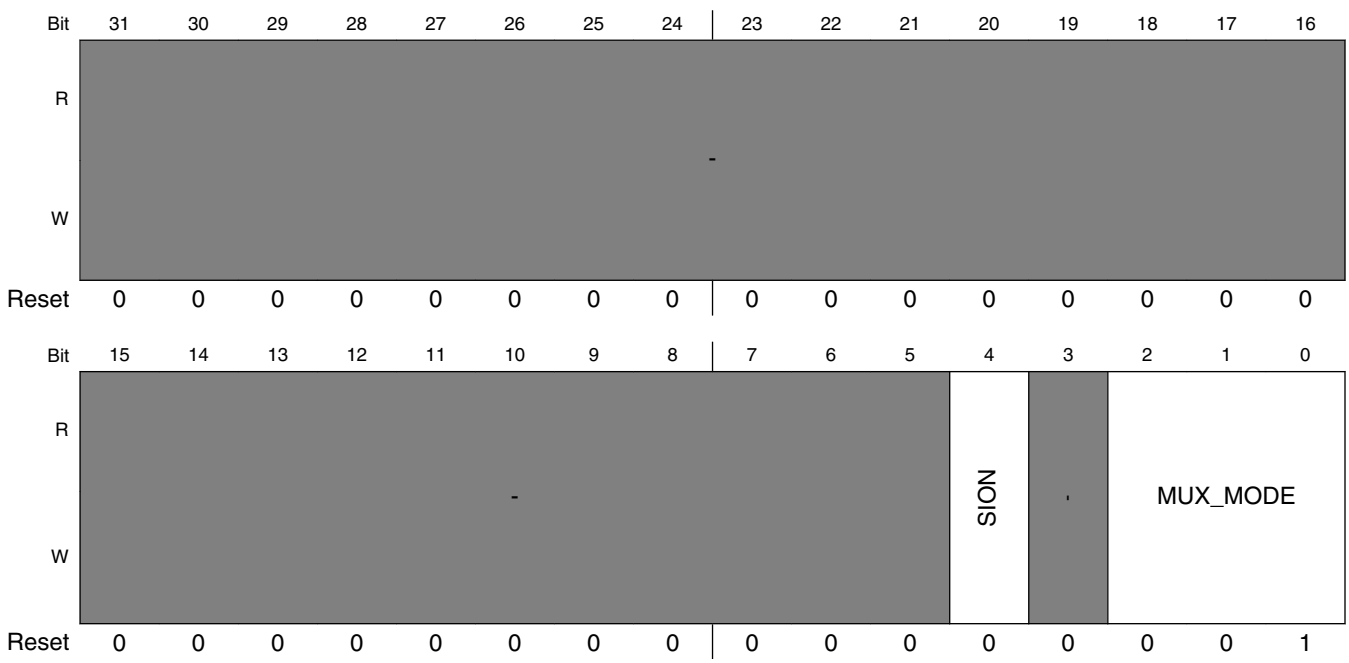


IOMUXC_SMUXC_PUART2_RTS field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART2_RTS. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: UART2_RTS. **NOTE:** Pad UART2_RTS is involved in Daisy Chain. • Config Register IOMUXC_ESDHC4_IPP_CARD_CLK_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_ESDHC4_IPP_DAT7_IN_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: RTS of instance: uart2. 001 Select mux mode: ALT1 mux port: GPIO[13] of instance: gpio6. 100 Select mux mode: ALT4 mux port: CLK of instance: esdhc4. 101 Select mux mode: ALT5 mux port: DAT7 of instance: esdhc4. 111 Select mux mode: ALT7 mux port: DATAOUT[13] of instance: usbphy2.

35.4.42 SW_MUX_CTL_PAD_UART3_TXD (IOMUXC_SMUXC_PUART3_TXD)

Address: IOMUXC_SMUXC_PUART3_TXD is 53FA_8000h base + A4h offset = 53FA_80A4h

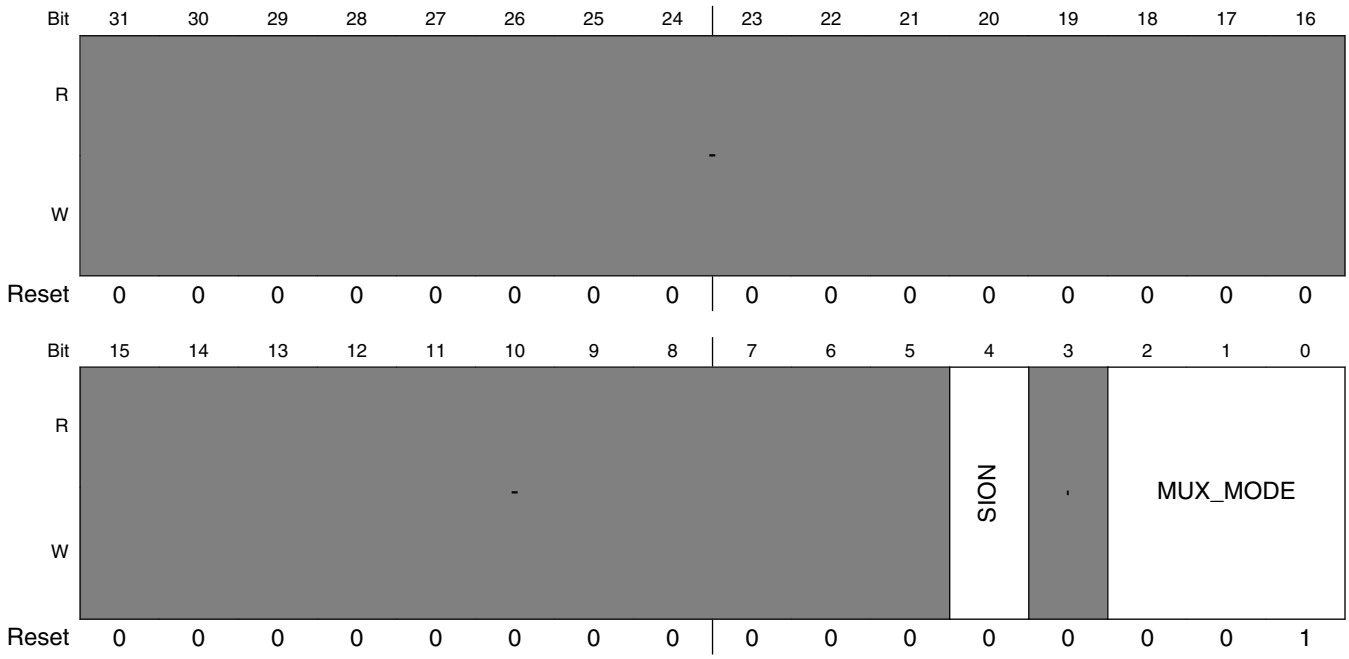


IOMUXC_SMUXC_PUART3_TXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART3_TXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: UART3_TXD. **NOTE:** Pad UART3_TXD is involved in Daisy Chain. • Config Register IOMUXC_ESDHC2_IPP_WP_ON_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_ESDHC4_IPP_DAT0_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_12_SELECT_INPUT for mode ALT6. 000 Select mux mode: ALT0 mux port: TXD_MUX of instance: uart3. 001 Select mux mode: ALT1 mux port: GPIO[14] of instance: gpio6. 011 Select mux mode: ALT3 mux port: DAT4 of instance: esdhc1. 100 Select mux mode: ALT4 mux port: DAT0 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: WP of instance: esdhc2. 110 Select mux mode: ALT6 mux port: D[12] of instance: weimv2. 111 Select mux mode: ALT7 mux port: DATAOUT[14] of instance: usbphy2.

35.4.43 SW_MUX_CTL_PAD_UART3_RXD (IOMUXC_SMUXC_PUART3_RXD)

Address: IOMUXC_SMUXC_PUART3_RXD is 53FA_8000h base + A8h offset = 53FA_80A8h



IOMUXC_SMUXC_PUART3_RXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART3_RXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: UART3_RXD. **NOTE:** Pad UART3_RXD is involved in Daisy Chain. • Config Register IOMUXC_ESDHC2_IPP_CARD_DET_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_ESDHC4_IPP_DAT1_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_13_SELECT_INPUT for mode ALT6. 000 Select mux mode: ALT0 mux port: RXD_MUX of instance: uart3. 001 Select mux mode: ALT1 mux port: GPIO[15] of instance: gpio6. 011 Select mux mode: ALT3 mux port: DAT5 of instance: esdhc1. 100 Select mux mode: ALT4 mux port: DAT1 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: CD of instance: esdhc2.

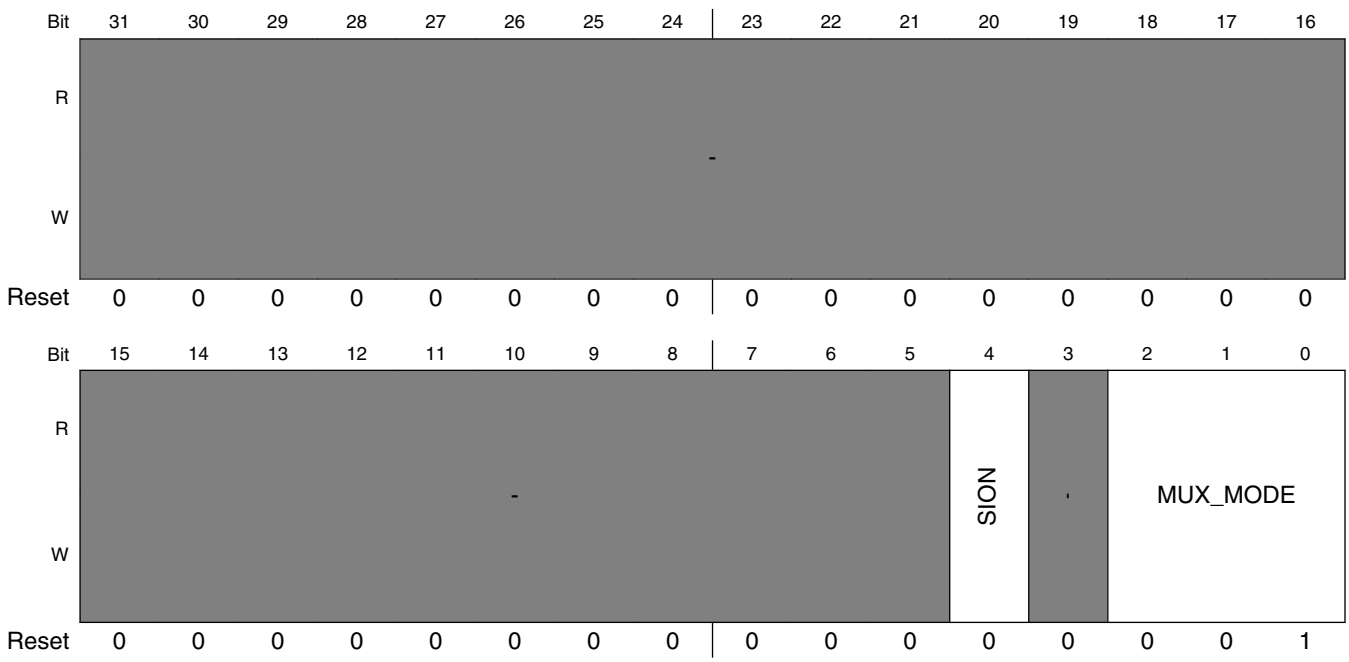
Table continues on the next page...

IOMUXC_SMUXC_PUART3_RXD field descriptions (continued)

Field	Description
	110 Select mux mode: ALT6 mux port: D[13] of instance: weimv2. 111 Select mux mode: ALT7 mux port: DATAOUT[15] of instance: usbphy2.

35.4.44 SW_MUX_CTL_PAD_UART4_TXD (IOMUXC_SMUXC_PUART4_TXD)

Address: IOMUXC_SMUXC_PUART4_TXD is 53FA_8000h base + ACh offset = 53FA_80ACh



IOMUXC_SMUXC_PUART4_TXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART4_TXD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: UART4_TXD. **NOTE:** Pad UART4_TXD is involved in Daisy Chain. • Config Register IOMUXC_ESDHC4_IPP_DAT2_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT for mode ALT2.

Table continues on the next page...

IOMUXC_SMUXC_PUART4_TXD field descriptions (continued)

Field	Description
	• Config Register IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_14_SELECT_INPUT for mode ALT6. 000 Select mux mode: ALT0 mux port: TXD_MUX of instance: uart4. 001 Select mux mode: ALT1 mux port: GPIO[16] of instance: gpio6. 010 Select mux mode: ALT2 mux port: CTS of instance: uart3. 011 Select mux mode: ALT3 mux port: DAT6 of instance: esdhc1. 100 Select mux mode: ALT4 mux port: DAT2 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: LCTL of instance: esdhc2. 110 Select mux mode: ALT6 mux port: D[14] of instance: weimv2.

35.4.45 SW_MUX_CTL_PAD_UART4_RXD (IOMUXC_SMUXC_PUART4_RXD)

Address: IOMUXC_SMUXC_PUART4_RXD is 53FA_8000h base + B0h offset = 53FA_80B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PUART4_RXD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad UART4_RXD. 0 Input Path is determined by functionality of the selected mux mode (regular).

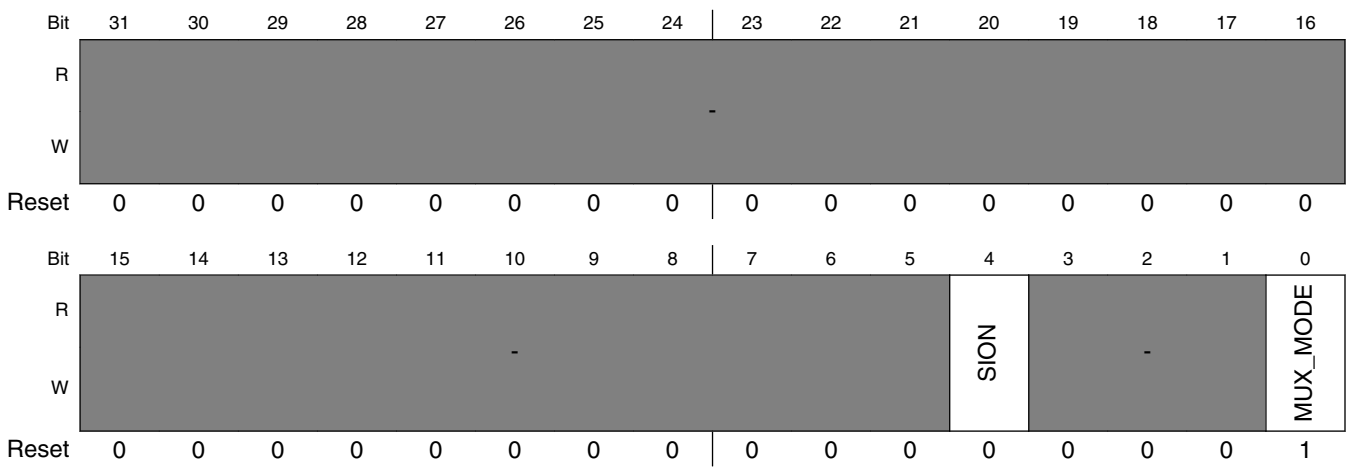
Table continues on the next page...

IOMUXC_SMUXC_PUART4_RXD field descriptions (continued)

Field	Description
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: UART4_RXD. **NOTE:** Pad UART4_RXD is involved in Daisy Chain. • Config Register IOMUXC_ESDHC4_IPP_DAT3_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT0. v IOMUXC_WEIMV2_IPP_IND_READ_DATA_15_SELECT_INPUT for mode ALT6. 000 Select mux mode: ALT0 mux port: RXD_MUX of instance: uart4. 001 Select mux mode: ALT1 mux port: GPIO[17] of instance: gpio6. 010 Select mux mode: ALT2 mux port: RTS of instance: uart3. 011 Select mux mode: ALT3 mux port: DAT7 of instance: esdhc1. 100 Select mux mode: ALT4 mux port: DAT3 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: LCTL of instance: esdhc1. 110 Select mux mode: ALT6 mux port: D[15] of instance: weimv2.

35.4.46 SW_MUX_CTL_PAD_CSPI_SCLK (IOMUXC_SMUXC_PCSPI_SCLK)

Address: IOMUXC_SMUXC_PCSPI_SCLK is 53FA_8000h base + B4h offset = 53FA_80B4h



IOMUXC_SMUXC_PCSPI_SCLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SMUXC_PCSPI_SCLK field descriptions (continued)

Field	Description
	1 Force input path of pad CSPI_SCLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–1 -	Reserved
0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: CSPI_SCLK. 00 Select mux mode: ALT0 mux port: SCLK of instance: cspi. 01 Select mux mode: ALT1 mux port: GPIO[8] of instance: gpio4.

35.4.47 SW_MUX_CTL_PAD_CSPI_MOSI (IOMUXC_SMUXC_PCSPI_MOSI)

Address: IOMUXC_SMUXC_PCSPI_MOSI is 53FA_8000h base + B8h offset = 53FA_80B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-											SION	-			MUX_MODE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PCSPI_MOSI field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad CSPI_MOSI. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–1 -	Reserved
0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: CSPI_MOSI. 00 Select mux mode: ALT0 mux port: MOSI of instance: cspi. 01 Select mux mode: ALT1 mux port: GPIO[9] of instance: gpio4.

35.4.48 SW_MUX_CTL_PAD_CSPI_MISO (IOMUXC_SMUXC_PCSPI_MISO)

Address: IOMUXC_SMUXC_PCSPI_MISO is 53FA_8000h base + BCh offset = 53FA_80BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

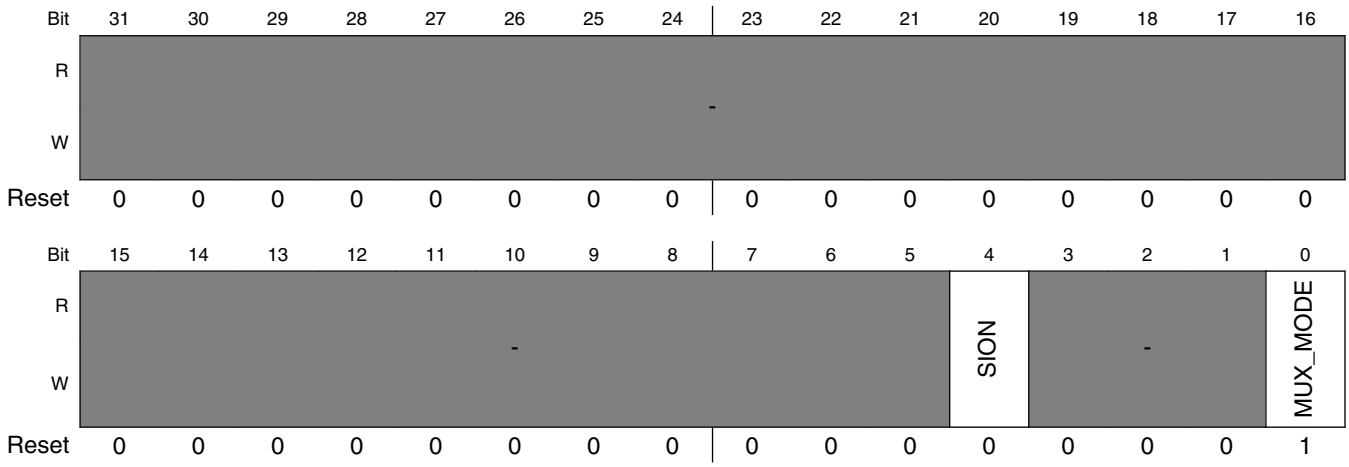
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-			MUX_MODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PCSPI_MISO field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad CSPI_MISO. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–1 -	Reserved
0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: CSPI_MISO. 00 Select mux mode: ALT0 mux port: MISO of instance: cspi. 01 Select mux mode: ALT1 mux port: GPIO[10] of instance: gpio4.

35.4.49 SW_MUX_CTL_PAD_CSPI_SS0 (IOMUXC_SMUXC_PCSPI_SS0)

Address: IOMUXC_SMUXC_PCSPI_SS0 is 53FA_8000h base + C0h offset = 53FA_80C0h

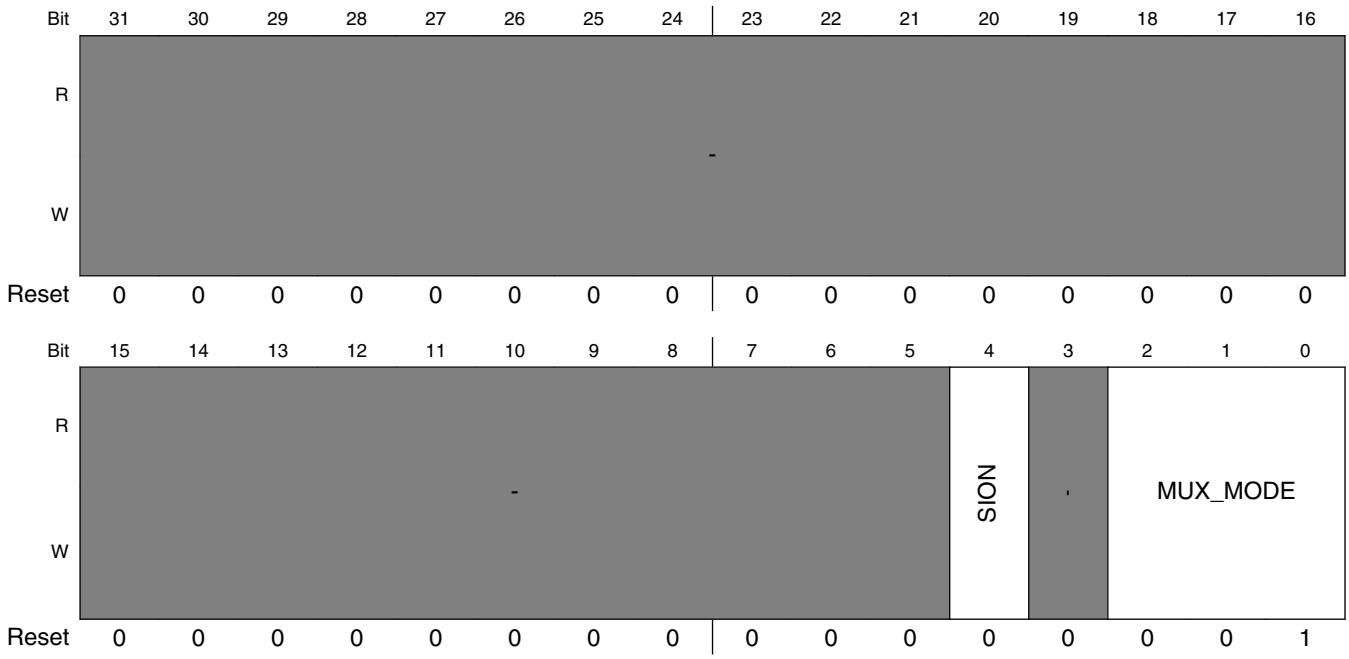


IOMUXC_SMUXC_PCSPI_SS0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad CSPI_SS0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–1 -	Reserved
0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: CSPI_SS0. 00 Select mux mode: ALT0 mux port: SS0 of instance: cspi. 01 Select mux mode: ALT1 mux port: GPIO[11] of instance: gpio4.

35.4.50 SW_MUX_CTL_PAD_ECSPI1_SCLK (IOMUXC_SMUXC_PECSPI1_SCLK)

Address: IOMUXC_SMUXC_PECSPI1_SCLK is 53FA_8000h base + C4h offset = 53FA_80C4h



IOMUXC_SMUXC_PECSPI1_SCLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad ECSPI1_SCLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ECSPI1_SCLK. **NOTE:** Pad ECSPI1_SCLK is involved in Daisy Chain. • Config Register IOMUXC_CSPI_IPP_IND_DATAREADY_B_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4. • Config RegisterConfig Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_8_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: SCLK of instance: ecspi1. 001 Select mux mode: ALT1 mux port: GPIO[12] of instance: gpio4. 010 Select mux mode: ALT2 mux port: RDY of instance: cspi. 011 Select mux mode: ALT3 mux port: RDY of instance: ecspi2. 100 Select mux mode: ALT4 mux port: RTS of instance: uart3.

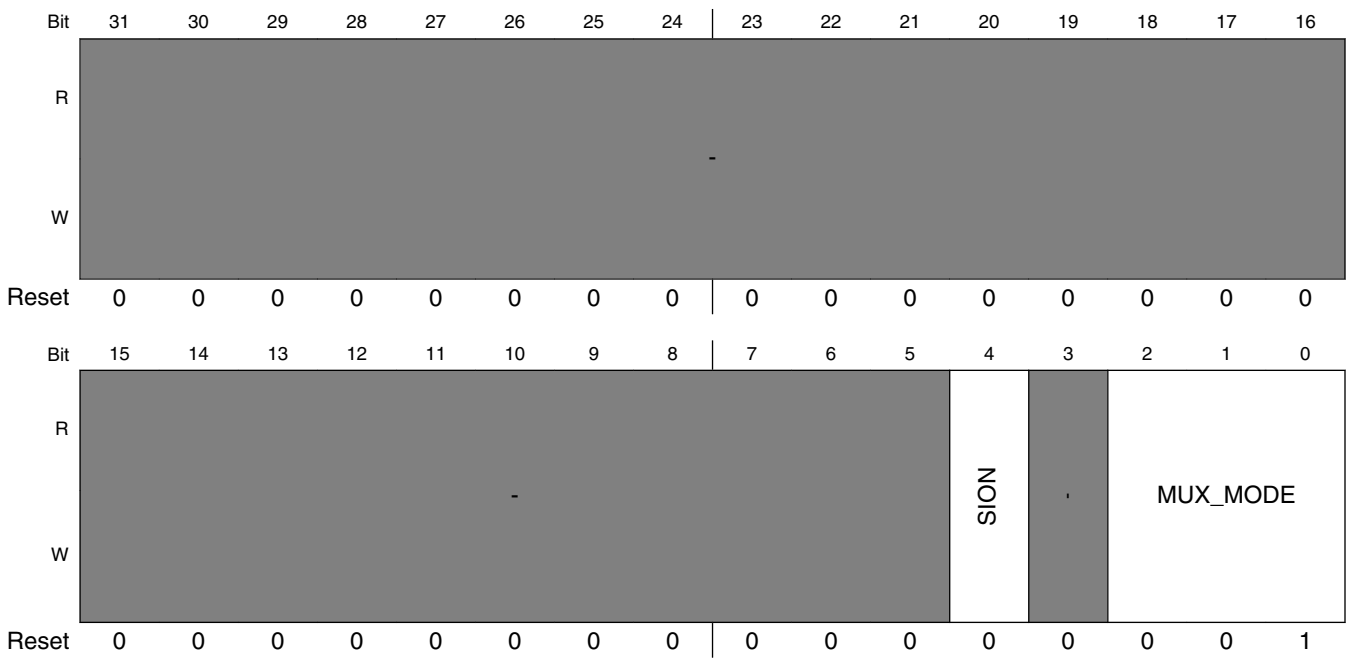
Table continues on the next page...

IOMUXC_SMUXC_PECSPI1_SCLK field descriptions (continued)

Field	Description
	101 Select mux mode: ALT5 mux port: SDCE[6] of instance: epdc. 111 Select mux mode: ALT7 mux port: D[8] of instance: weimv2.

35.4.51 SW_MUX_CTL_PAD_ECSPI1_MOSI (IOMUXC_SMUXC_PECSPI1_MOSI)

Address: IOMUXC_SMUXC_PECSPI1_MOSI is 53FA_8000h base + C8h offset = 53FA_80C8h



IOMUXC_SMUXC_PECSPI1_MOSI field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad ECSPI1_MOSI. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ECSPI1_MOSI. **NOTE:** Pad ECSPI1_MOSI is involved in Daisy Chain. • Config Register IOMUXC_CSPI_IPP_IND_SS1_B_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_9_SELECT_INPUT for mode ALT7.

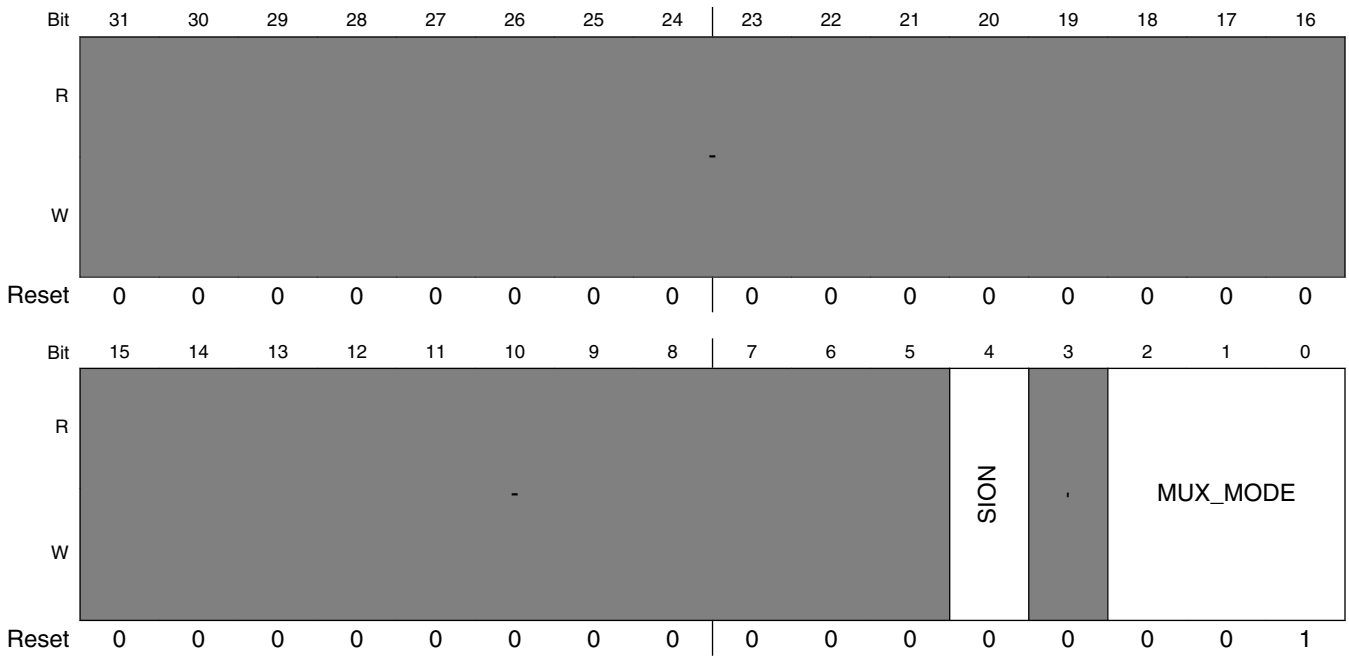
Table continues on the next page...

IOMUXC_SMUXC_PECSPI1_MOSI field descriptions (continued)

Field	Description
	000 Select mux mode: ALT0 mux port: MOSI of instance: ecspi1. 001 Select mux mode: ALT1 mux port: GPIO[13] of instance: gpio4. 010 Select mux mode: ALT2 mux port: SS1 of instance: cspi. 011 Select mux mode: ALT3 mux port: SS1 of instance: ecspi2. 100 Select mux mode: ALT4 mux port: CTS of instance: uart3. 101 Select mux mode: ALT5 mux port: SDCE[7] of instance: epdc. 111 Select mux mode: ALT7 mux port: D[9] of instance: weimv2.

35.4.52 SW_MUX_CTL_PAD_ECSPI1_MISO (IOMUXC_SMUXC_PECSPI1_MISO)

Address: IOMUXC_SMUXC_PECSPI1_MISO is 53FA_8000h base + CCh offset = 53FA_80CCh



IOMUXC_SMUXC_PECSPI1_MISO field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad ECSPI1_MISO. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved

Table continues on the next page...

IOMUXC_SMUXC_PECSPI1_MISO field descriptions (continued)

Field	Description
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ECSPI1_MISO. **NOTE:** Pad ECSPI1_MISO is involved in Daisy Chain. • Config Register IOMUXC_CSPI_IPP_IND_SS2_B_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_10_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: MISO of instance: ecspi1. 001 Select mux mode: ALT1 mux port: GPIO[14] of instance: gpio4. 010 Select mux mode: ALT2 mux port: SS2 of instance: cspi. 011 Select mux mode: ALT3 mux port: SS2 of instance: ecspi2. 100 Select mux mode: ALT4 mux port: RTS of instance: uart4. 101 Select mux mode: ALT5 mux port: SDCE[8] of instance: epdc. 111 Select mux mode: ALT7 mux port: D[10] of instance: weimv2.

35.4.53 SW_MUX_CTL_PAD_ECSPI1_SS0 (IOMUXC_SMUXC_PECSPI1_SS0)

Address: IOMUXC_SMUXC_PECSPI1_SS0 is 53FA_8000h base + D0h offset = 53FA_80D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PECSPI1_SS0 field descriptions

Field	Description
31–5 -	Reserved

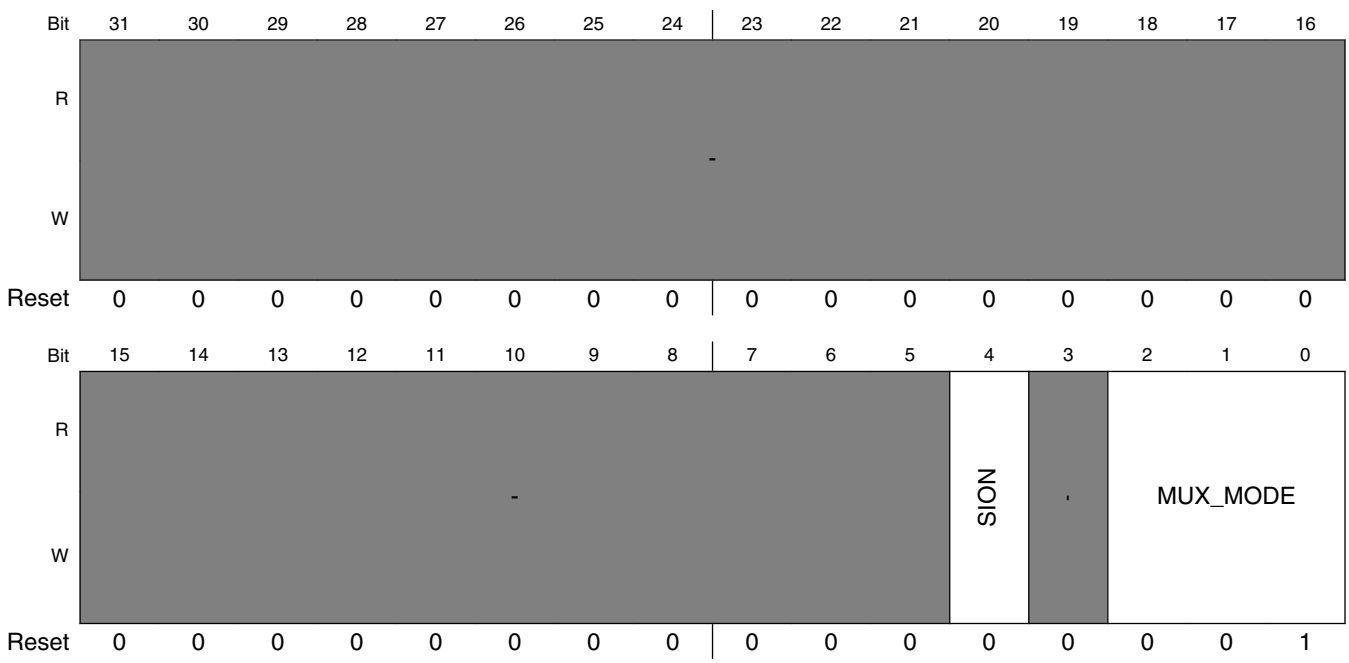
Table continues on the next page...

IOMUXC_SMUXC_PECSPI1_SS0 field descriptions (continued)

Field	Description
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad ECSPI1_SS0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: ECSPI1_SS0. **NOTE:** Pad ECSPI1_SS0 is involved in Daisy Chain. • Config Register IOMUXC_CSPI_IPP_IND_SS3_B_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_11_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: SS0 of instance: ecspi1. 001 Select mux mode: ALT1 mux port: GPIO[15] of instance: gpio4. 010 Select mux mode: ALT2 mux port: SS3 of instance: cspi. 011 Select mux mode: ALT3 mux port: SS3 of instance: ecspi2. 100 Select mux mode: ALT4 mux port: CTS of instance: uart4. 101 Select mux mode: ALT5 mux port: SDCE[9] of instance: epdc. 111 Select mux mode: ALT7 mux port: D[11] of instance: weimv2.

35.4.54 SW_MUX_CTL_PAD_ECSPI2_SCLK (IOMUXC_SMUXC_PECSPI2_SCLK)

Address: IOMUXC_SMUXC_PECSPI2_SCLK is 53FA_8000h base + D4h offset = 53FA_80D4h

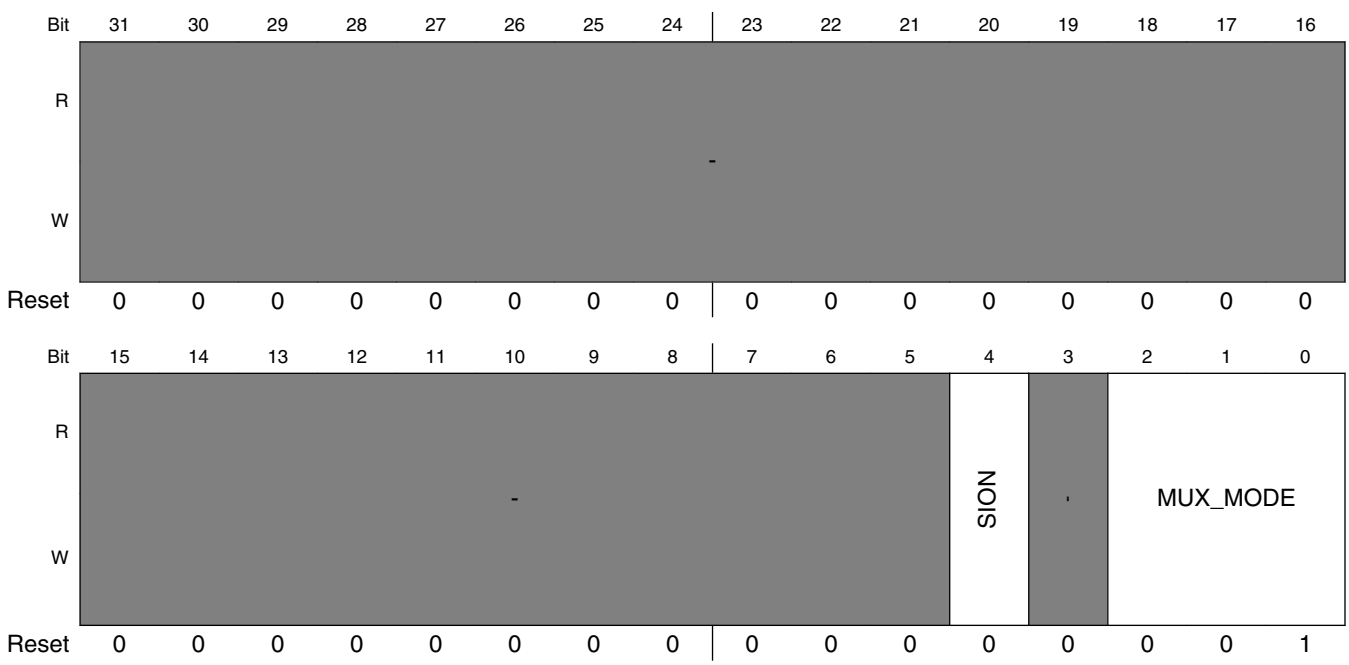


IOMUXC_SMUXC_PECSPI2_SCLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad ECSPI2_SCLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI2_SCLK. **NOTE:** Pad ECSPI2_SCLK is involved in Daisy Chain. • Config Register IOMUXC_UART5_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_8_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: SCLK of instance: ecspi2. 001 Select mux mode: ALT1 mux port: GPIO[16] of instance: gpio4. 010 Select mux mode: ALT2 mux port: WR_RWN of instance: elcdif. 011 Select mux mode: ALT3 mux port: RDY of instance: ecspi1. 100 Select mux mode: ALT4 mux port: RTS of instance: uart5. 101 Select mux mode: ALT5 mux port: DOTCLK of instance: elcdif. 110 Select mux mode: ALT6 mux port: CEN[4] of instance: rawnand. 111 Select mux mode: ALT7 mux port: D[8] of instance: weimv2.

35.4.55 SW_MUX_CTL_PAD_ECSPI2_MOSI (IOMUXC_SMUXC_PECSPI2_MOSI)

Address: IOMUXC_SMUXC_PECSPI2_MOSI is 53FA_8000h base + D8h offset = 53FA_80D8h

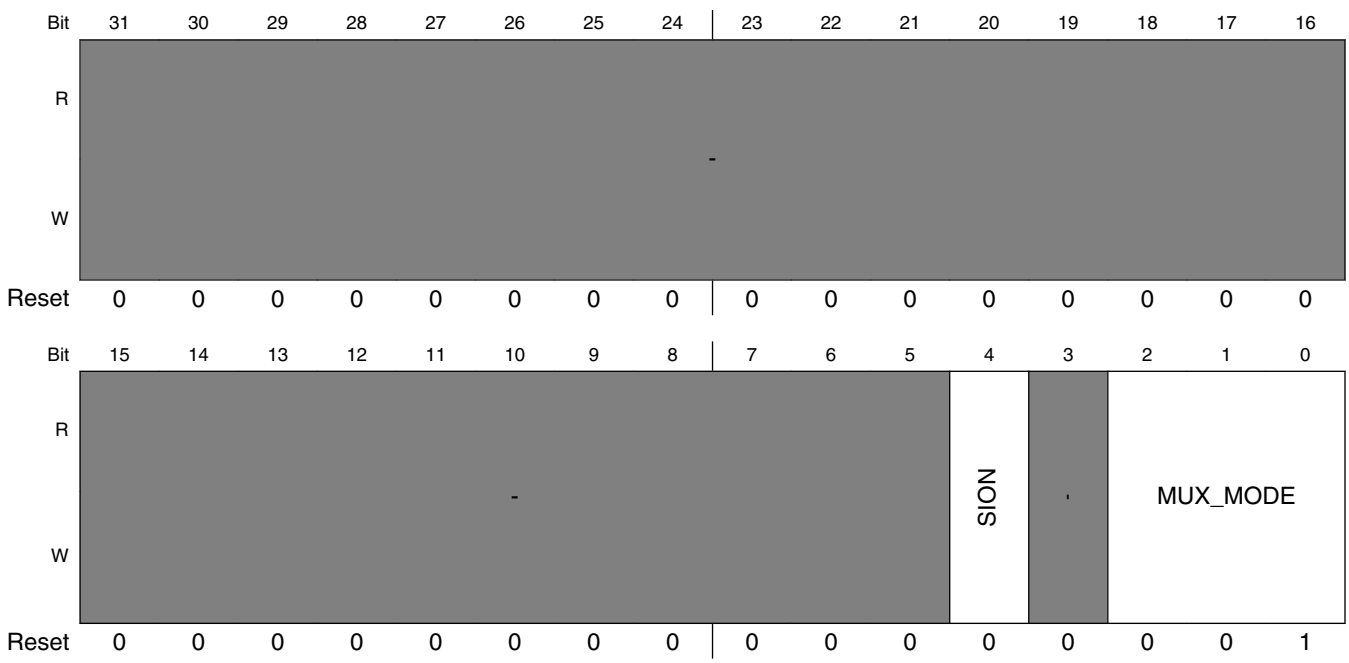


IOMUXC_SMUXC_PECSPI2_MOSI field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad ECSPI2_MOSI. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI2_MOSI. **NOTE:** Pad ECSPI2_MOSI is involved in Daisy Chain. • Config Register IOMUXC_UART5_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_9_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: MOSI of instance: ecspi2. 001 Select mux mode: ALT1 mux port: GPIO[17] of instance: gpio4. 010 Select mux mode: ALT2 mux port: RD_E of instance: elcdif. 011 Select mux mode: ALT3 mux port: SS1 of instance: ecspi1. 100 Select mux mode: ALT4 mux port: CTS of instance: uart5. 101 Select mux mode: ALT5 mux port: ENABLE of instance: elcdif. 110 Select mux mode: ALT6 mux port: CEN[5] of instance: rawnand. 111 Select mux mode: ALT7 mux port: D[9] of instance: weimv2.

35.4.56 SW_MUX_CTL_PAD_ECSPI2_MISO (IOMUXC_SMUXC_PECSPI2_MISO)

Address: IOMUXC_SMUXC_PECSPI2_MISO is 53FA_8000h base + DCh offset = 53FA_80DCh

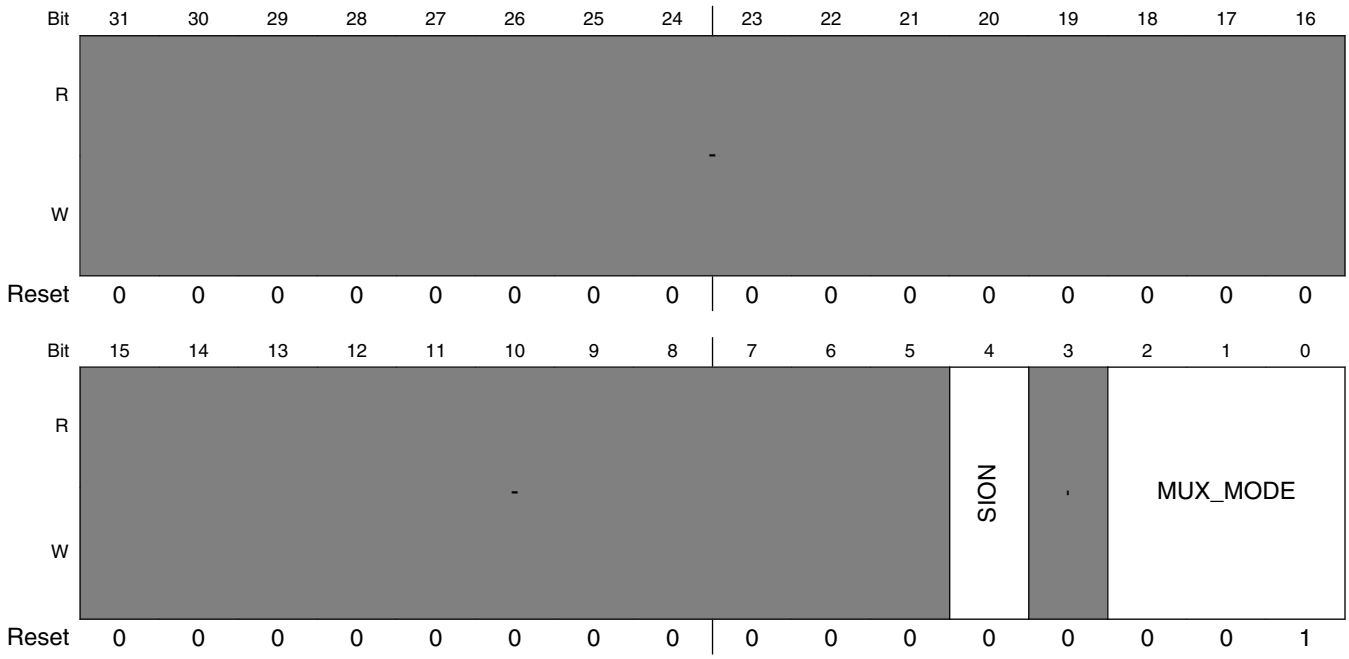


IOMUXC_SMUXC_PECSPI2_MISO field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad ECSPI2_MISO. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI2_MISO. **NOTE:** Pad ECSPI2_MISO is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_VSYNC_I_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_10_SELECT_INPUT for mode ALT7. 101 Select mux mode: ALT5 mux port: VSYNC of instance: elcdif. 110 Select mux mode: ALT6 mux port: CEN[6] of instance: rawnand. 111 Select mux mode: ALT7 mux port: D[10] of instance: weimv2. 000 Select mux mode: ALT0 mux port: MISO of instance: ecspi2. 001 Select mux mode: ALT1 mux port: GPIO[18] of instance: gpio4. 010 Select mux mode: ALT2 mux port: RS of instance: elcdif. 011 Select mux mode: ALT3 mux port: SS2 of instance: ecspi1. 100 Select mux mode: ALT4 mux port: TXD_MUX of instance: uart5.

35.4.57 SW_MUX_CTL_PAD_ECSPI2_SS0 (IOMUXC_SMUXC_PECSPI2_SS0)

Address: IOMUXC_SMUXC_PECSPI2_SS0 is 53FA_8000h base + E0h offset = 53FA_80E0h



IOMUXC_SMUXC_PECSPI2_SS0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad ECSPI2_SS0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: ECSPI2_SS0. **NOTE:** Pad ECSPI2_SS0 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_BUSY_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_11_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: SS0 of instance: ecspi2. 001 Select mux mode: ALT1 mux port: GPIO[19] of instance: gpio4. 010 Select mux mode: ALT2 mux port: CS of instance: elcdif. 011 Select mux mode: ALT3 mux port: SS3 of instance: ecspi1. 100 Select mux mode: ALT4 mux port: RXD_MUX of instance: uart5. 101 Select mux mode: ALT5 mux port: HSYNC of instance: elcdif.

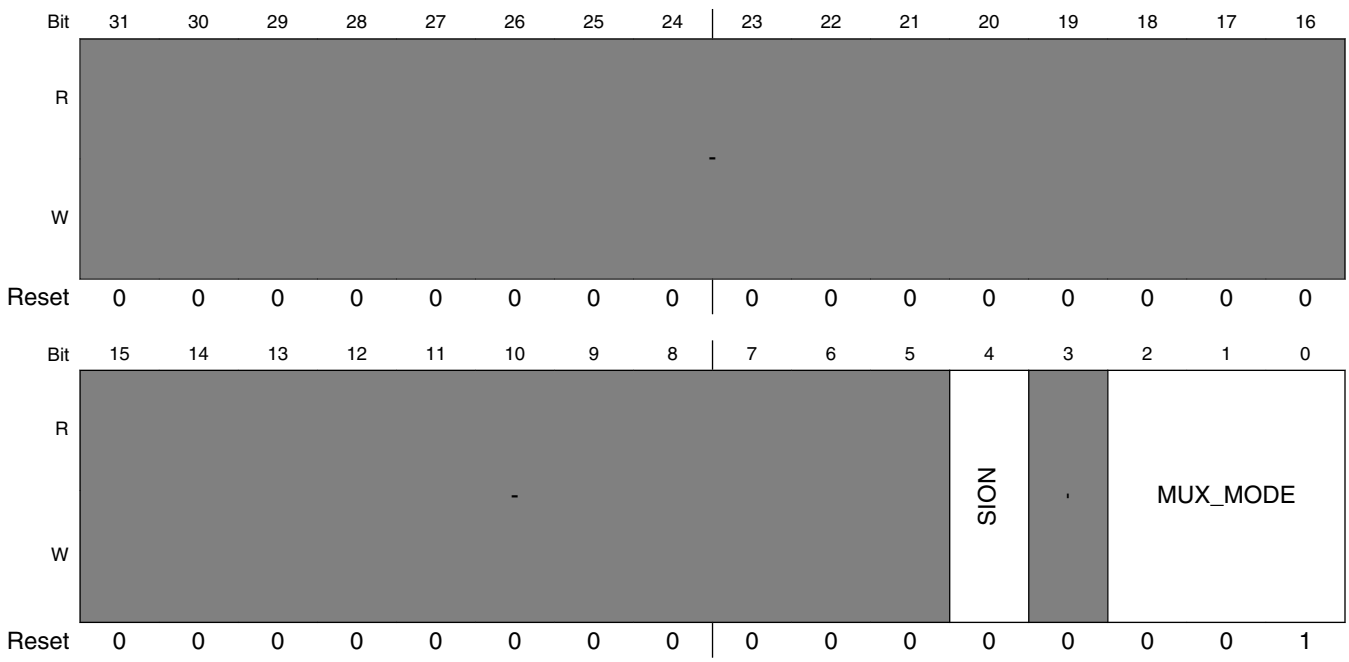
Table continues on the next page...

IOMUXC_SMUXC_PECSPI2_SS0 field descriptions (continued)

Field	Description
	110 Select mux mode: ALT6 mux port: CEN[7] of instance: rawnand. 111 Select mux mode: ALT7 mux port: D[11] of instance: weimv2.

35.4.58 SW_MUX_CTL_PAD_SD1_CLK (IOMUXC_SMUXC_PSD1_CLK)

Address: IOMUXC_SMUXC_PSD1_CLK is 53FA_8000h base + E4h offset = 53FA_80E4h



IOMUXC_SMUXC_PSD1_CLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD1_CLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: SD1_CLK. 000 Select mux mode: ALT0 mux port: CLK of instance: esdhc1. 001 Select mux mode: ALT1 mux port: GPIO[0] of instance: gpio5. 111 Select mux mode: ALT7 mux port: CLKO of instance: ccm.

35.4.59 SW_MUX_CTL_PAD_SD1_CMD (IOMUXC_SMUXC_PSD1_CMD)

Address: IOMUXC_SMUXC_PSD1_CMD is 53FA_8000h base + E8h offset = 53FA_80E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

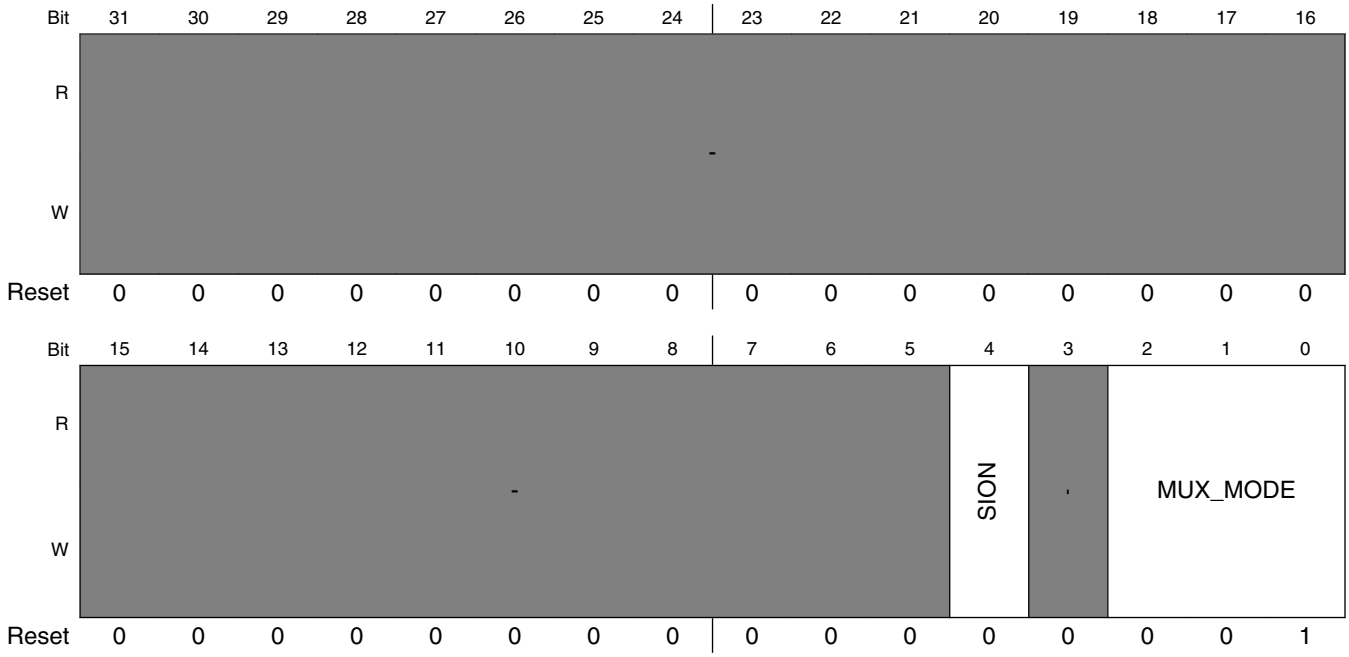
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD1_CMD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD1_CMD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: SD1_CMD. 000 Select mux mode: ALT0 mux port: CMD of instance: esdhc1. 001 Select mux mode: ALT1 mux port: GPIO[1] of instance: gpio5. 111 Select mux mode: ALT7 mux port: CLKO2 of instance: ccm.

35.4.60 SW_MUX_CTL_PAD_SD1_D0 (IOMUXC_SMUXC_PSD1_D0)

Address: IOMUXC_SMUXC_PSD1_D0 is 53FA_8000h base + ECh offset = 53FA_80ECh

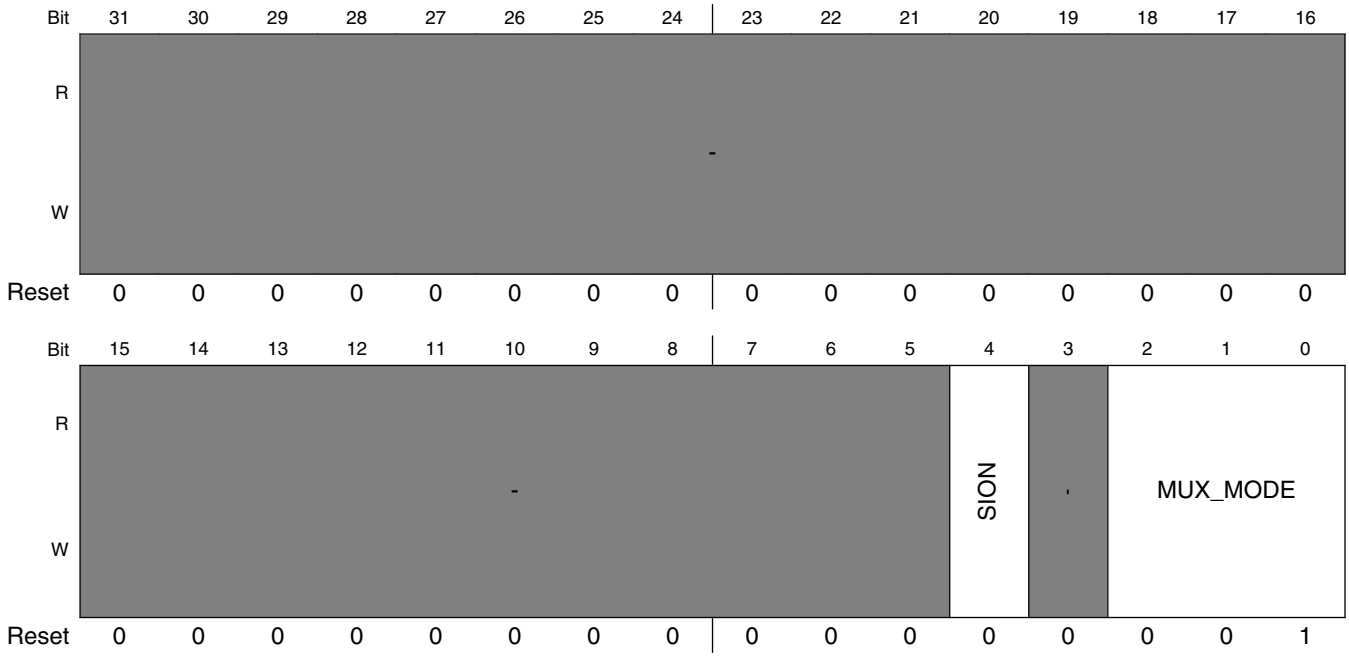


IOMUXC_SMUXC_PSD1_D0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD1_D0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: SD1_D0. **NOTE:** Pad SD1_D0 is involved in Daisy Chain. • Config Register IOMUXC_CCM_PLL1_BYPASS_CLK_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: DAT0 of instance: esdhc1. 001 Select mux mode: ALT1 mux port: GPIO[2] of instance: gpio5. 111 Select mux mode: ALT7 mux port: PLL1_BYP of instance: ccm.

35.4.61 SW_MUX_CTL_PAD_SD1_D1 (IOMUXC_SMUXC_PSD1_D1)

Address: IOMUXC_SMUXC_PSD1_D1 is 53FA_8000h base + F0h offset = 53FA_80F0h

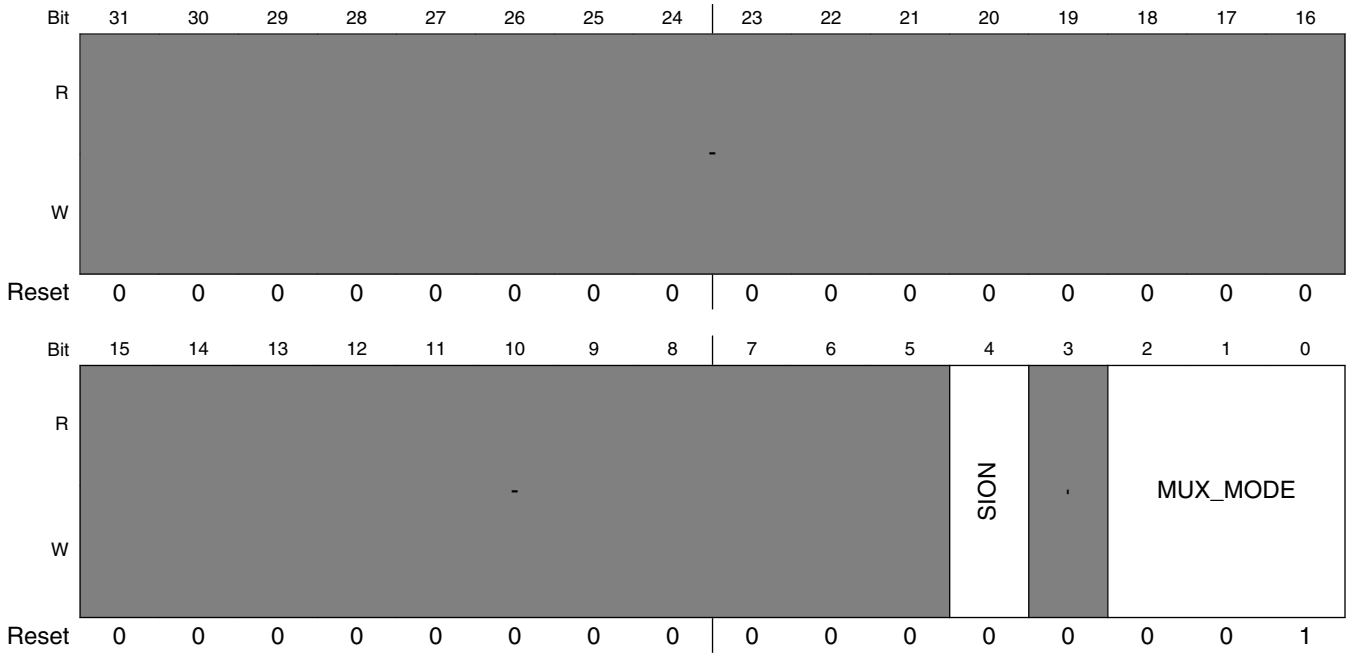


IOMUXC_SMUXC_PSD1_D1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD1_D1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: SD1_D1. **NOTE:** Pad SD1_D1 is involved in Daisy Chain. • Config Register IOMUXC_CCM_PLL2_BYPASS_CLK_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: DAT1 of instance: esdhc1. 001 Select mux mode: ALT1 mux port: GPIO[3] of instance: gpio5. 111 Select mux mode: ALT7 mux port: PLL2_BYP of instance: ccm.

35.4.62 SW_MUX_CTL_PAD_SD1_D2 (IOMUXC_SMUXC_PSD1_D2)

Address: IOMUXC_SMUXC_PSD1_D2 is 53FA_8000h base + F4h offset = 53FA_80F4h



IOMUXC_SMUXC_PSD1_D2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD1_D2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: SD1_D2. **NOTE:** Pad SD1_D2 is involved in Daisy Chain. • Config Register IOMUXC_CCM_PLL3_BYPASS_CLK_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: DAT2 of instance: esdhc1. 001 Select mux mode: ALT1 mux port: GPIO[4] of instance: gpio5. 111 Select mux mode: ALT7 mux port: PLL3_BYP of instance: ccm.

35.4.63 SW_MUX_CTL_PAD_SD1_D3 (IOMUXC_SMUXC_PSD1_D3)

Address: IOMUXC_SMUXC_PSD1_D3 is 53FA_8000h base + F8h offset = 53FA_80F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

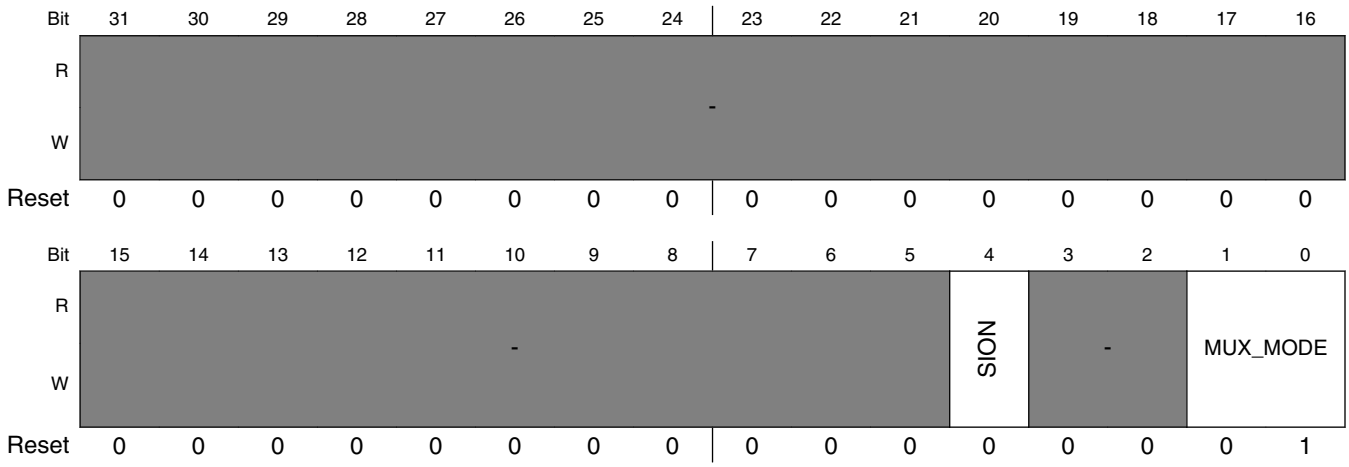
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-			MUX_MODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD1_D3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD1_D3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–1 -	Reserved
0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: SD1_D3. 00 Select mux mode: ALT0 mux port: DAT3 of instance: esdhc1. 01 Select mux mode: ALT1 mux port: GPIO[5] of instance: gpio5.

35.4.64 SW_MUX_CTL_PAD_SD2_CLK (IOMUXC_SMUXC_PSD2_CLK)

Address: IOMUXC_SMUXC_PSD2_CLK is 53FA_8000h base + FCh offset = 53FA_80FCh



IOMUXC_SMUXC_PSD2_CLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_CLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: SD2_CLK. 00 Select mux mode: ALT0 mux port: CLK of instance: esdhc2. 01 Select mux mode: ALT1 mux port: GPIO[6] of instance: gpio5. 10 Select mux mode: ALT2 mux port: SCLK of instance: mshc.

35.4.65 SW_MUX_CTL_PAD_SD2_CMD (IOMUXC_SMUXC_PSD2_CMD)

Address: IOMUXC_SMUXC_PSD2_CMD is 53FA_8000h base + 100h offset = 53FA_8100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

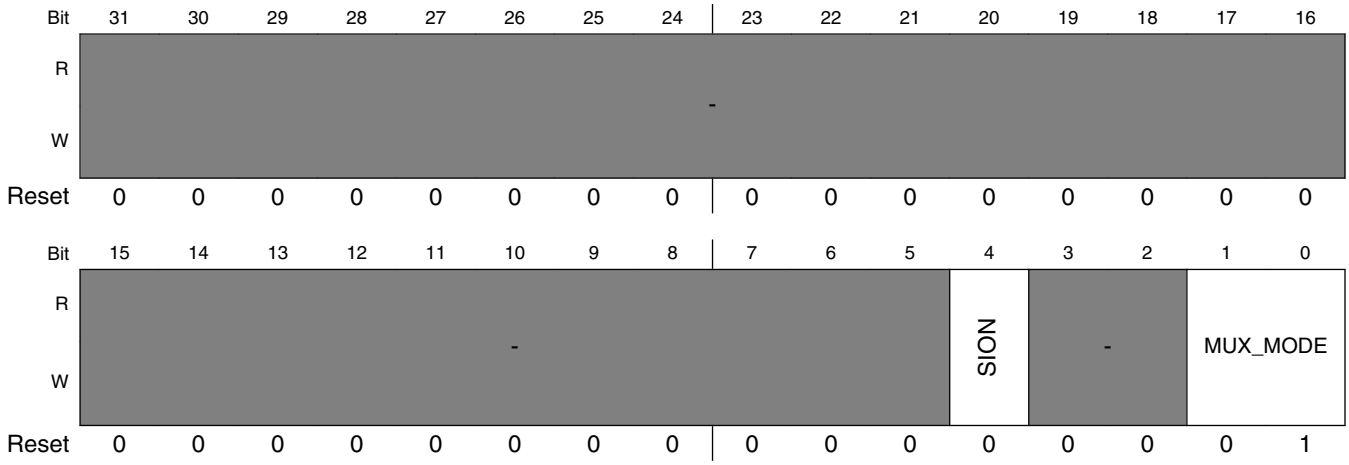
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD2_CMD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_CMD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: SD2_CMD. 00 Select mux mode: ALT0 mux port: CMD of instance: esdhc2. 01 Select mux mode: ALT1 mux port: GPIO[7] of instance: gpio5. 10 Select mux mode: ALT2 mux port: BS of instance: mshc.

35.4.66 SW_MUX_CTL_PAD_SD2_D0 (IOMUXC_SMUXC_PSD2_D0)

Address: IOMUXC_SMUXC_PSD2_D0 is 53FA_8000h base + 104h offset = 53FA_8104h



IOMUXC_SMUXC_PSD2_D0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_D0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD2_D0. **NOTE:** Pad SD2_D0 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: DAT0 of instance: esdhc2. 01 Select mux mode: ALT1 mux port: GPIO[8] of instance: gpio5. 10 Select mux mode: ALT2 mux port: DATA[0] of instance: mshc. 11 Select mux mode: ALT3 mux port: COL[4] of instance: kpp.

35.4.67 SW_MUX_CTL_PAD_SD2_D1 (IOMUXC_SMUXC_PSD2_D1)

Address: IOMUXC_SMUXC_PSD2_D1 is 53FA_8000h base + 108h offset = 53FA_8108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

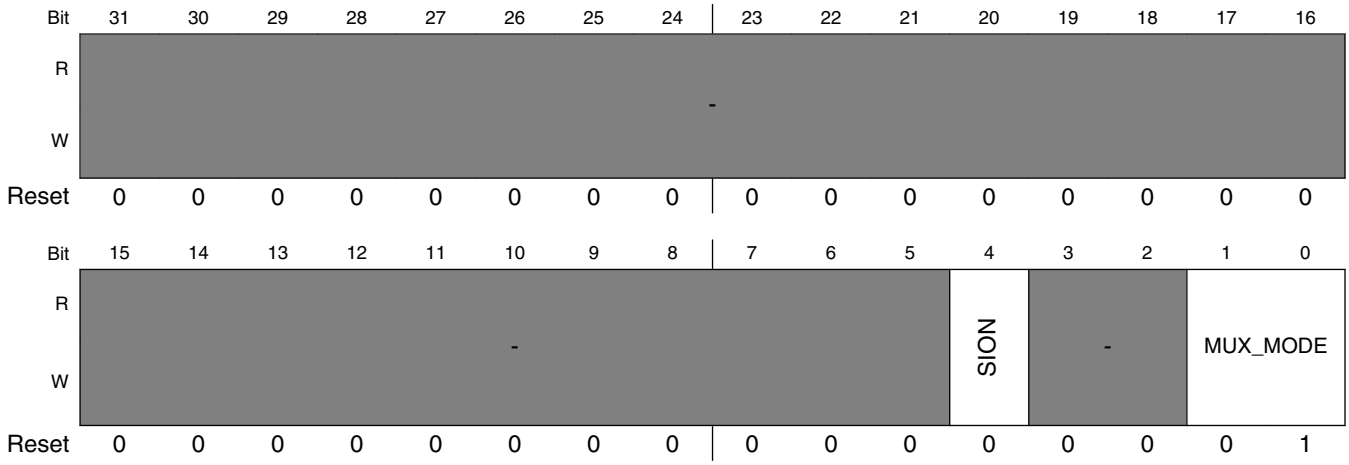
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD2_D1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_D1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD2_D1. **NOTE:** Pad SD2_D1 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: DAT1 of instance: esdhc2. 01 Select mux mode: ALT1 mux port: GPIO[9] of instance: gpio5. 10 Select mux mode: ALT2 mux port: DATA[1] of instance: mshc. 11 Select mux mode: ALT3 mux port: ROW[4] of instance: kpp.

35.4.68 SW_MUX_CTL_PAD_SD2_D2 (IOMUXC_SMUXC_PSD2_D2)

Address: IOMUXC_SMUXC_PSD2_D2 is 53FA_8000h base + 10Ch offset = 53FA_810Ch



IOMUXC_SMUXC_PSD2_D2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_D2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD2_D2. **NOTE:** Pad SD2_D2 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: DAT2 of instance: esdhc2. 01 Select mux mode: ALT1 mux port: GPIO[10] of instance: gpio5. 10 Select mux mode: ALT2 mux port: DATA[2] of instance: mshc. 11 Select mux mode: ALT3 mux port: COL[5] of instance: kpp.

35.4.69 SW_MUX_CTL_PAD_SD2_D3 (IOMUXC_SMUXC_PSD2_D3)

Address: IOMUXC_SMUXC_PSD2_D3 is 53FA_8000h base + 110h offset = 53FA_8110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

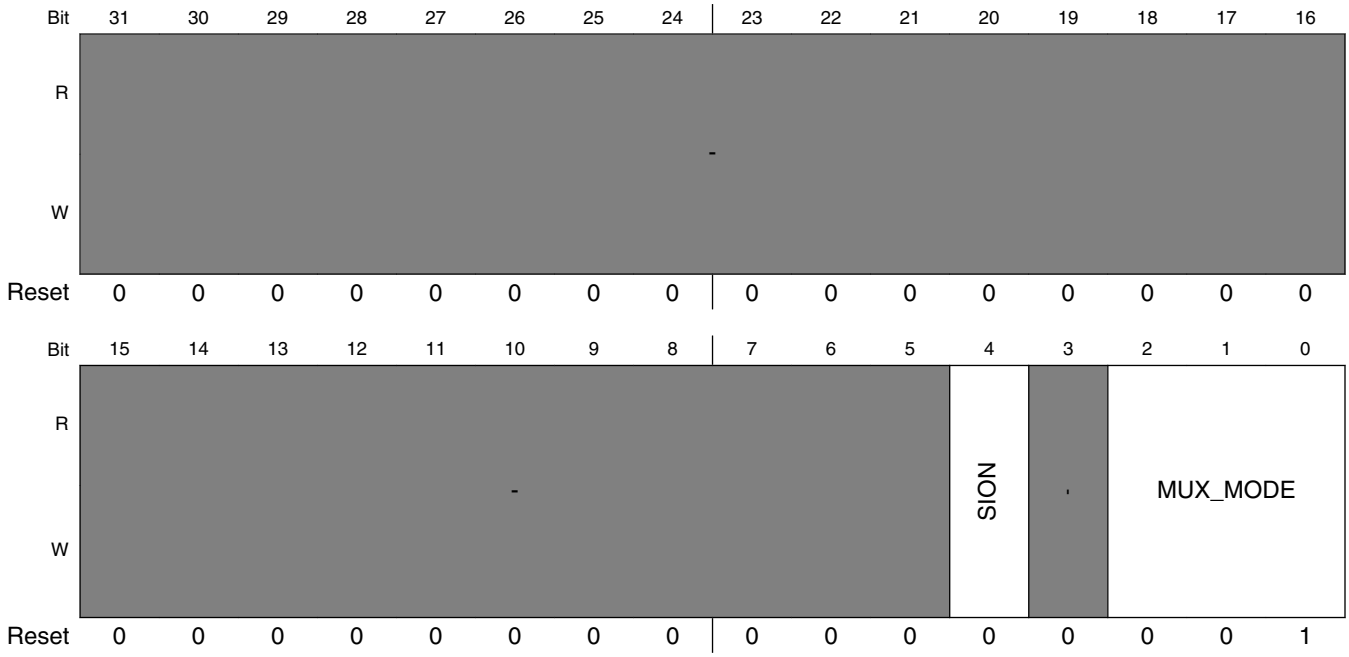
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD2_D3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_D3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD2_D3. **NOTE:** Pad SD2_D3 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: DAT3 of instance: esdhc2. 01 Select mux mode: ALT1 mux port: GPIO[11] of instance: gpio5. 10 Select mux mode: ALT2 mux port: DATA[3] of instance: mshc. 11 Select mux mode: ALT3 mux port: ROW[5] of instance: kpp.

35.4.70 SW_MUX_CTL_PAD_SD2_D4 (IOMUXC_SMUXC_PSD2_D4)

Address: IOMUXC_SMUXC_PSD2_D4 is 53FA_8000h base + 114h offset = 53FA_8114h

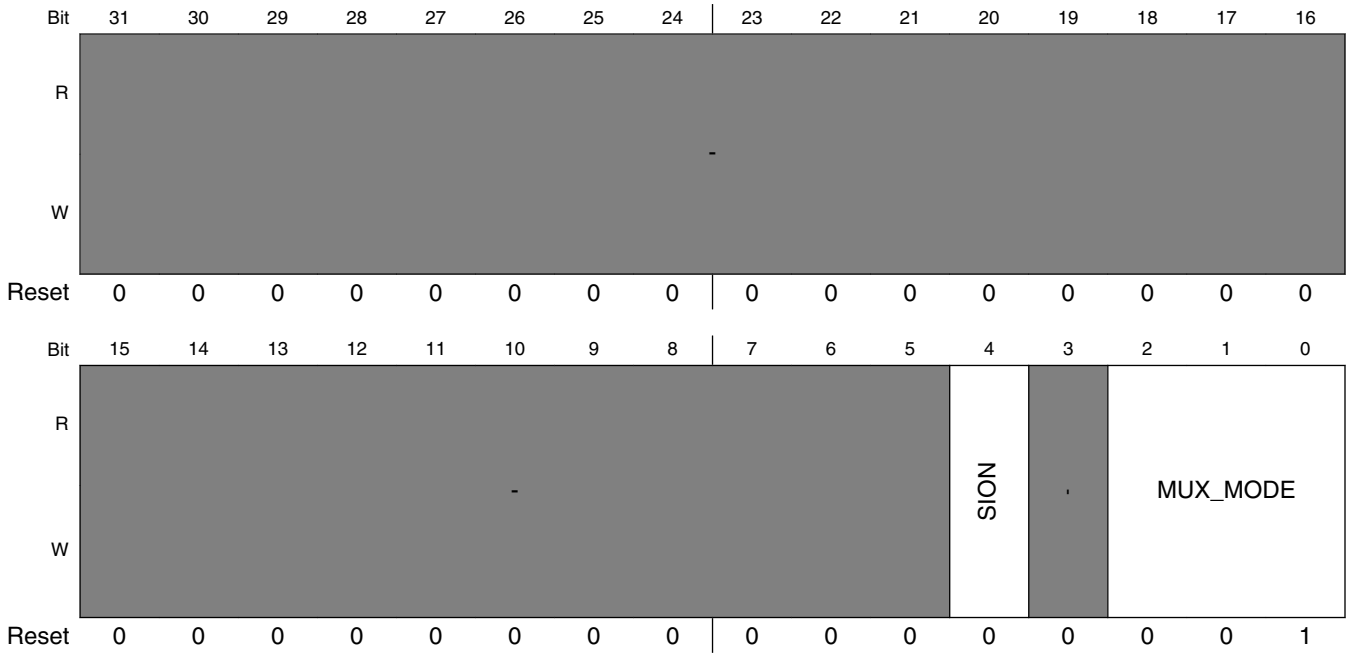


IOMUXC_SMUXC_PSD2_D4 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_D4. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD2_D4. **NOTE:** Pad SD2_D4 is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT for mode ALT3. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_0_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: DAT4 of instance: esdhc2. 001 Select mux mode: ALT1 mux port: GPIO[12] of instance: gpio5. 010 Select mux mode: ALT2 mux port: AUD4_RXFS of instance: audmux. 011 Select mux mode: ALT3 mux port: COL[6] of instance: kpp. 100 Select mux mode: ALT4 mux port: D[0] of instance: weimv2. 111 Select mux mode: ALT7 mux port: CCM_OUT_0 of instance: ccm.

35.4.71 SW_MUX_CTL_PAD_SD2_D5 (IOMUXC_SMUXC_PSD2_D5)

Address: IOMUXC_SMUXC_PSD2_D5 is 53FA_8000h base + 118h offset = 53FA_8118h

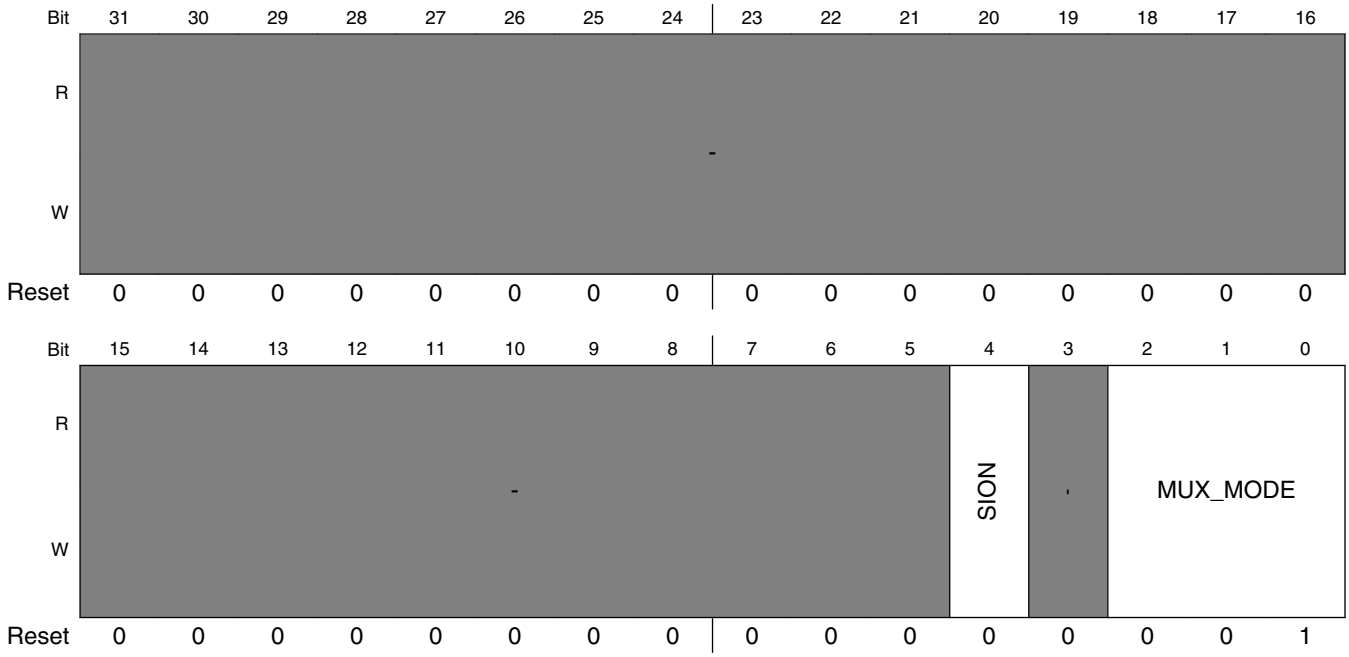


IOMUXC_SMUXC_PSD2_D5 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_D5. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD2_D5. **NOTE:** Pad SD2_D5 is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT for mode ALT3. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_1_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: DAT5 of instance: esdhc2. 001 Select mux mode: ALT1 mux port: GPIO[13] of instance: gpio5. 010 Select mux mode: ALT2 mux port: AUD4_RXC of instance: audmux. 011 Select mux mode: ALT3 mux port: ROW[6] of instance: kpp. 100 Select mux mode: ALT4 mux port: D[1] of instance: weimv2. 111 Select mux mode: ALT7 mux port: CCM_OUT_1 of instance: ccm.

35.4.72 SW_MUX_CTL_PAD_SD2_D6 (IOMUXC_SMUXC_PSD2_D6)

Address: IOMUXC_SMUXC_PSD2_D6 is 53FA_8000h base + 11Ch offset = 53FA_811Ch

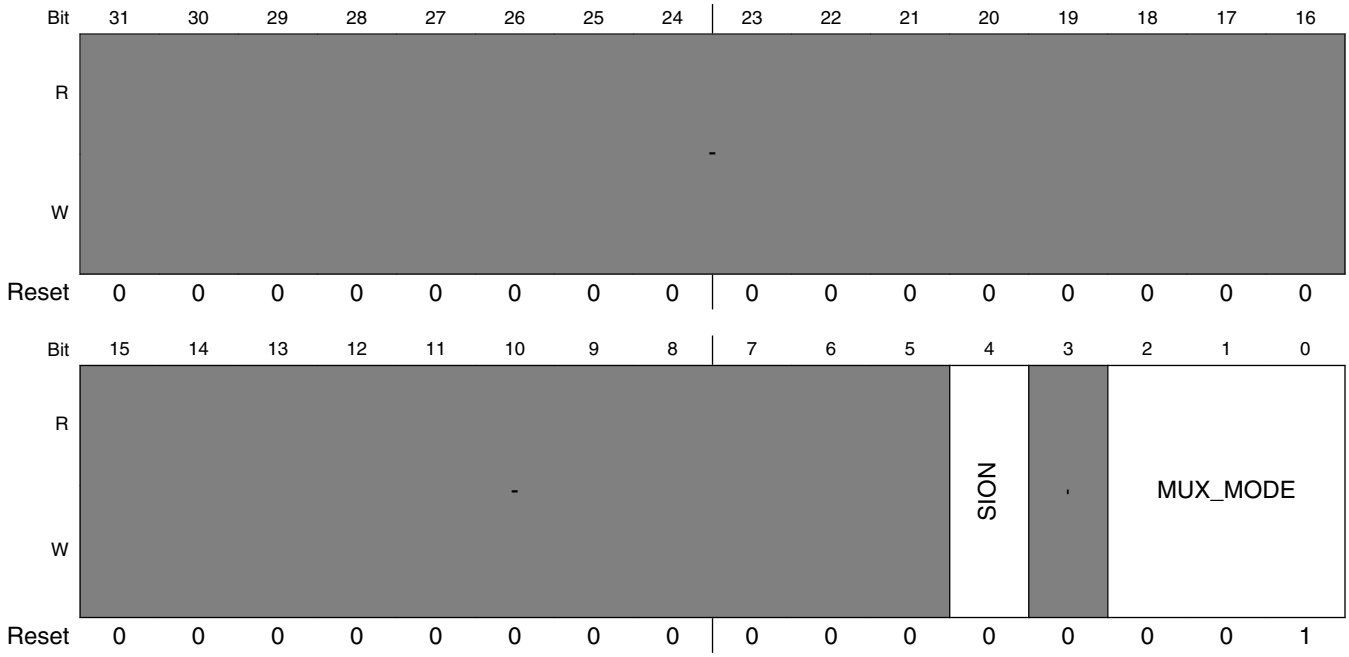


IOMUXC_SMUXC_PSD2_D6 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_D6. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD2_D6. **NOTE:** Pad SD2_D6 is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT for mode ALT3. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_2_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: DAT6 of instance: esdhc2. 001 Select mux mode: ALT1 mux port: GPIO[14] of instance: gpio5. 010 Select mux mode: ALT2 mux port: AUD4_RXD of instance: audmux. 011 Select mux mode: ALT3 mux port: COL[7] of instance: kpp. 100 Select mux mode: ALT4 mux port: D[2] of instance: weimv2. 111 Select mux mode: ALT7 mux port: CCM_OUT_2 of instance: ccm.

35.4.73 SW_MUX_CTL_PAD_SD2_D7 (IOMUXC_SMUXC_PSD2_D7)

Address: IOMUXC_SMUXC_PSD2_D7 is 53FA_8000h base + 120h offset = 53FA_8120h

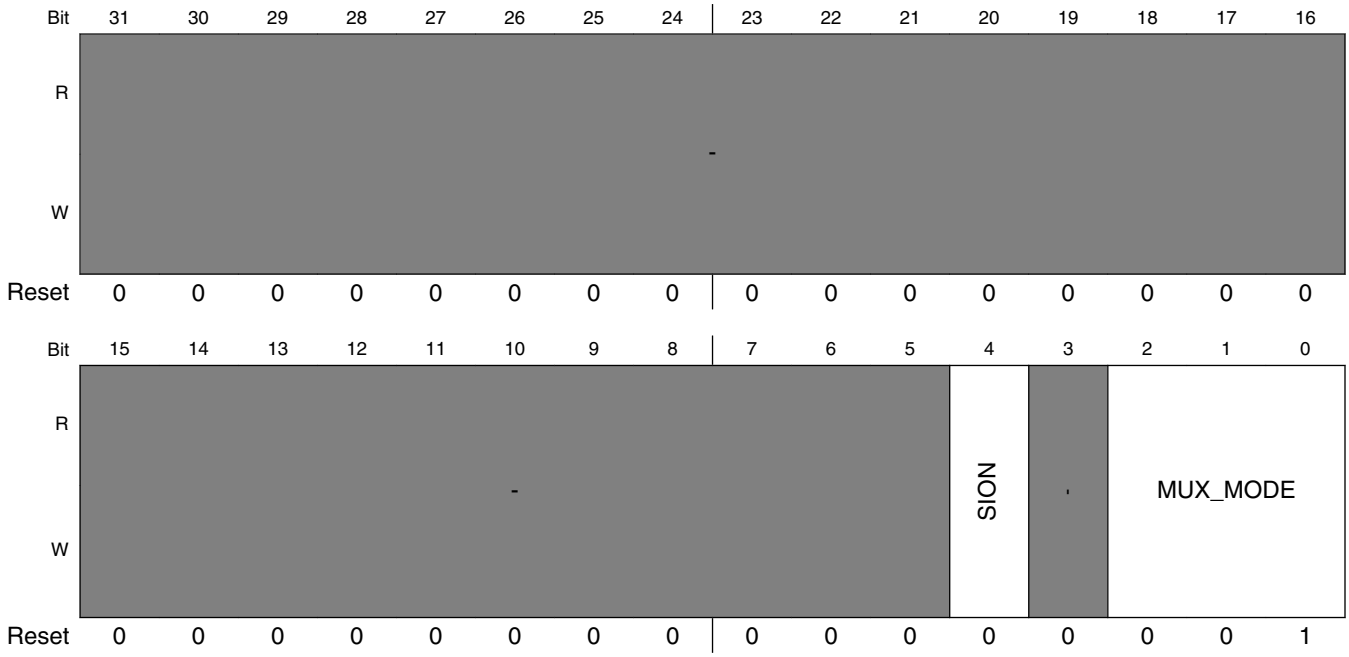


IOMUXC_SMUXC_PSD2_D7 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_D7. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD2_D7. **NOTE:** Pad SD2_D7 is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT for mode ALT3. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_3_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: DAT7 of instance: esdhc2. 001 Select mux mode: ALT1 mux port: GPIO[15] of instance: gpio5. 010 Select mux mode: ALT2 mux port: AUD4_TXFS of instance: audmux. 011 Select mux mode: ALT3 mux port: ROW[7] of instance: kpp. 100 Select mux mode: ALT4 mux port: D[3] of instance: weimv2. 111 Select mux mode: ALT7 mux port: STOP of instance: ccm.

35.4.74 SW_MUX_CTL_PAD_SD2_WP (IOMUXC_SMUXC_PSD2_WP)

Address: IOMUXC_SMUXC_PSD2_WP is 53FA_8000h base + 124h offset = 53FA_8124h



IOMUXC_SMUXC_PSD2_WP field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_WP. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: SD2_WP. **NOTE:** Pad SD2_WP is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_ESDHC2_IPP_WP_ON_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_4_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: WP of instance: esdhc2. 001 Select mux mode: ALT1 mux port: GPIO[16] of instance: gpio5. 010 Select mux mode: ALT2 mux port: AUD4_TXD of instance: audmux. 100 Select mux mode: ALT4 mux port: D[4] of instance: weimv2. 111 Select mux mode: ALT7 mux port: WAIT of instance: ccm.

35.4.75 SW_MUX_CTL_PAD_SD2_CD (IOMUXC_SMUXC_PSD2_CD)

Address: IOMUXC_SMUXC_PSD2_CD is 53FA_8000h base + 128h offset = 53FA_8128h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

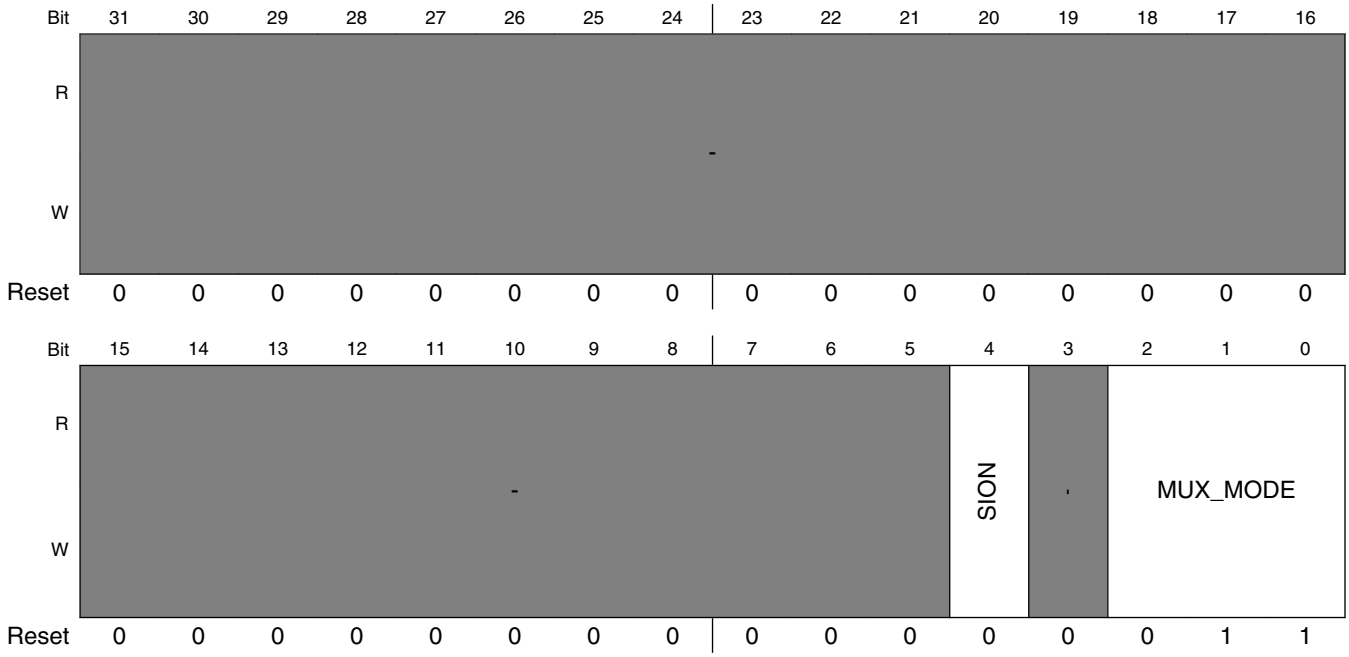
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD2_CD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD2_CD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: SD2_CD. **NOTE:** Pad SD2_CD is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT for mode ALT2. • Config Register IOMUXC_ESDHC2_IPP_CARD_DET_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_5_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: CD of instance: esdhc2. 001 Select mux mode: ALT1 mux port: GPIO[17] of instance: gpio5. 010 Select mux mode: ALT2 mux port: AUD4_TXC of instance: audmux. 100 Select mux mode: ALT4 mux port: D[5] of instance: weimv2. 111 Select mux mode: ALT7 mux port: REF_EN_B of instance: ccm.

35.4.76 SW_MUX_CTL_PAD_DISP_D0 (IOMUXC_SMUXC_PDISP_D0)

Address: IOMUXC_SMUXC_PDISP_D0 is 53FA_8000h base + 12Ch offset = 53FA_812Ch

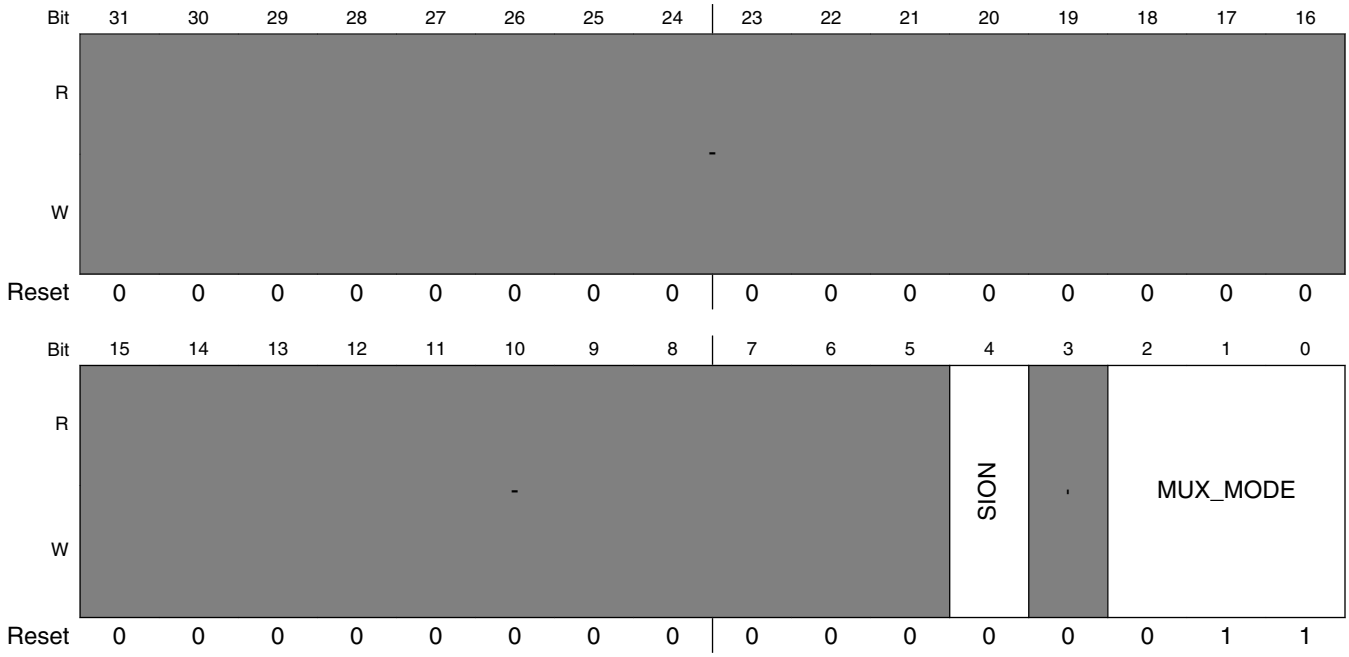


IOMUXC_SMUXC_PDISP_D0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_D0. **NOTE:** Pad DISP_D0 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_0_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_FEC_FEC_TX_CLK_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: DAT[0] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[0] of instance: gpio2. 010 Select mux mode: ALT2 mux port: TX_CLK of instance: fec. 011 Select mux mode: ALT3 mux port: A[16] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[0] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[0] of instance: usbphy1.

35.4.77 SW_MUX_CTL_PAD_DISP_D1 (IOMUXC_SMUXC_PDISP_D1)

Address: IOMUXC_SMUXC_PDISP_D1 is 53FA_8000h base + 130h offset = 53FA_8130h

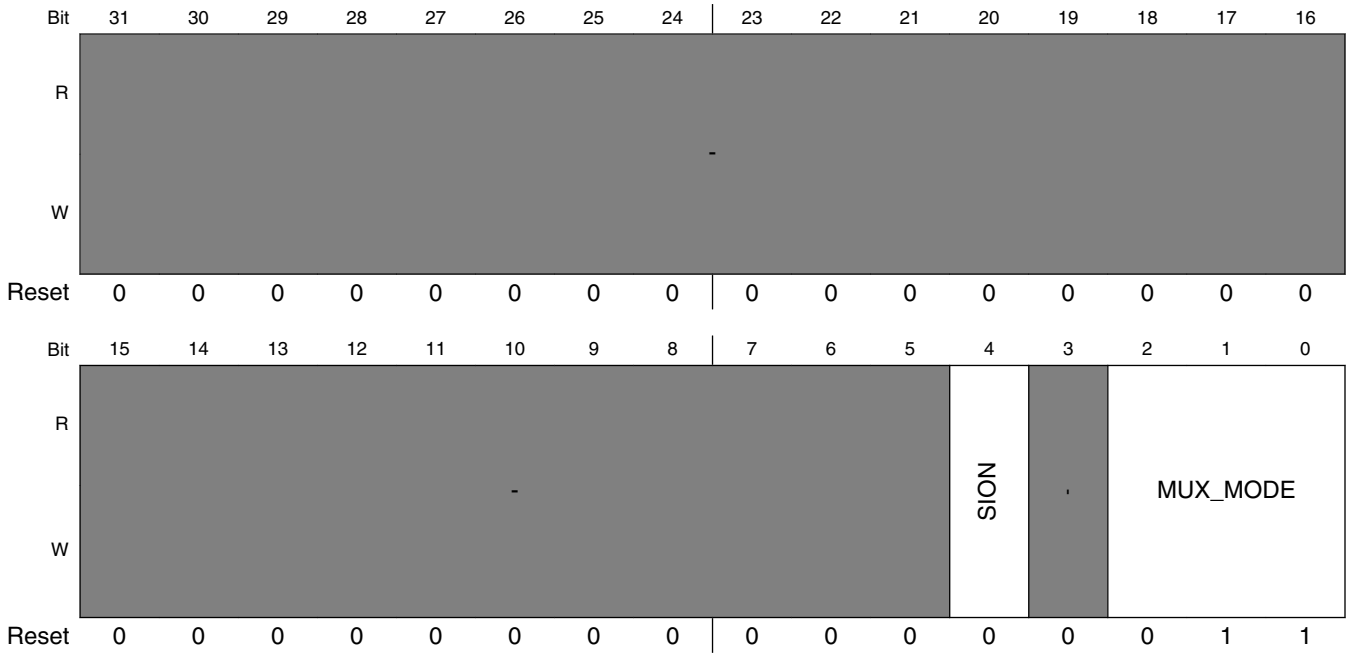


IOMUXC_SMUXC_PDISP_D1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_D1. **NOTE:** Pad DISP_D1 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_1_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: DAT[1] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[1] of instance: gpio2. 010 Select mux mode: ALT2 mux port: RX_ER of instance: fec. 011 Select mux mode: ALT3 mux port: A[17] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[1] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[1] of instance: usbphy1.

35.4.78 SW_MUX_CTL_PAD_DISP_D2 (IOMUXC_SMUXC_PDISP_D2)

Address: IOMUXC_SMUXC_PDISP_D2 is 53FA_8000h base + 134h offset = 53FA_8134h

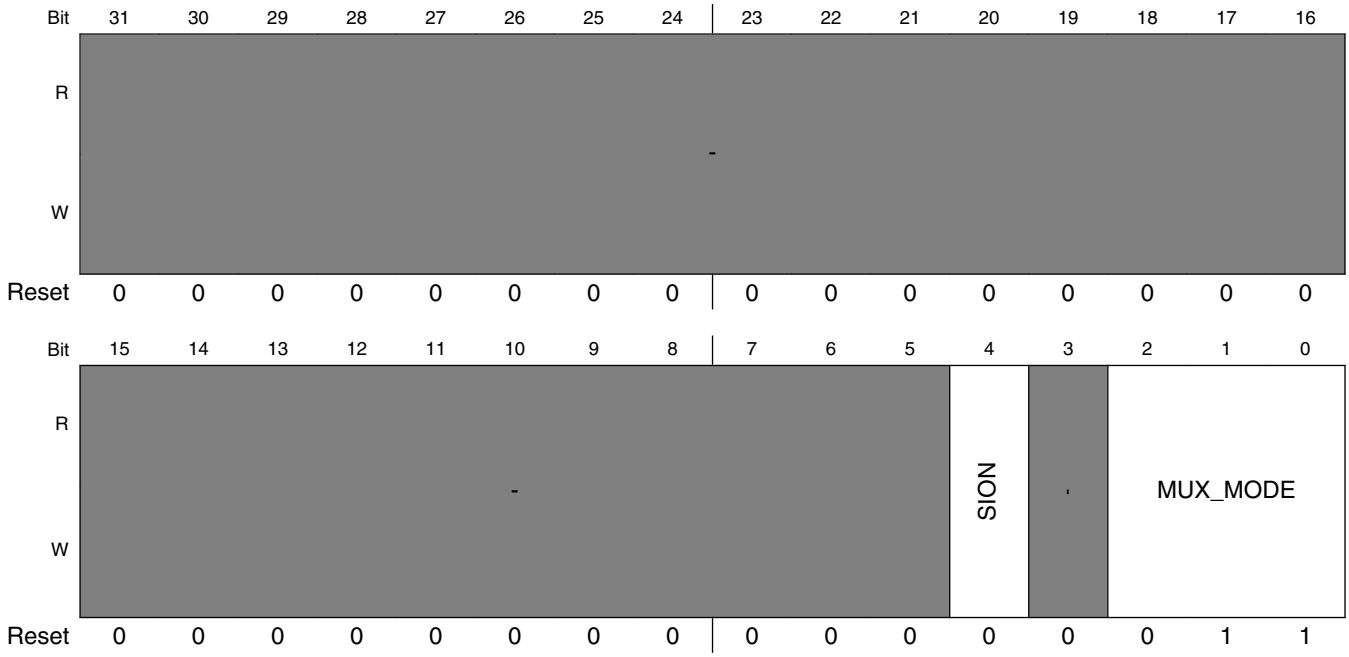


IOMUXC_SMUXC_PDISP_D2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_D2. **NOTE:** Pad DISP_D2 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_2_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_FEC_FEC_RX_DV_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: DAT[2] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[2] of instance: gpio2. 010 Select mux mode: ALT2 mux port: RX_DV of instance: fec. 011 Select mux mode: ALT3 mux port: A[18] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[2] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[2] of instance: usbphy1.

35.4.79 SW_MUX_CTL_PAD_DISP_D3 (IOMUXC_SMUXC_PDISP_D3)

Address: IOMUXC_SMUXC_PDISP_D3 is 53FA_8000h base + 138h offset = 53FA_8138h

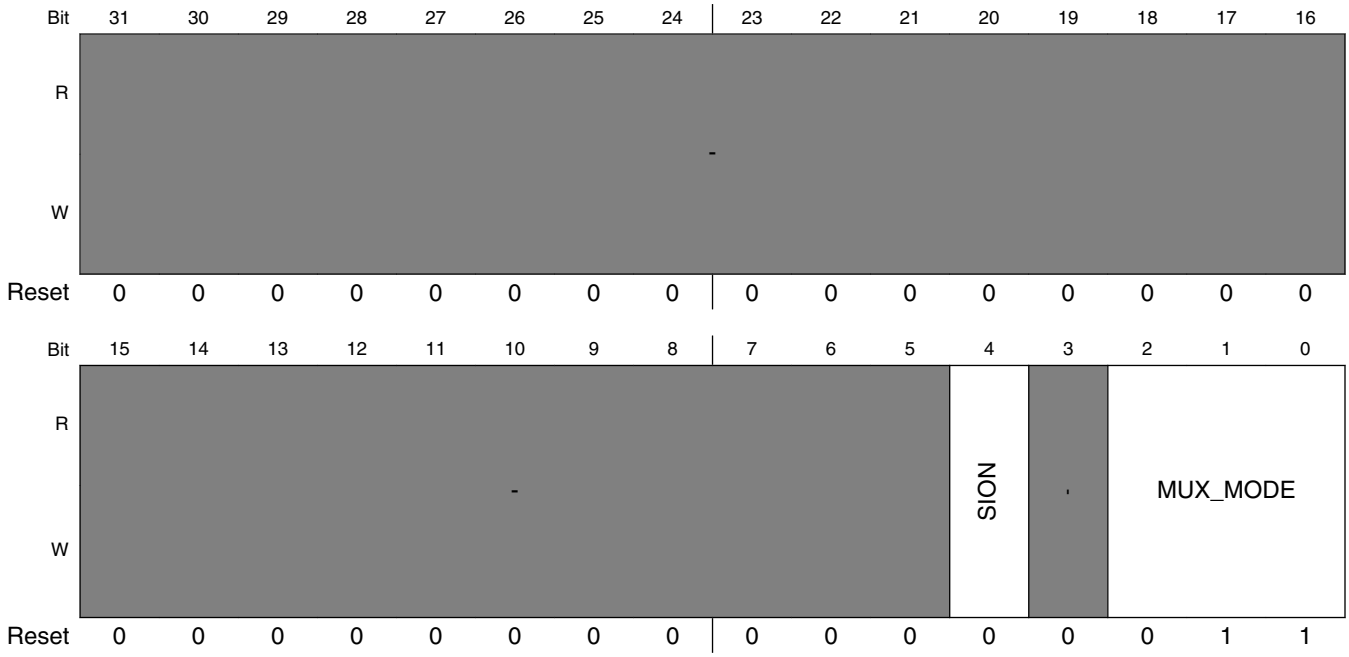


IOMUXC_SMUXC_PDISP_D3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: DISP_D3. **NOTE:** Pad DISP_D3 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_3_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_FEC_FEC_COL_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_FEC_FEC_RDATA_1_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: DAT[3] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[3] of instance: gpio2. 010 Select mux mode: ALT2 mux port: RDATA[1] of instance: fec. 011 Select mux mode: ALT3 mux port: A[19] of instance: weimv2. 100 Select mux mode: ALT4 mux port: COL of instance: fec. 110 Select mux mode: ALT6 mux port: DEBUG_PC[3] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[3] of instance: usbphy1.

35.4.80 SW_MUX_CTL_PAD_DISP_D4 (IOMUXC_SMUXC_PDISP_D4)

Address: IOMUXC_SMUXC_PDISP_D4 is 53FA_8000h base + 13Ch offset = 53FA_813Ch

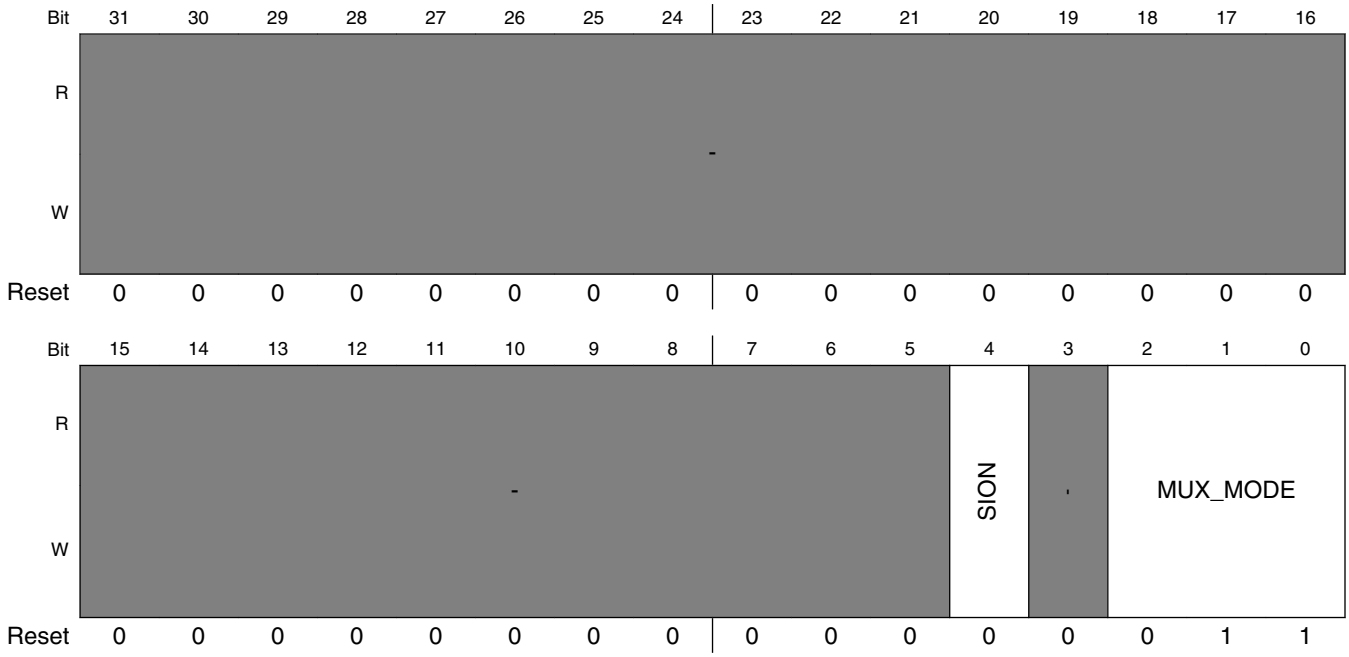


IOMUXC_SMUXC_PDISP_D4 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D4. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_D4. **NOTE:** Pad DISP_D4 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_4_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_FEC_FEC_RDATA_0_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: DAT[4] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[4] of instance: gpio2. 010 Select mux mode: ALT2 mux port: RDATA[0] of instance: fec. 011 Select mux mode: ALT3 mux port: A[20] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[4] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[4] of instance: usbphy1.

35.4.81 SW_MUX_CTL_PAD_DISP_D5 (IOMUXC_SMUXC_PDISP_D5)

Address: IOMUXC_SMUXC_PDISP_D5 is 53FA_8000h base + 140h offset = 53FA_8140h

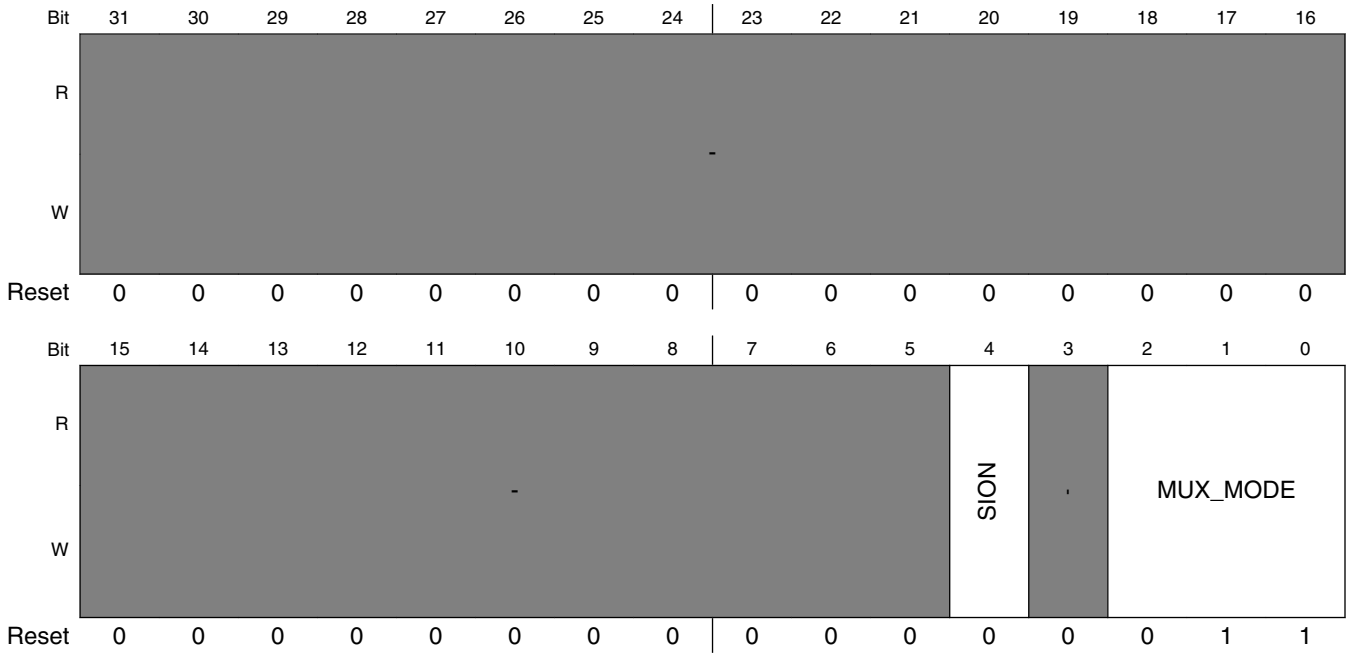


IOMUXC_SMUXC_PDISP_D5 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D5. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_D5. **NOTE:** Pad DISP_D5 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_5_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: DAT[5] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[5] of instance: gpio2. 010 Select mux mode: ALT2 mux port: TX_EN of instance: fec. 011 Select mux mode: ALT3 mux port: A[21] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[5] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[5] of instance: usbphy1.

35.4.82 SW_MUX_CTL_PAD_DISP_D6 (IOMUXC_SMUXC_PDISP_D6)

Address: IOMUXC_SMUXC_PDISP_D6 is 53FA_8000h base + 144h offset = 53FA_8144h

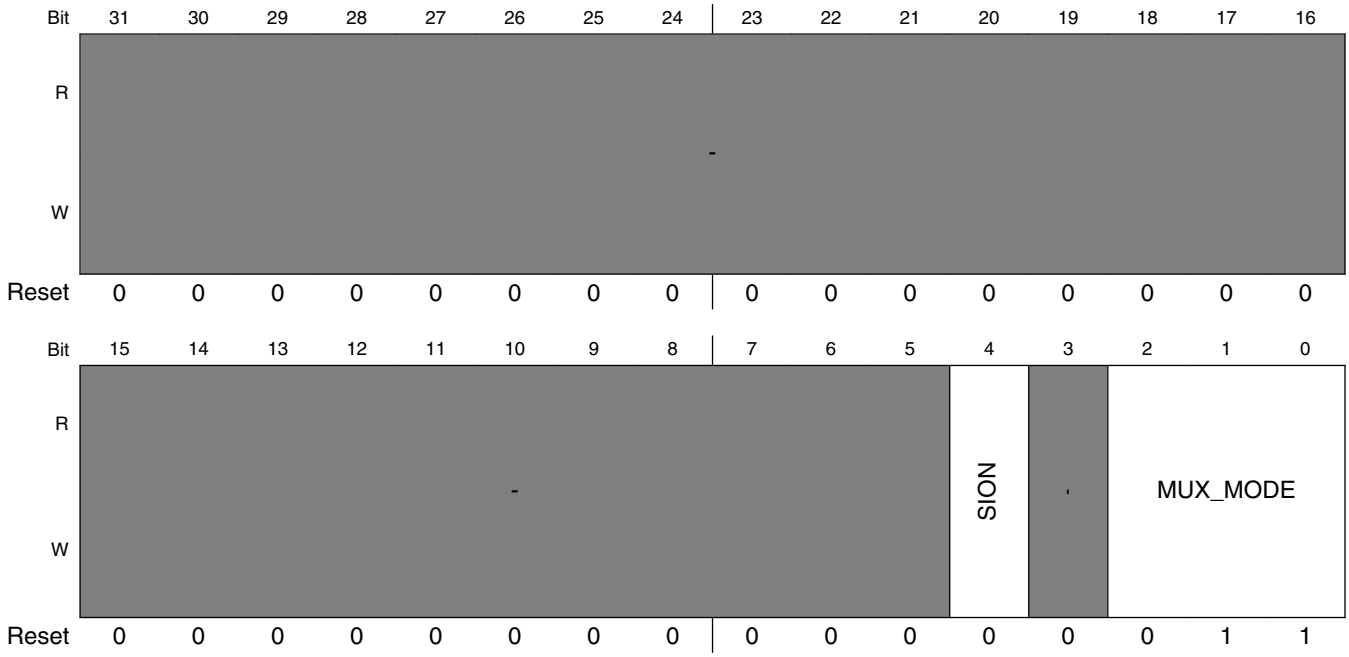


IOMUXC_SMUXC_PDISP_D6 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D6. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: DISP_D6. **NOTE:** Pad DISP_D6 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_6_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT for mode ALT4. 000 Select mux mode: ALT0 mux port: DAT[6] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[6] of instance: gpio2. 010 Select mux mode: ALT2 mux port: TDATA[1] of instance: fec. 011 Select mux mode: ALT3 mux port: A[22] of instance: weimv2. 100 Select mux mode: ALT4 mux port: RX_CLK of instance: fec. 110 Select mux mode: ALT6 mux port: DEBUG_PC[6] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[6] of instance: usbphy1.

35.4.83 SW_MUX_CTL_PAD_DISP_D7 (IOMUXC_SMUXC_PDISP_D7)

Address: IOMUXC_SMUXC_PDISP_D7 is 53FA_8000h base + 148h offset = 53FA_8148h

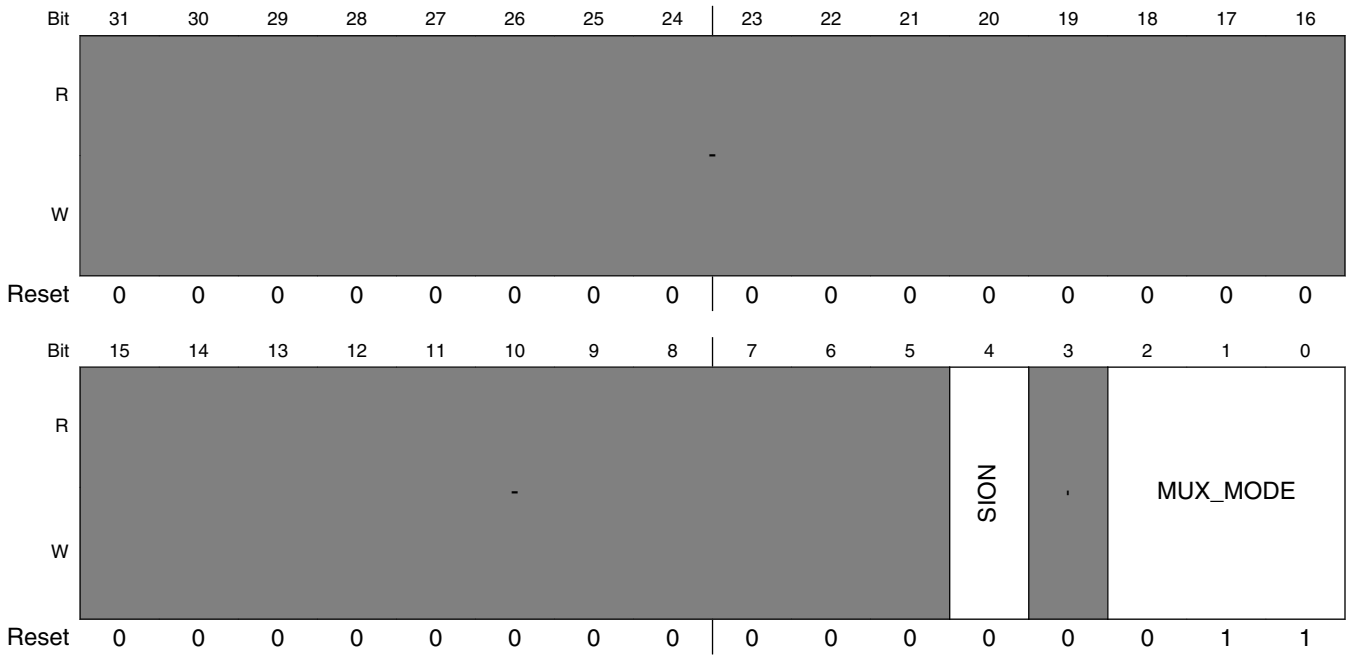


IOMUXC_SMUXC_PDISP_D7 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D7. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_D7. **NOTE:** Pad DISP_D7 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_7_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: DAT[7] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[7] of instance: gpio2. 010 Select mux mode: ALT2 mux port: TDATA[0] of instance: fec. 011 Select mux mode: ALT3 mux port: A[23] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[7] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[7] of instance: usbphy1.

35.4.84 SW_MUX_CTL_PAD_DISP_WR (IOMUXC_SMUXC_PDISP_WR)

Address: IOMUXC_SMUXC_PDISP_WR is 53FA_8000h base + 14Ch offset = 53FA_814Ch

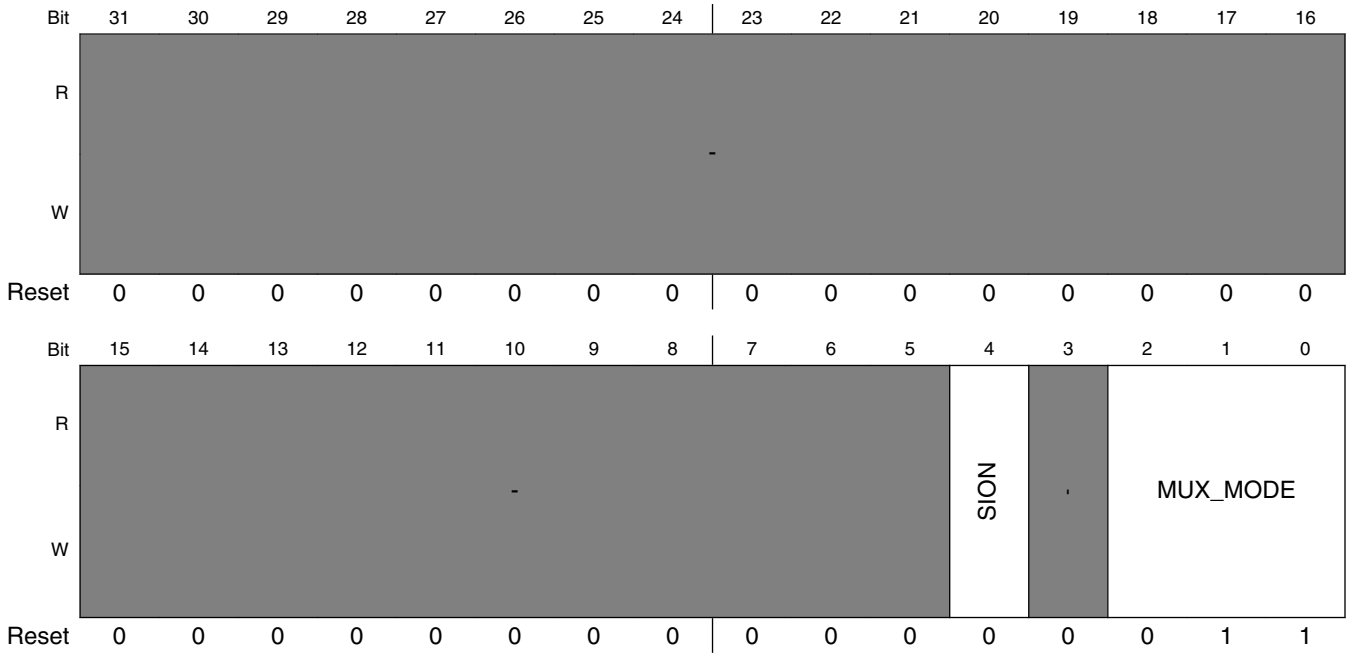


IOMUXC_SMUXC_PDISP_WR field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_WR. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_WR. 000 Select mux mode: ALT0 mux port: WR_RWN of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[16] of instance: gpio2. 010 Select mux mode: ALT2 mux port: DOTCLK of instance: elcdif. 011 Select mux mode: ALT3 mux port: A[24] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[8] of instance: sdma. 111 Select mux mode: ALT7 mux port: AVALID of instance: usbphy1.

35.4.85 SW_MUX_CTL_PAD_DISP_RD (IOMUXC_SMUXC_PDISP_RD)

Address: IOMUXC_SMUXC_PDISP_RD is 53FA_8000h base + 150h offset = 53FA_8150h

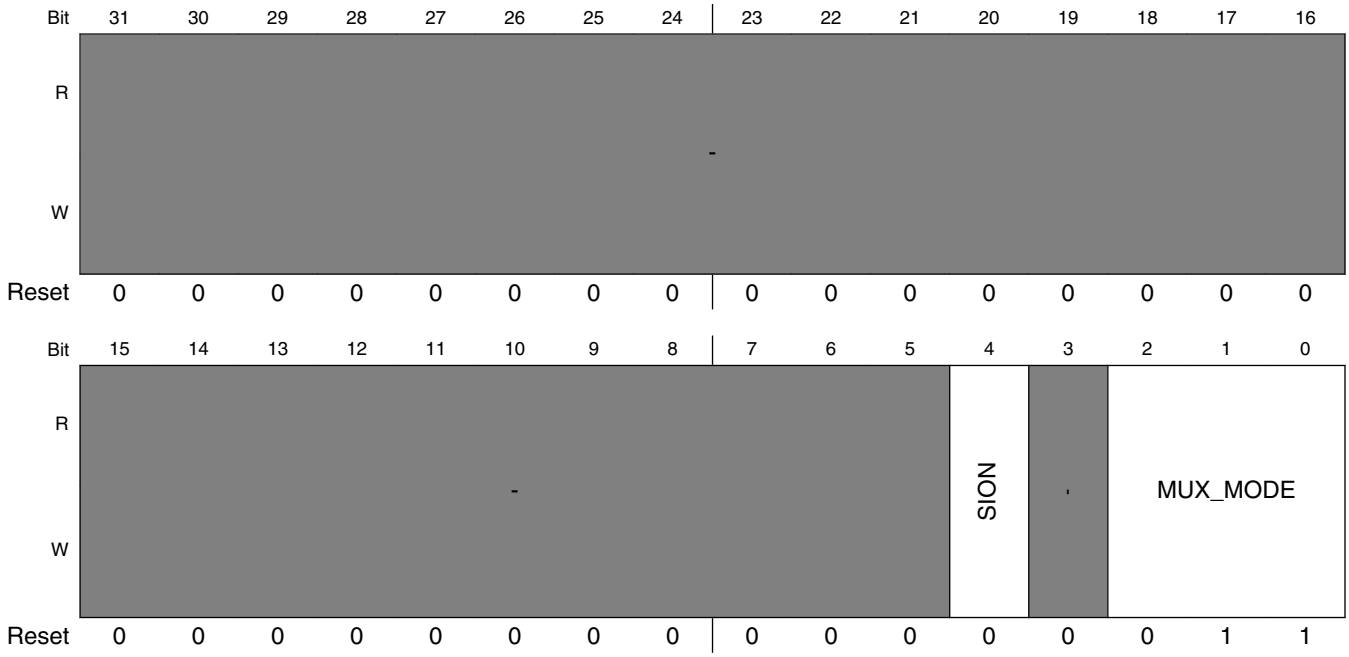


IOMUXC_SMUXC_PDISP_RD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_RD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_RD. 000 Select mux mode: ALT0 mux port: RD_E of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[19] of instance: gpio2. 010 Select mux mode: ALT2 mux port: ENABLE of instance: elcdif. 011 Select mux mode: ALT3 mux port: A[25] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[9] of instance: sdma. 111 Select mux mode: ALT7 mux port: BVALID of instance: usbphy1.

35.4.86 SW_MUX_CTL_PAD_DISP_RS (IOMUXC_SMUXC_PDISP_RS)

Address: IOMUXC_SMUXC_PDISP_RS is 53FA_8000h base + 154h offset = 53FA_8154h

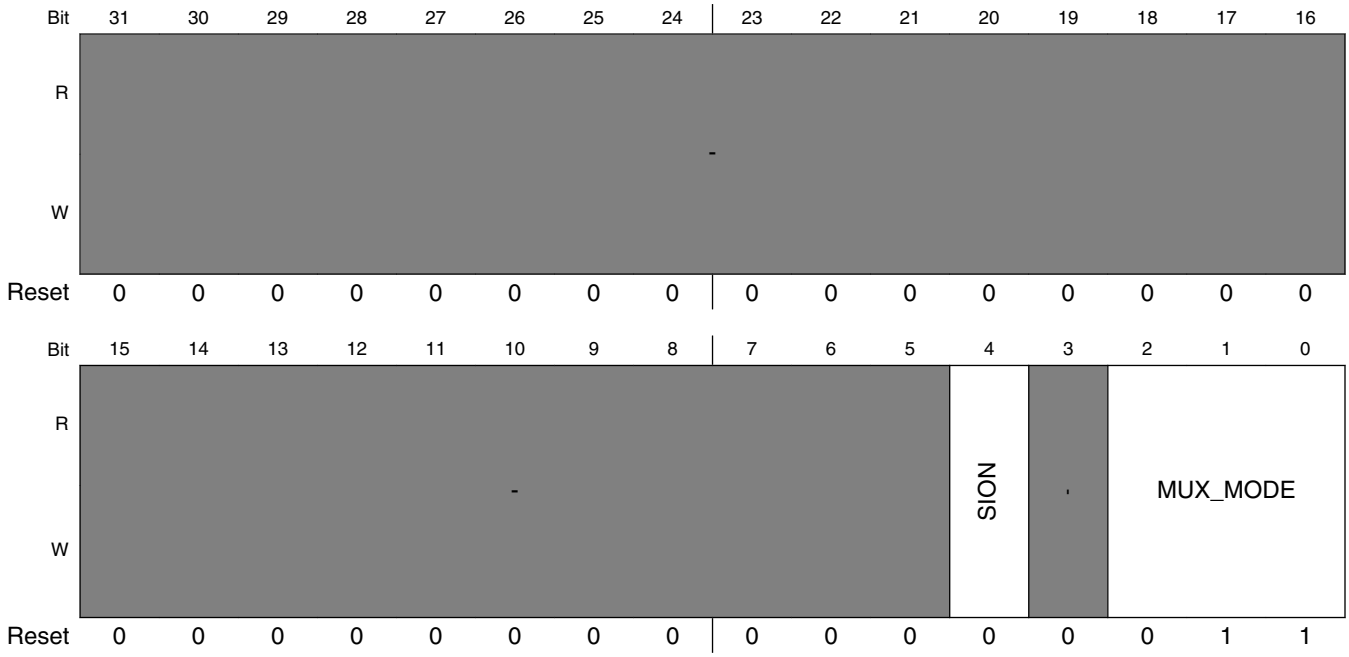


IOMUXC_SMUXC_PDISP_RS field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_RS. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: DISP_RS. **NOTE:** Pad DISP_RS is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_VSYNC_I_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: RS of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[17] of instance: gpio2. 010 Select mux mode: ALT2 mux port: VSYNC of instance: elcdif. 011 Select mux mode: ALT3 mux port: A[26] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[10] of instance: sdma. 111 Select mux mode: ALT7 mux port: ENDSESSION of instance: usbphy1.

35.4.87 SW_MUX_CTL_PAD_DISP_CS (IOMUXC_SMUXC_PDISP_CS)

Address: IOMUXC_SMUXC_PDISP_CS is 53FA_8000h base + 158h offset = 53FA_8158h

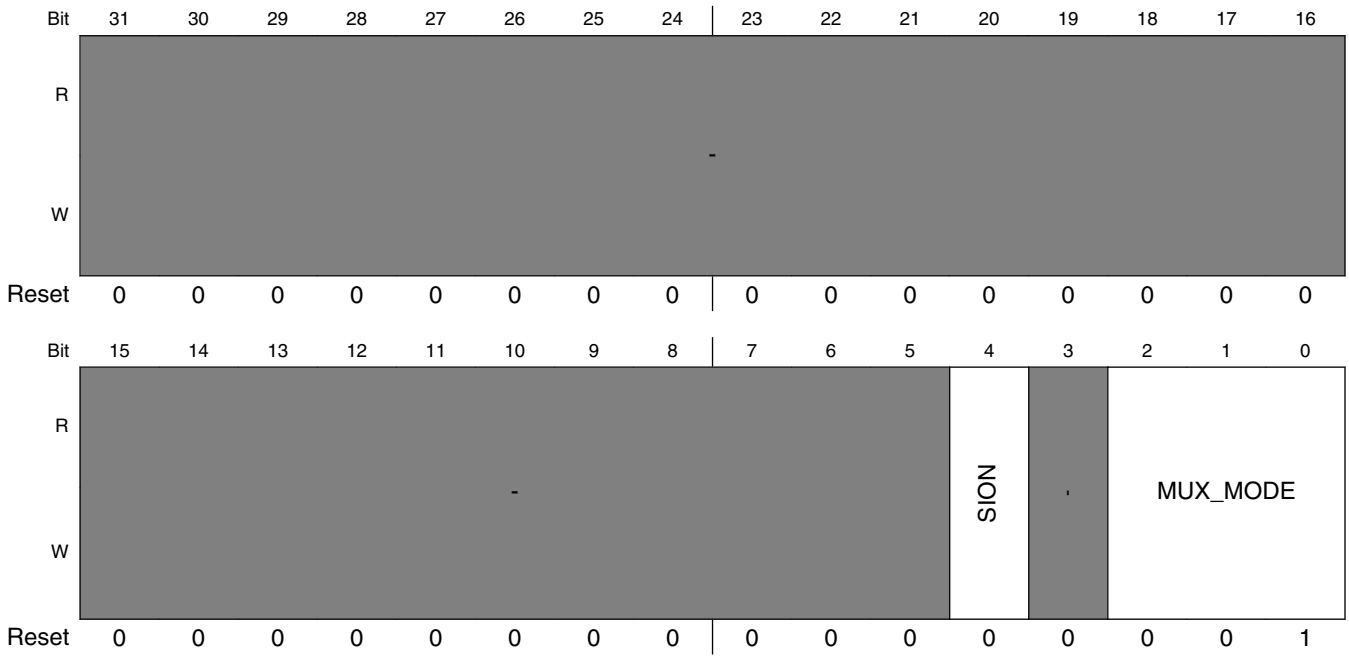


IOMUXC_SMUXC_PDISP_CS field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_CS. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: DISP_CS. **NOTE:** Pad DISP_CS is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_BUSY_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: CS of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[21] of instance: gpio2. 010 Select mux mode: ALT2 mux port: HSYNC of instance: elcdif. 011 Select mux mode: ALT3 mux port: A[27] of instance: weimv2. 100 Select mux mode: ALT4 mux port: CS[3] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[11] of instance: sdma. 111 Select mux mode: ALT7 mux port: IDDIG of instance: usbphy1.

35.4.88 SW_MUX_CTL_PAD_DISP_BUSY (IOMUXC_SMUXC_PDISP_BUSY)

Address: IOMUXC_SMUXC_PDISP_BUSY is 53FA_8000h base + 15Ch offset = 53FA_815Ch



IOMUXC_SMUXC_PDISP_BUSY field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_BUSY. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: DISP_BUSY. **NOTE:** Pad DISP_BUSY is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_BUSY_SELECT_INPUT for mode ALT0. 000 Select mux mode: ALT0 mux port: HSYNC of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[18] of instance: gpio2. 100 Select mux mode: ALT4 mux port: CS[3] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[12] of instance: sdma. 111 Select mux mode: ALT7 mux port: HOSTDISCONNECT of instance: usbphy1.

35.4.89 SW_MUX_CTL_PAD_DISP_RESET (IOMUXC_SMUXC_PDISP_RESET)

Address: IOMUXC_SMUXC_PDISP_RESET is 53FA_8000h base + 160h offset = 53FA_8160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

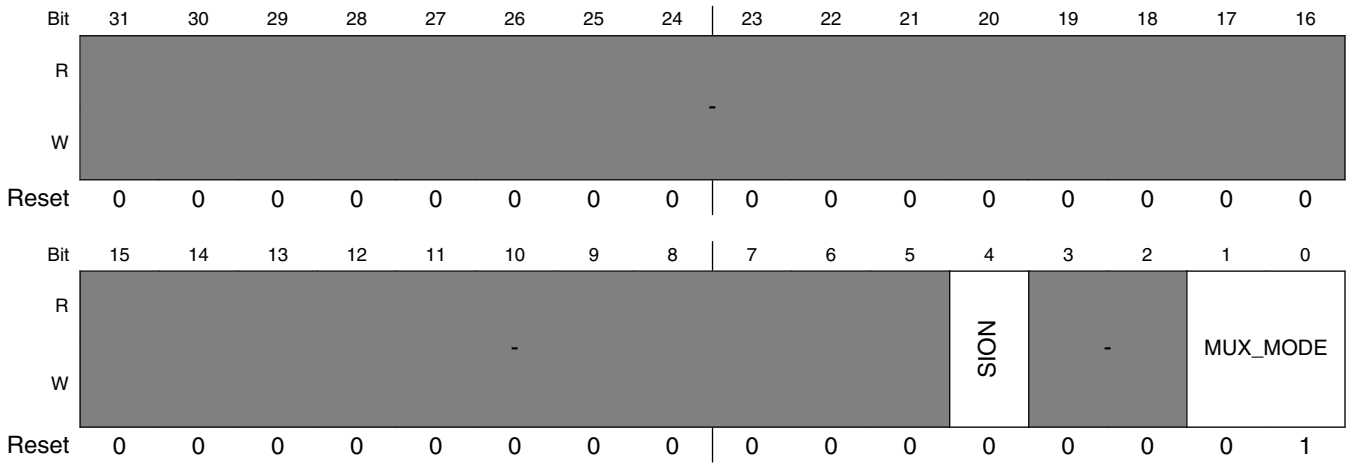
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PDISP_RESET field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_RESET. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: DISP_RESET. 000 Select mux mode: ALT0 mux port: RESET of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[20] of instance: gpio2. 100 Select mux mode: ALT4 mux port: CS[3] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_PC[13] of instance: sdma. 111 Select mux mode: ALT7 mux port: BISTOK of instance: usbphy1.

35.4.90 SW_MUX_CTL_PAD_SD3_CMD (IOMUXC_SMUXC_PSD3_CMD)

Address: IOMUXC_SMUXC_PSD3_CMD is 53FA_8000h base + 164h offset = 53FA_8164h

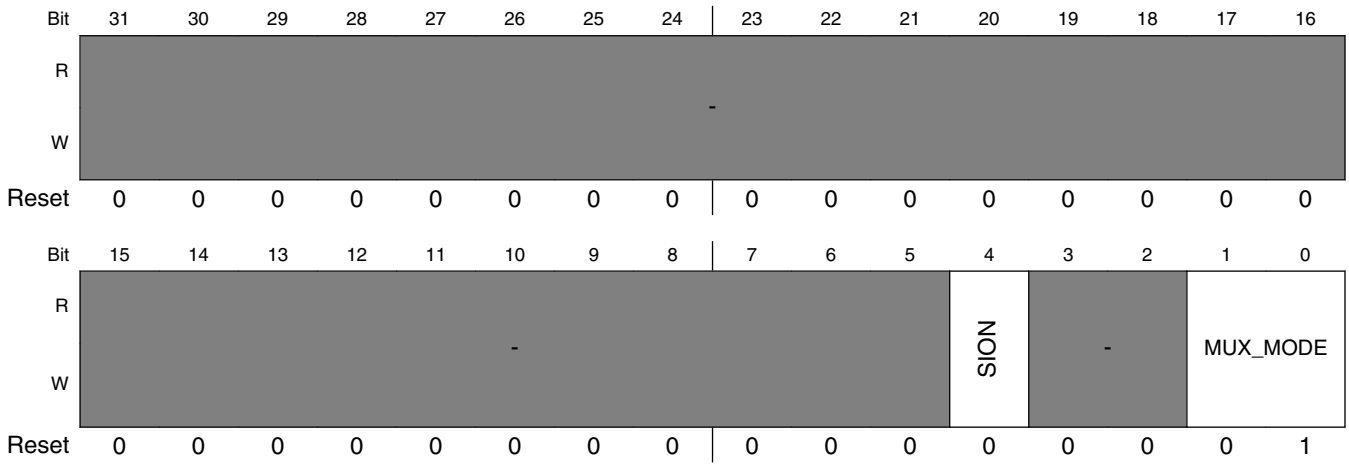


IOMUXC_SMUXC_PSD3_CMD field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_CMD. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD3_CMD. 00 Select mux mode: ALT0 mux port: CMD of instance: esdhc3. 01 Select mux mode: ALT1 mux port: GPIO[18] of instance: gpio5. 10 Select mux mode: ALT2 mux port: WRN of instance: rawnand. 11 Select mux mode: ALT3 mux port: CMD of instance: ssp.

35.4.91 SW_MUX_CTL_PAD_SD3_CLK (IOMUXC_SMUXC_PSD3_CLK)

Address: IOMUXC_SMUXC_PSD3_CLK is 53FA_8000h base + 168h offset = 53FA_8168h

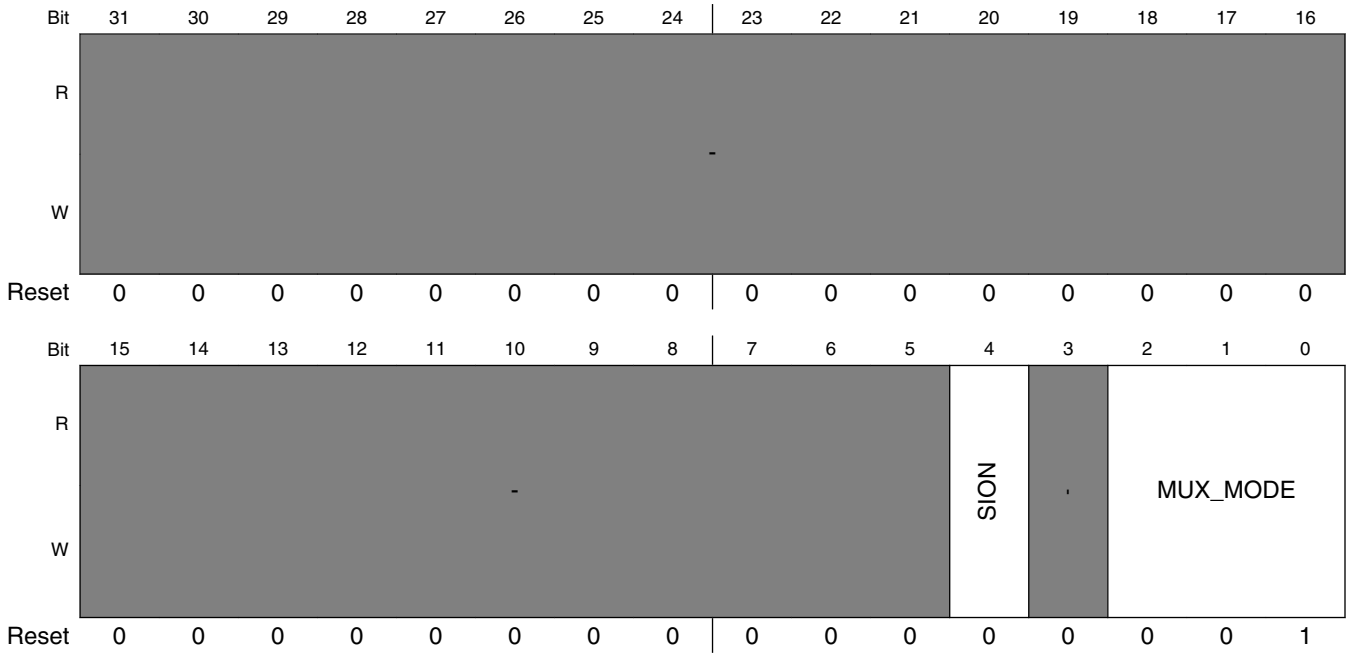


IOMUXC_SMUXC_PSD3_CLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_CLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD3_CLK. 00 Select mux mode: ALT0 mux port: CLK of instance: esdhc3. 01 Select mux mode: ALT1 mux port: GPIO[19] of instance: gpio5. 10 Select mux mode: ALT2 mux port: RDN of instance: rawnand. 11 Select mux mode: ALT3 mux port: CLK of instance: ssp.

35.4.92 SW_MUX_CTL_PAD_SD3_D0 (IOMUXC_SMUXC_PSD3_D0)

Address: IOMUXC_SMUXC_PSD3_D0 is 53FA_8000h base + 16Ch offset = 53FA_816Ch

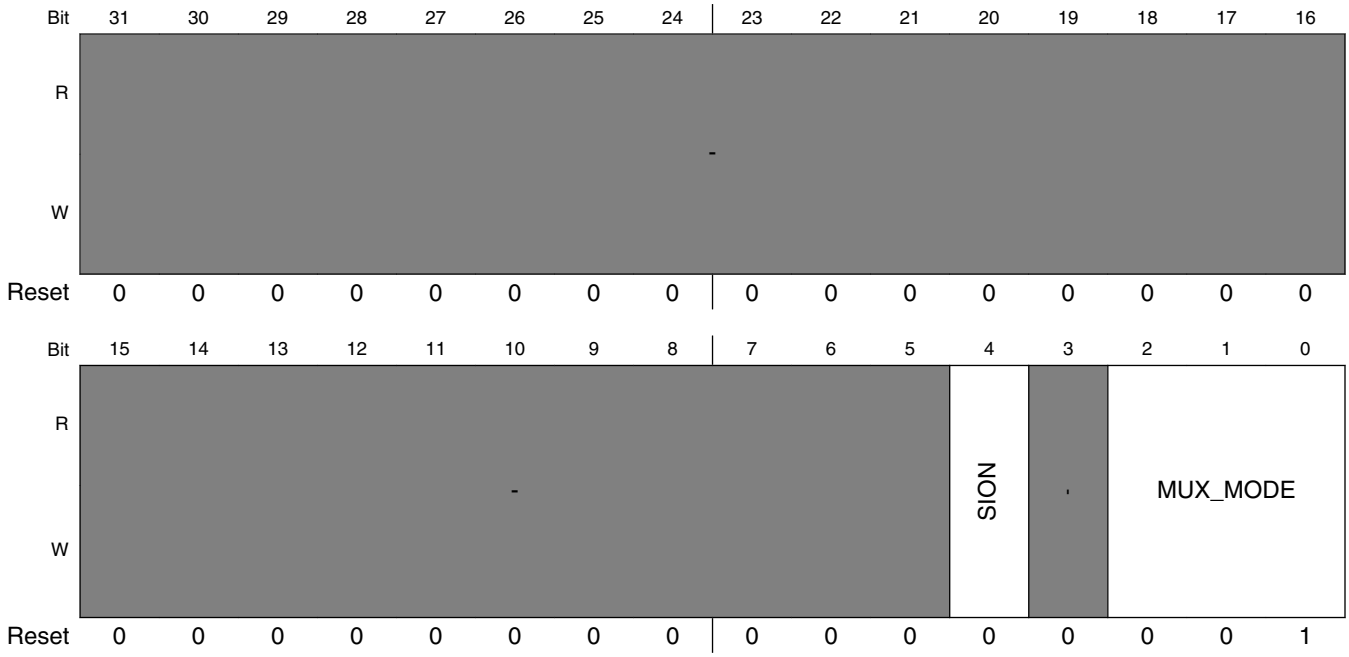


IOMUXC_SMUXC_PSD3_D0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_D0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: SD3_D0. **NOTE:** Pad SD3_D0 is involved in Daisy Chain. • Config Register IOMUXC_CCM_PLL1_BYPASS_CLK_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: DAT0 of instance: esdhc3. 001 Select mux mode: ALT1 mux port: GPIO[20] of instance: gpio5. 010 Select mux mode: ALT2 mux port: D[4] of instance: rawnand. 011 Select mux mode: ALT3 mux port: D0 of instance: ssp. 111 Select mux mode: ALT7 mux port: PLL1_BYP of instance: ccm.

35.4.93 SW_MUX_CTL_PAD_SD3_D1 (IOMUXC_SMUXC_PSD3_D1)

Address: IOMUXC_SMUXC_PSD3_D1 is 53FA_8000h base + 170h offset = 53FA_8170h

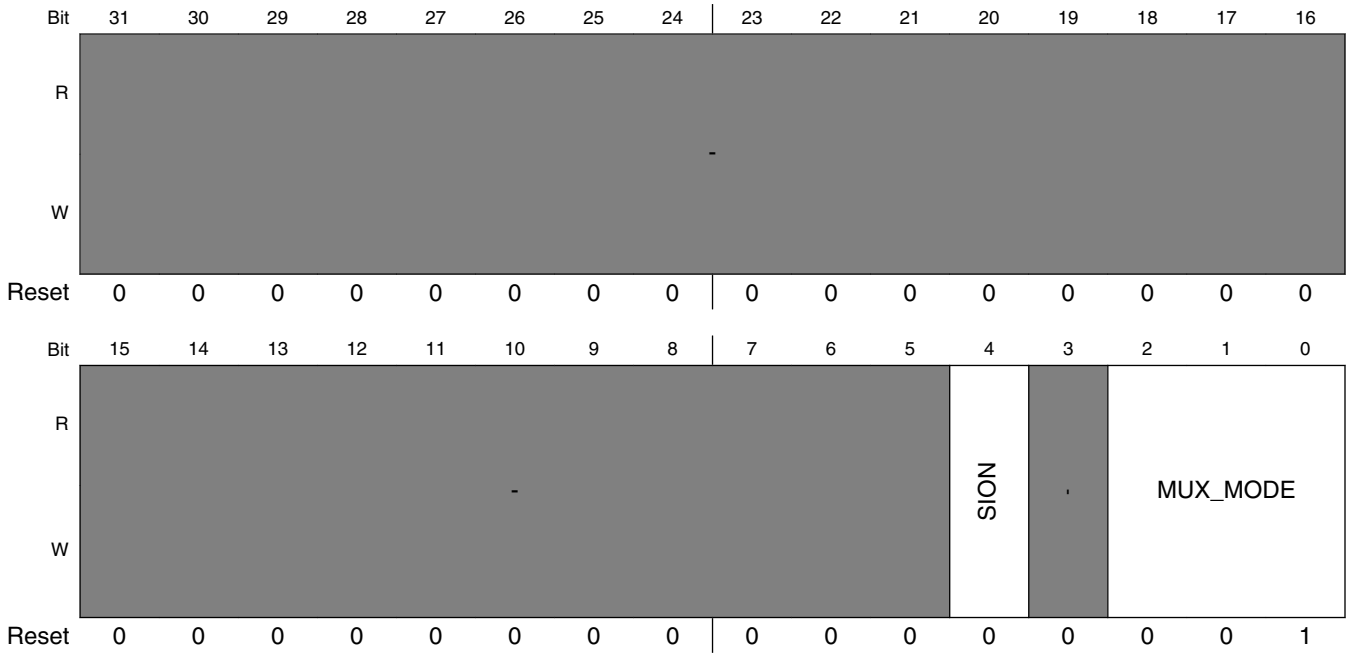


IOMUXC_SMUXC_PSD3_D1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_D1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: SD3_D1. **NOTE:** Pad SD3_D1 is involved in Daisy Chain. • Config Register IOMUXC_CCM_PLL2_BYPASS_CLK_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: DAT1 of instance: esdhc3. 001 Select mux mode: ALT1 mux port: GPIO[21] of instance: gpio5. 010 Select mux mode: ALT2 mux port: D[5] of instance: rawnand. 011 Select mux mode: ALT3 mux port: D1 of instance: ssp. 111 Select mux mode: ALT7 mux port: PLL2_BYP of instance: ccm.

35.4.94 SW_MUX_CTL_PAD_SD3_D2 (IOMUXC_SMUXC_PSD3_D2)

Address: IOMUXC_SMUXC_PSD3_D2 is 53FA_8000h base + 174h offset = 53FA_8174h



IOMUXC_SMUXC_PSD3_D2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_D2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: SD3_D2. **NOTE:** Pad SD3_D2 is involved in Daisy Chain. • Config Register IOMUXC_CCM_PLL3_BYPASS_CLK_SELECT_INPUT for mode ALT7. 000 Select mux mode: ALT0 mux port: DAT2 of instance: esdhc3. 001 Select mux mode: ALT1 mux port: GPIO[22] of instance: gpio5. 010 Select mux mode: ALT2 mux port: D[6] of instance: rawnand. 011 Select mux mode: ALT3 mux port: D2 of instance: ssp. 111 Select mux mode: ALT7 mux port: PLL3_BYP of instance: ccm.

35.4.95 SW_MUX_CTL_PAD_SD3_D3 (IOMUXC_SMUXC_PSD3_D3)

Address: IOMUXC_SMUXC_PSD3_D3 is 53FA_8000h base + 178h offset = 53FA_8178h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

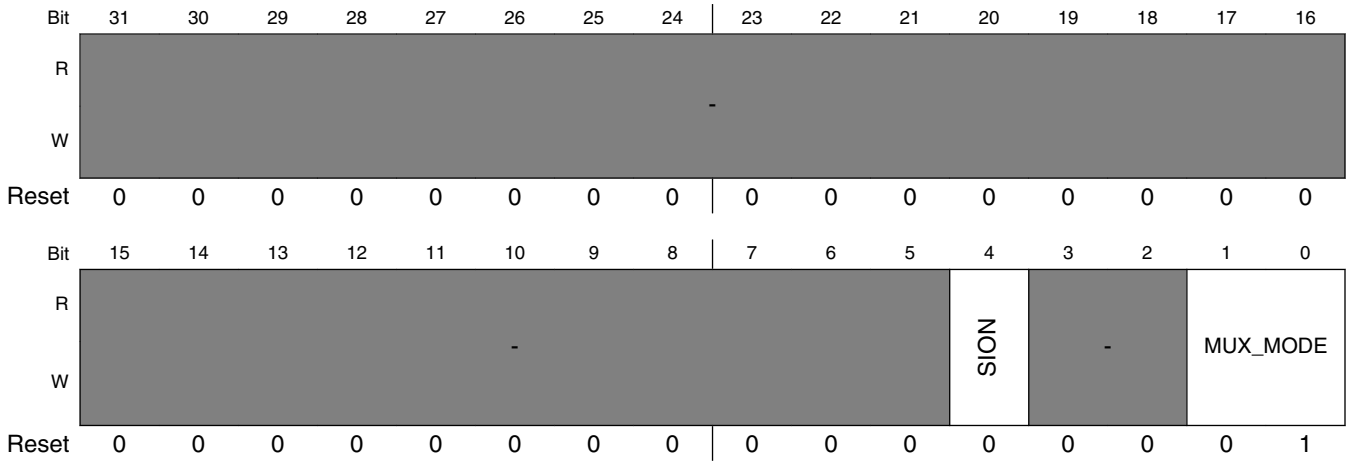
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD3_D3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_D3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD3_D3. 00 Select mux mode: ALT0 mux port: DAT3 of instance: esdhc3. 01 Select mux mode: ALT1 mux port: GPIO[23] of instance: gpio5. 10 Select mux mode: ALT2 mux port: D[7] of instance: rawnand. 11 Select mux mode: ALT3 mux port: D3 of instance: ssp.

35.4.96 SW_MUX_CTL_PAD_SD3_D4 (IOMUXC_SMUXC_PSD3_D4)

Address: IOMUXC_SMUXC_PSD3_D4 is 53FA_8000h base + 17Ch offset = 53FA_817Ch



IOMUXC_SMUXC_PSD3_D4 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_D4. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD3_D4. 00 Select mux mode: ALT0 mux port: DAT4 of instance: esdhc3. 01 Select mux mode: ALT1 mux port: GPIO[24] of instance: gpio5. 10 Select mux mode: ALT2 mux port: D[0] of instance: rawnand. 11 Select mux mode: ALT3 mux port: D4 of instance: ssp.

35.4.97 SW_MUX_CTL_PAD_SD3_D5 (IOMUXC_SMUXC_PSD3_D5)

Address: IOMUXC_SMUXC_PSD3_D5 is 53FA_8000h base + 180h offset = 53FA_8180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

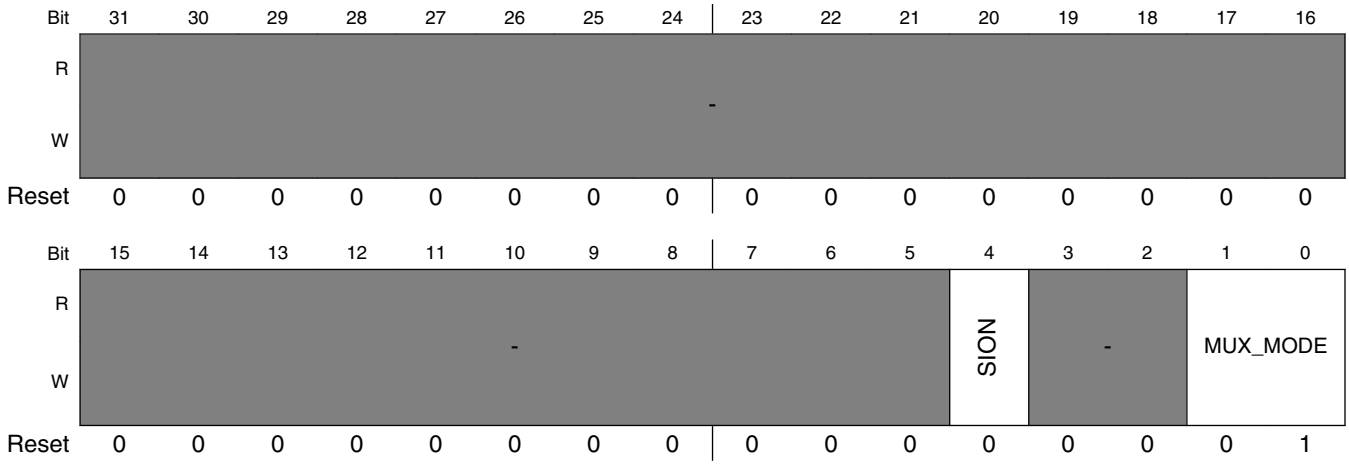
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD3_D5 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_D5. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD3_D5. 00 Select mux mode: ALT0 mux port: DAT5 of instance: esdhc3. 01 Select mux mode: ALT1 mux port: GPIO[25] of instance: gpio5. 10 Select mux mode: ALT2 mux port: D[1] of instance: rawnand. 11 Select mux mode: ALT3 mux port: D5 of instance: ssp.

35.4.98 SW_MUX_CTL_PAD_SD3_D6 (IOMUXC_SMUXC_PSD3_D6)

Address: IOMUXC_SMUXC_PSD3_D6 is 53FA_8000h base + 184h offset = 53FA_8184h



IOMUXC_SMUXC_PSD3_D6 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_D6. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD3_D6. 00 Select mux mode: ALT0 mux port: DAT6 of instance: esdhc3. 01 Select mux mode: ALT1 mux port: GPIO[26] of instance: gpio5. 10 Select mux mode: ALT2 mux port: D[2] of instance: rawnand. 11 Select mux mode: ALT3 mux port: D6 of instance: ssp.

35.4.99 SW_MUX_CTL_PAD_SD3_D7 (IOMUXC_SMUXC_PSD3_D7)

Address: IOMUXC_SMUXC_PSD3_D7 is 53FA_8000h base + 188h offset = 53FA_8188h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

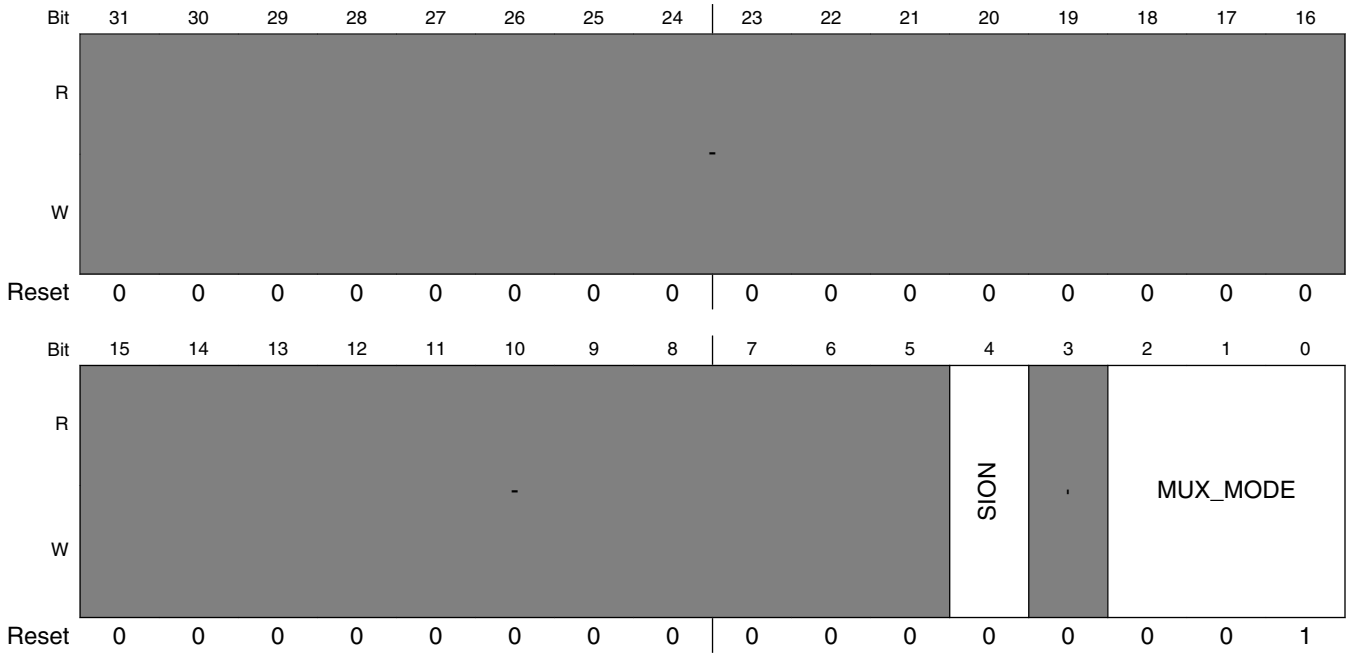
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PSD3_D7 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_D7. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: SD3_D7. 00 Select mux mode: ALT0 mux port: DAT7 of instance: esdhc3. 01 Select mux mode: ALT1 mux port: GPIO[27] of instance: gpio5. 10 Select mux mode: ALT2 mux port: D[3] of instance: rawnand. 11 Select mux mode: ALT3 mux port: D7 of instance: ssp.

35.4.100 SW_MUX_CTL_PAD_SD3_WP (IOMUXC_SMUXC_PSD3_WP)

Address: IOMUXC_SMUXC_PSD3_WP is 53FA_8000h base + 18Ch offset = 53FA_818Ch

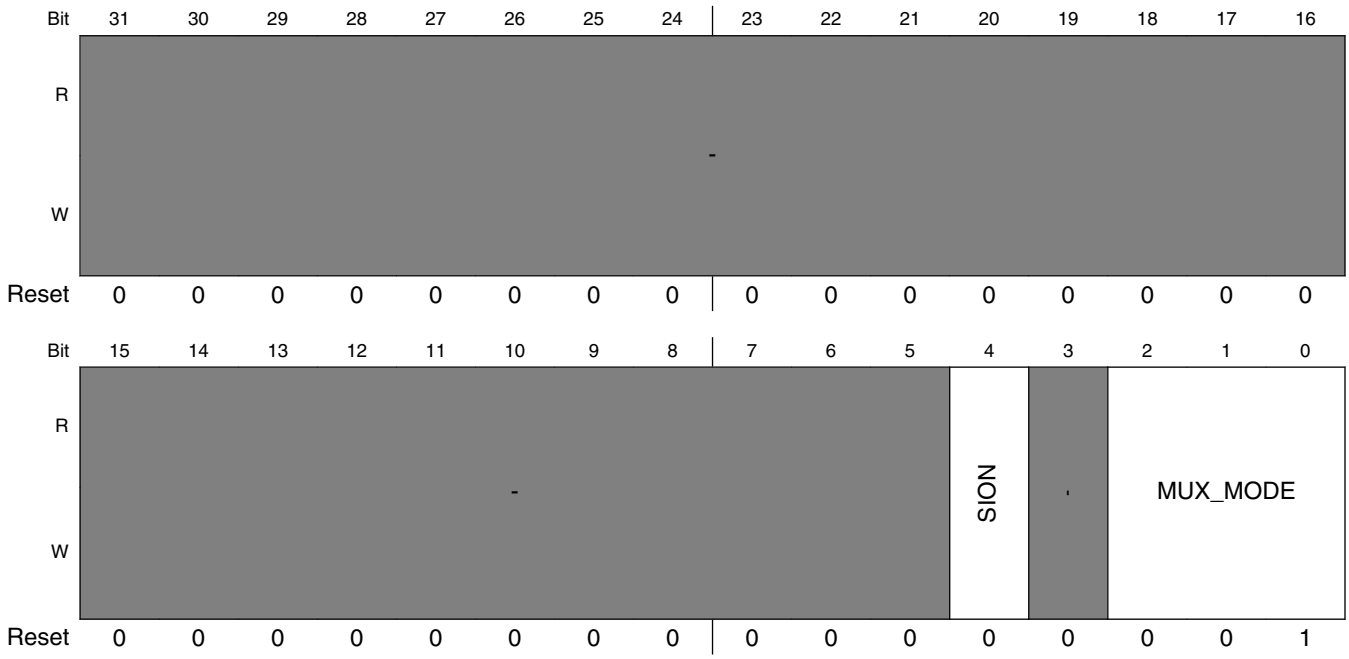


IOMUXC_SMUXC_PSD3_WP field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad SD3_WP. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: SD3_WP. 000 Select mux mode: ALT0 mux port: WP of instance: esdhc3. 001 Select mux mode: ALT1 mux port: GPIO[28] of instance: gpio5. 010 Select mux mode: ALT2 mux port: RESETN of instance: rawnand. 011 Select mux mode: ALT3 mux port: CD of instance: ssp. 100 Select mux mode: ALT4 mux port: LCTL of instance: esdhc4. 101 Select mux mode: ALT5 mux port: CS[3] of instance: weimv2.

35.4.101 SW_MUX_CTL_PAD_DISP_D8 (IOMUXC_SMUXC_PDISP_D8)

Address: IOMUXC_SMUXC_PDISP_D8 is 53FA_8000h base + 190h offset = 53FA_8190h



IOMUXC_SMUXC_PDISP_D8 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D8. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: DISP_D8. **NOTE:** Pad DISP_D8 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_8_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_ESDHC4_IPP_CMD_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_FEC_FEC_TX_CLK_SELECT_INPUT for mode ALT6. • Config Register IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT for mode ALT5. 000 Select mux mode: ALT0 mux port: DAT[8] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[8] of instance: gpio2. 010 Select mux mode: ALT2 mux port: CLE of instance: rawnand. 011 Select mux mode: ALT3 mux port: LCTL of instance: esdhc1. 100 Select mux mode: ALT4 mux port: CMD of instance: esdhc4.

Table continues on the next page...

IOMUXC_SMUXC_PDISP_D8 field descriptions (continued)

Field	Description
	101 Select mux mode: ALT5 mux port: COL[4] of instance: kpp. 110 Select mux mode: ALT6 mux port: TX_CLK of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[0] of instance: usbphy1.

35.4.102 SW_MUX_CTL_PAD_DISP_D9 (IOMUXC_SMUXC_PDISP_D9)

Address: IOMUXC_SMUXC_PDISP_D9 is 53FA_8000h base + 194h offset = 53FA_8194h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PDISP_D9 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D9. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: DISP_D9. **NOTE:** Pad DISP_D9 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_9_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_ESDHC4_IPP_CARD_CLK_IN_SELECT_INPUT for mode ALT4.

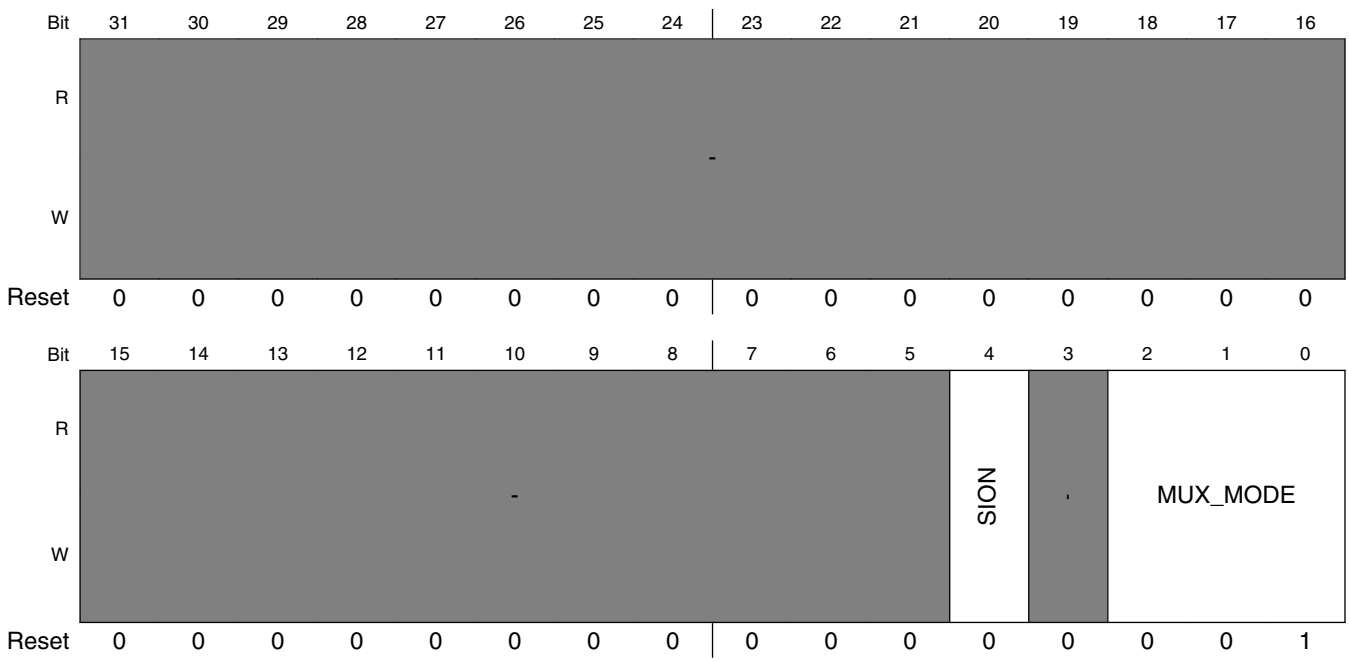
Table continues on the next page...

IOMUXC_SMUXC_PDISP_D9 field descriptions (continued)

Field	Description
	• Config Register IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT for mode ALT6. • Config Register IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT for mode ALT5. 000 Select mux mode: ALT0 mux port: DAT[9] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[9] of instance: gpio2. 010 Select mux mode: ALT2 mux port: ALE of instance: rawnand. 011 Select mux mode: ALT3 mux port: LCTL of instance: esdhc2. 100 Select mux mode: ALT4 mux port: CLK of instance: esdhc4. 101 Select mux mode: ALT5 mux port: ROW[4] of instance: kpp. 110 Select mux mode: ALT6 mux port: RX_ER of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[1] of instance: usbphy1.

35.4.103 SW_MUX_CTL_PAD_DISP_D10 (IOMUXC_SMUXC_PDISP_D10)

Address: IOMUXC_SMUXC_PDISP_D10 is 53FA_8000h base + 198h offset = 53FA_8198h



IOMUXC_SMUXC_PDISP_D10 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D10. 0 Input Path is determined by functionality of the selected mux mode (regular).

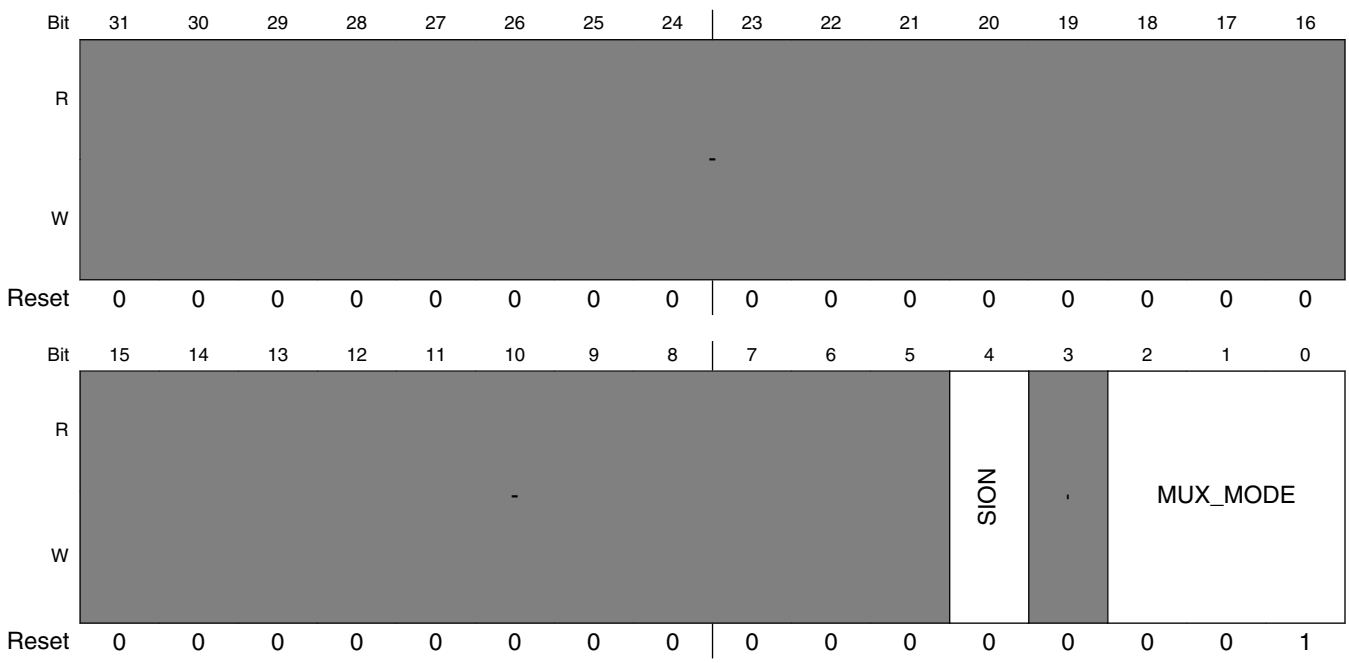
Table continues on the next page...

IOMUXC_SMUXC_PDISP_D10 field descriptions (continued)

Field	Description
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: DISP_D10. **NOTE:** Pad DISP_D10 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_10_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_ESDHC4_IPP_DAT0_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_FEC_FEC_RX_DV_SELECT_INPUT for mode ALT6. • Config Register IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT for mode ALT5. 000 Select mux mode: ALT0 mux port: DAT[10] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[10] of instance: gpio2. 010 Select mux mode: ALT2 mux port: CEN[0] of instance: rawnand. 011 Select mux mode: ALT3 mux port: LCTL of instance: esdhc3. 100 Select mux mode: ALT4 mux port: DAT0 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: COL[5] of instance: kpp. 110 Select mux mode: ALT6 mux port: RX_DV of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[2] of instance: usbphy1.

35.4.104 SW_MUX_CTL_PAD_DISP_D11 (IOMUXC_SMUXC_PDISP_D11)

Address: IOMUXC_SMUXC_PDISP_D11 is 53FA_8000h base + 19Ch offset = 53FA_819Ch

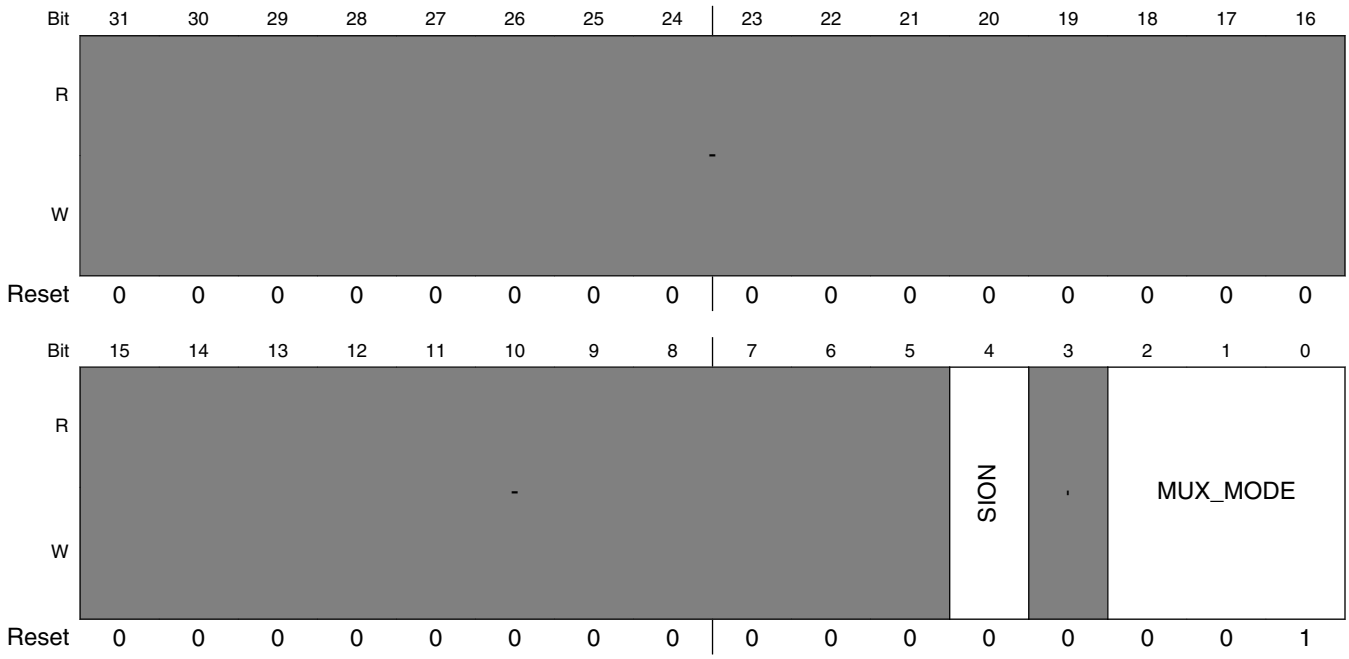


IOMUXC_SMUXC_PDISP_D11 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D11. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: DISP_D11. **NOTE:** Pad DISP_D11 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_11_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_ESDHC4_IPP_DAT1_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_FEC_FEC_RDATA_1_SELECT_INPUT for mode ALT6. • Config Register IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT for mode ALT5. 000 Select mux mode: ALT0 mux port: DAT[11] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[11] of instance: gpio2. 010 Select mux mode: ALT2 mux port: CEN[1] of instance: rawnand. 100 Select mux mode: ALT4 mux port: DAT1 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: ROW[5] of instance: kpp. 110 Select mux mode: ALT6 mux port: RDATA[1] of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[3] of instance: usbphy1.

35.4.105 SW_MUX_CTL_PAD_DISP_D12 (IOMUXC_SMUXC_PDISP_D12)

Address: IOMUXC_SMUXC_PDISP_D12 is 53FA_8000h base + 1A0h offset = 53FA_81A0h



IOMUXC_SMUXC_PDISP_D12 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D12. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: DISP_D12. **NOTE:** Pad DISP_D12 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_12_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_ESDHC4_IPP_DAT2_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_FEC_FEC_RDATA_0_SELECT_INPUT for mode ALT6. • Config Register IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT for mode ALT5. 000 Select mux mode: ALT0 mux port: DAT[12] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[12] of instance: gpio2. 010 Select mux mode: ALT2 mux port: CEN[2] of instance: rawnand. 011 Select mux mode: ALT3 mux port: CD of instance: esdhc1. 100 Select mux mode: ALT4 mux port: DAT2 of instance: esdhc4.

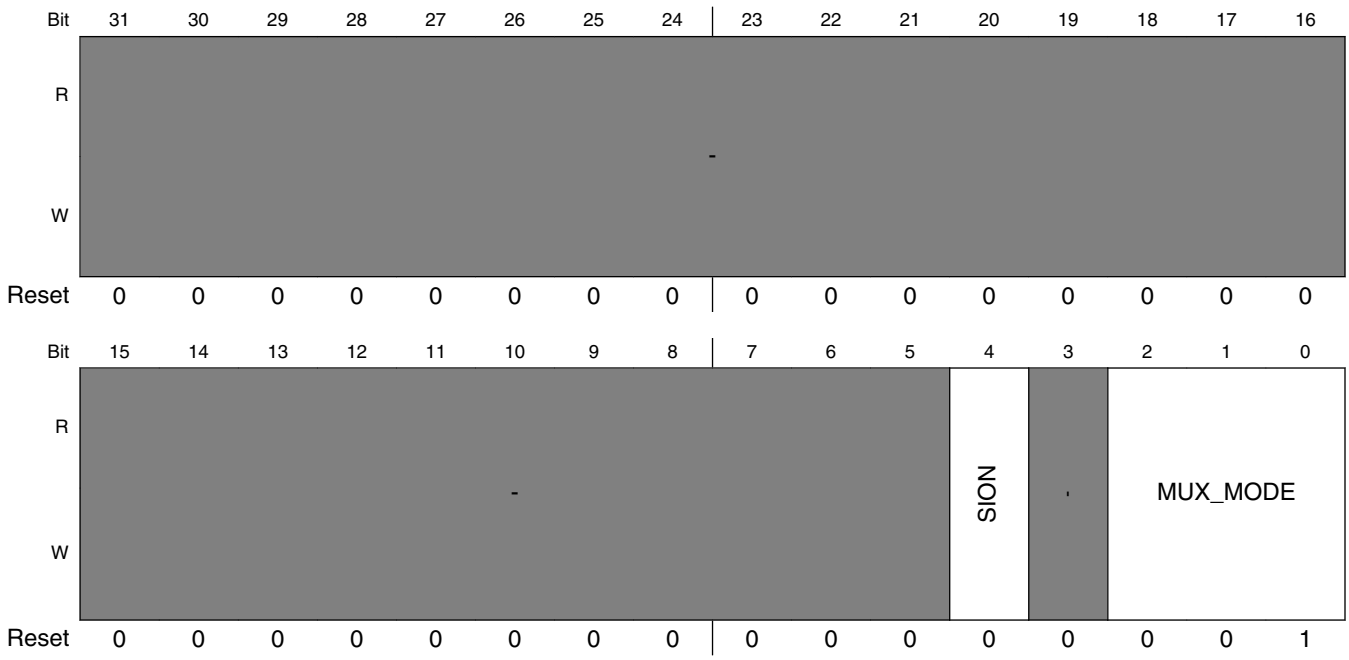
Table continues on the next page...

IOMUXC_SMUXC_PDISP_D12 field descriptions (continued)

Field	Description
	101 Select mux mode: ALT5 mux port: COL[6] of instance: kpp. 110 Select mux mode: ALT6 mux port: RDATA[0] of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[4] of instance: usbphy1.

35.4.106 SW_MUX_CTL_PAD_DISP_D13 (IOMUXC_SMUXC_PDISP_D13)

Address: IOMUXC_SMUXC_PDISP_D13 is 53FA_8000h base + 1A4h offset = 53FA_81A4h



IOMUXC_SMUXC_PDISP_D13 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D13. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: DISP_D13. **NOTE:** Pad DISP_D13 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_13_SELECT_INPUT for mode ALT0.

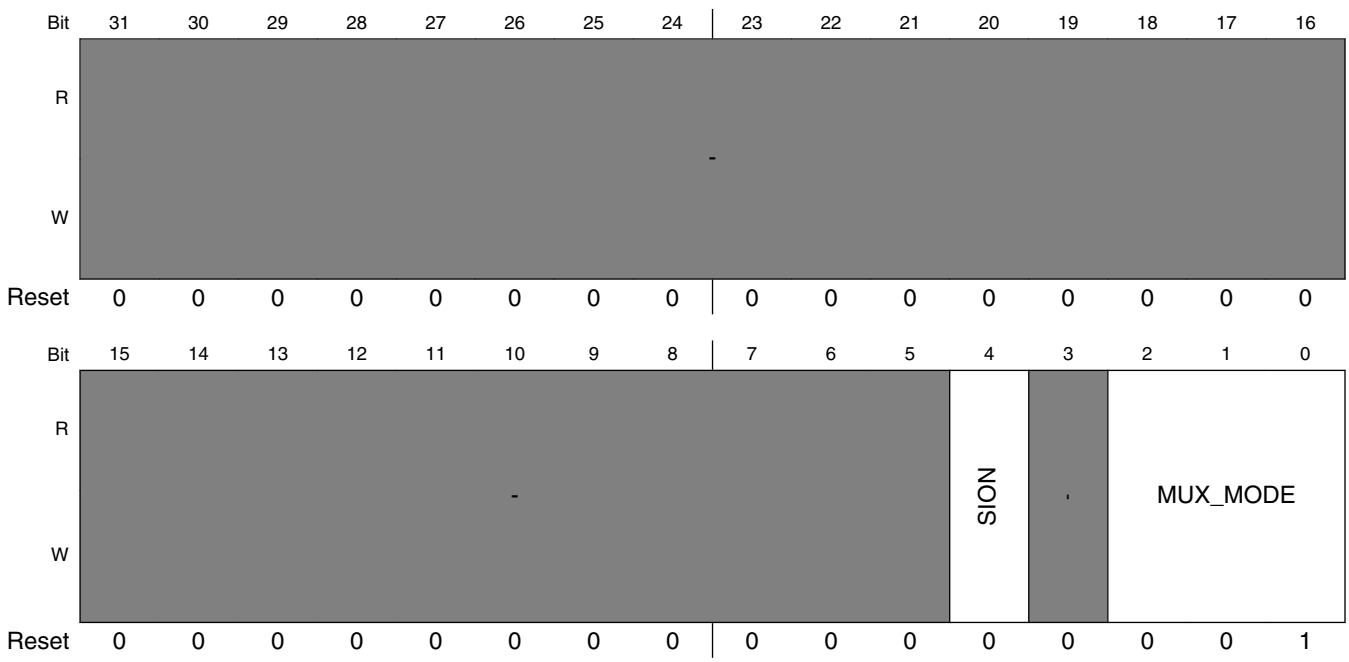
Table continues on the next page...

IOMUXC_SMUXC_PDISP_D13 field descriptions (continued)

Field	Description
	• Config Register IOMUXC_ESDHC4_IPP_DAT3_IN_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT for mode ALT5. 000 Select mux mode: ALT0 mux port: DAT[13] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[13] of instance: gpio2. 010 Select mux mode: ALT2 mux port: CEN[3] of instance: rawnand. 011 Select mux mode: ALT3 mux port: CD of instance: esdhc3. 100 Select mux mode: ALT4 mux port: DAT3 of instance: esdhc4. 101 Select mux mode: ALT5 mux port: ROW[6] of instance: kpp. 110 Select mux mode: ALT6 mux port: TX_EN of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[5] of instance: usbphy1.

35.4.107 SW_MUX_CTL_PAD_DISP_D14 (IOMUXC_SMUXC_PDISP_D14)

Address: IOMUXC_SMUXC_PDISP_D14 is 53FA_8000h base + 1A8h offset = 53FA_81A8h



IOMUXC_SMUXC_PDISP_D14 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D14. 0 Input Path is determined by functionality of the selected mux mode (regular).

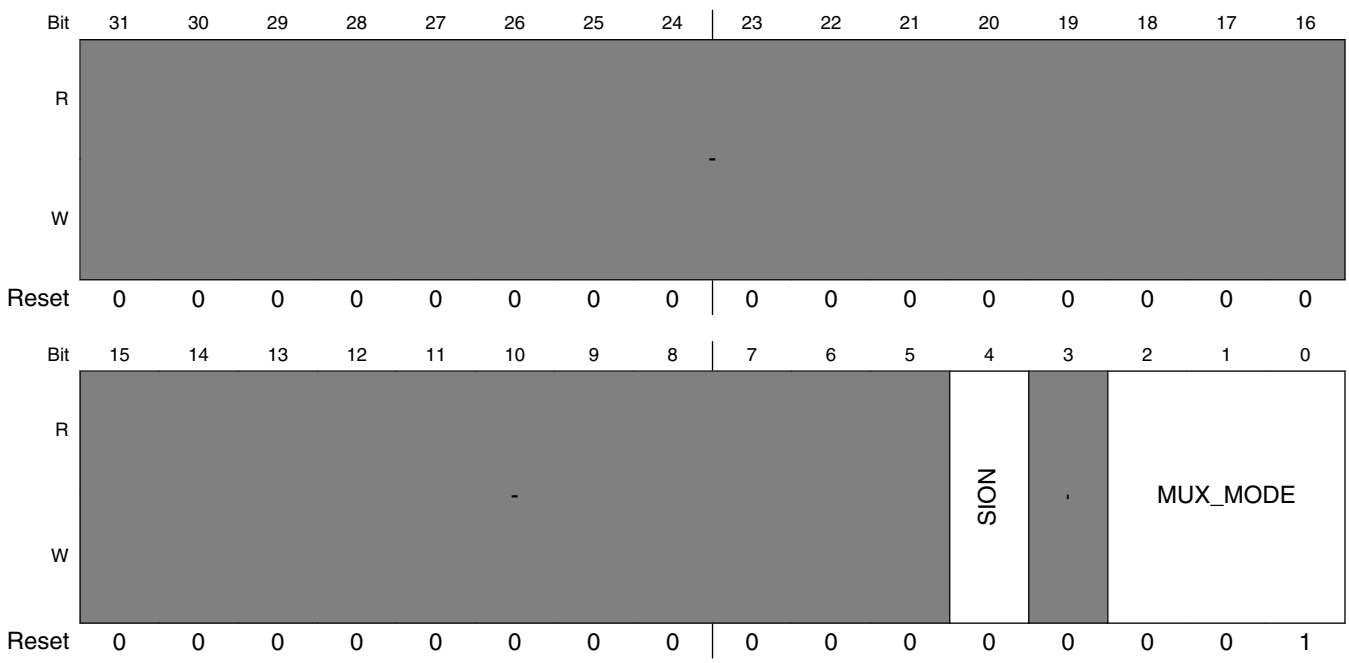
Table continues on the next page...

IOMUXC_SMUXC_PDISP_D14 field descriptions (continued)

Field	Description
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: DISP_D14. **NOTE:** Pad DISP_D14 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_14_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: DAT[14] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[14] of instance: gpio2. 010 Select mux mode: ALT2 mux port: READY0 of instance: rawnand. 011 Select mux mode: ALT3 mux port: WP of instance: esdhc1. 100 Select mux mode: ALT4 mux port: WP of instance: esdhc4. 101 Select mux mode: ALT5 mux port: COL[7] of instance: kpp. 110 Select mux mode: ALT6 mux port: TDATA[1] of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[6] of instance: usbphy1.

35.4.108 SW_MUX_CTL_PAD_DISP_D15 (IOMUXC_SMUXC_PDISP_D15)

Address: IOMUXC_SMUXC_PDISP_D15 is 53FA_8000h base + 1ACh offset = 53FA_81ACh



IOMUXC_SMUXC_PDISP_D15 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad DISP_D15. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: DISP_D15. **NOTE:** Pad DISP_D15 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_15_SELECT_INPUT for mode ALT0. • Config Register IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT for mode ALT5. • Config Register IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: DAT[15] of instance: elcdif. 001 Select mux mode: ALT1 mux port: GPIO[15] of instance: gpio2. 010 Select mux mode: ALT2 mux port: DQS of instance: rawnand. 011 Select mux mode: ALT3 mux port: RST of instance: esdhc3. 100 Select mux mode: ALT4 mux port: CD of instance: esdhc4. 101 Select mux mode: ALT5 mux port: ROW[7] of instance: kpp. 110 Select mux mode: ALT6 mux port: TDATA[0] of instance: fec. 111 Select mux mode: ALT7 mux port: DATAOUT[7] of instance: usbphy1.

35.4.109 SW_MUX_CTL_PAD_EPDC_D0 (IOMUXC_SMUXC_PEPDC_D0)

Address: IOMUXC_SMUXC_PEPDC_D0 is 53FA_8000h base + 1B0h offset = 53FA_81B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

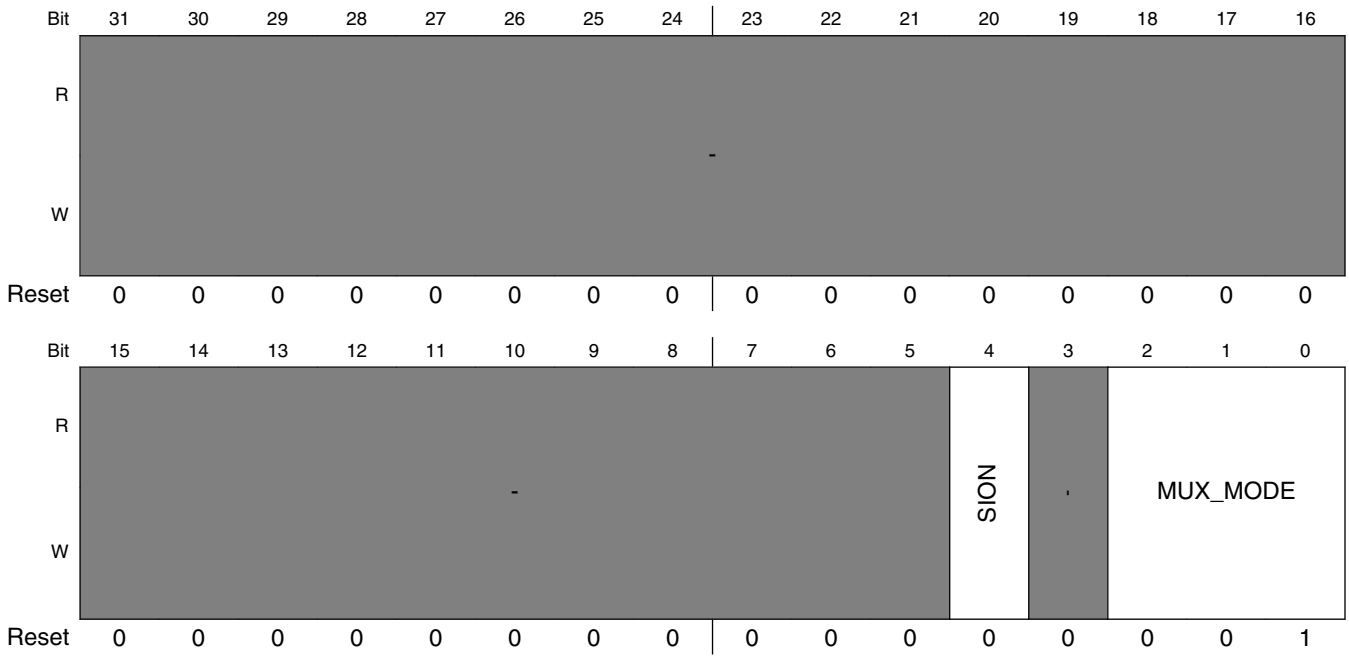
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_D0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_D0. **NOTE:** Pad EPDC_D0 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_0_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[0] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[0] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[0] of instance: weimv2. 011 Select mux mode: ALT3 mux port: RS of instance: elcdif. 100 Select mux mode: ALT4 mux port: DOTCLK of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVT_CHN_LINES[0] of instance: sdma. 111 Select mux mode: ALT7 mux port: DATAOUT[0] of instance: usbphy2.

35.4.110 SW_MUX_CTL_PAD_EPDC_D1 (IOMUXC_SMUXC_PEPDC_D1)

Address: IOMUXC_SMUXC_PEPDC_D1 is 53FA_8000h base + 1B4h offset = 53FA_81B4h



IOMUXC_SMUXC_PEPDC_D1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_D1. **NOTE:** Pad EPDC_D1 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_1_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[1] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[1] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[1] of instance: weimv2. 011 Select mux mode: ALT3 mux port: CS of instance: elcdif. 100 Select mux mode: ALT4 mux port: ENABLE of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVT_CHN_LINES[1] of instance: sdma. 111 Select mux mode: ALT7 mux port: DATAOUT[1] of instance: usbphy2.

35.4.111 SW_MUX_CTL_PAD_EPDC_D2 (IOMUXC_SMUXC_PEPDC_D2)

Address: IOMUXC_SMUXC_PEPDC_D2 is 53FA_8000h base + 1B8h offset = 53FA_81B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

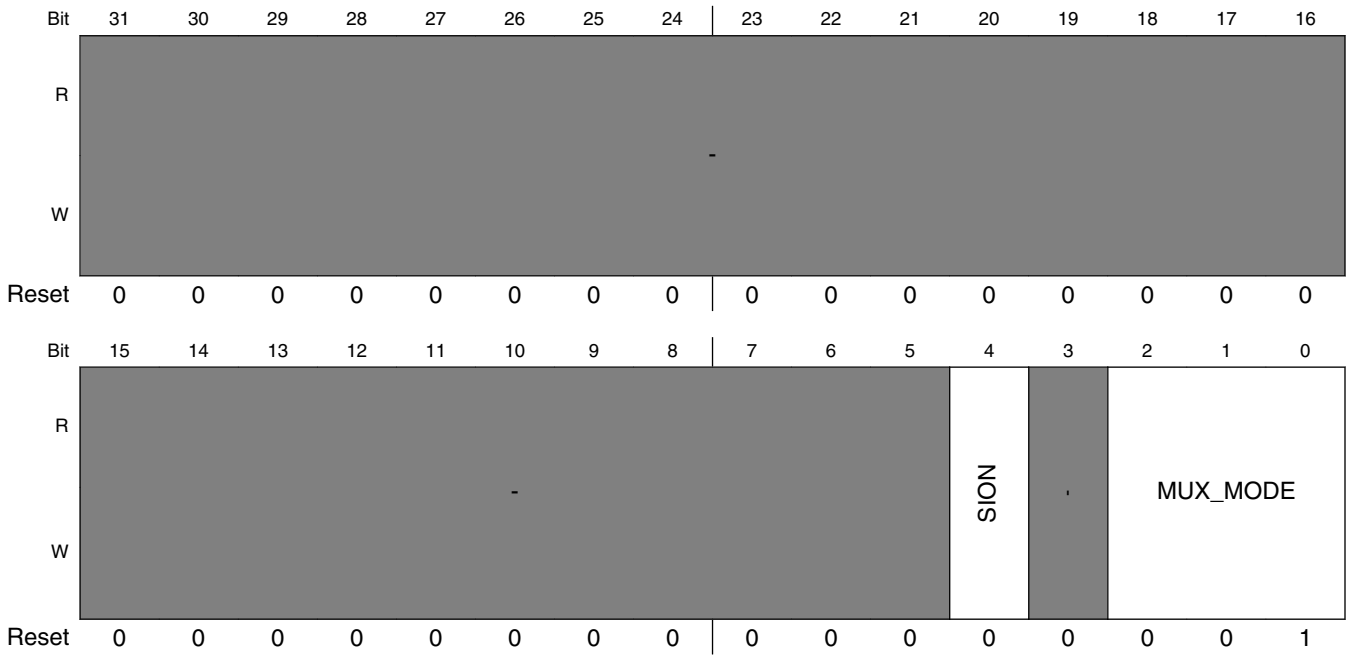
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_D2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_D2. **NOTE:** Pad EPDC_D2 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_VSYNC_I_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_2_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[2] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[2] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[2] of instance: weimv2. 011 Select mux mode: ALT3 mux port: WR_RWN of instance: elcdif. 100 Select mux mode: ALT4 mux port: VSYNC of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVT_CHN_LINES[2] of instance: sdma. 111 Select mux mode: ALT7 mux port: DATAOUT[2] of instance: usbphy2.

35.4.112 SW_MUX_CTL_PAD_EPDC_D3 (IOMUXC_SMUXC_PEPDC_D3)

Address: IOMUXC_SMUXC_PEPDC_D3 is 53FA_8000h base + 1BCh offset = 53FA_81BCh



IOMUXC_SMUXC_PEPDC_D3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_D3. **NOTE:** Pad EPDC_D3 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_BUSY_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_3_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[3] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[3] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[3] of instance: weimv2. 011 Select mux mode: ALT3 mux port: RD_E of instance: elcdif. 100 Select mux mode: ALT4 mux port: HSYNC of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVT_CHN_LINES[3] of instance: sdma. 111 Select mux mode: ALT7 mux port: DATAOUT[3] of instance: usbphy2.

35.4.113 SW_MUX_CTL_PAD_EPDC_D4 (IOMUXC_SMUXC_PEPDC_D4)

Address: IOMUXC_SMUXC_PEPDC_D4 is 53FA_8000h base + 1C0h offset = 53FA_81C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

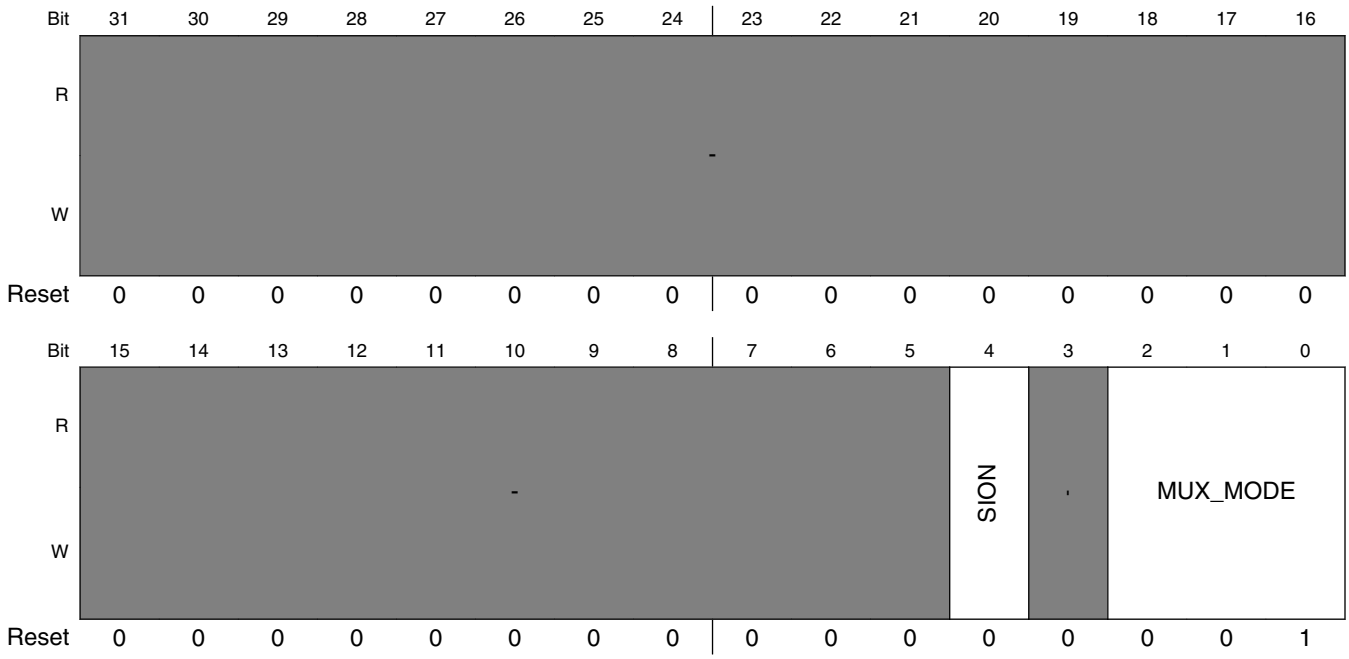
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_D4 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D4. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EPDC_D4. **NOTE:** Pad EPDC_D4 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_4_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[4] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[4] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[4] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_EVT_CHN_LINES[4] of instance: sdma. 111 Select mux mode: ALT7 mux port: DATAOUT[4] of instance: usbphy2.

35.4.114 SW_MUX_CTL_PAD_EPDC_D5 (IOMUXC_SMUXC_PEPDC_D5)

Address: IOMUXC_SMUXC_PEPDC_D5 is 53FA_8000h base + 1C4h offset = 53FA_81C4h



IOMUXC_SMUXC_PEPDC_D5 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D5. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EPDC_D5. **NOTE:** Pad EPDC_D5 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_5_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[5] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[5] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[5] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_EVT_CHN_LINES[5] of instance: sdma. 111 Select mux mode: ALT7 mux port: DATAOUT[5] of instance: usbphy2.

35.4.115 SW_MUX_CTL_PAD_EPDC_D6 (IOMUXC_SMUXC_PEPDC_D6)

Address: IOMUXC_SMUXC_PEPDC_D6 is 53FA_8000h base + 1C8h offset = 53FA_81C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

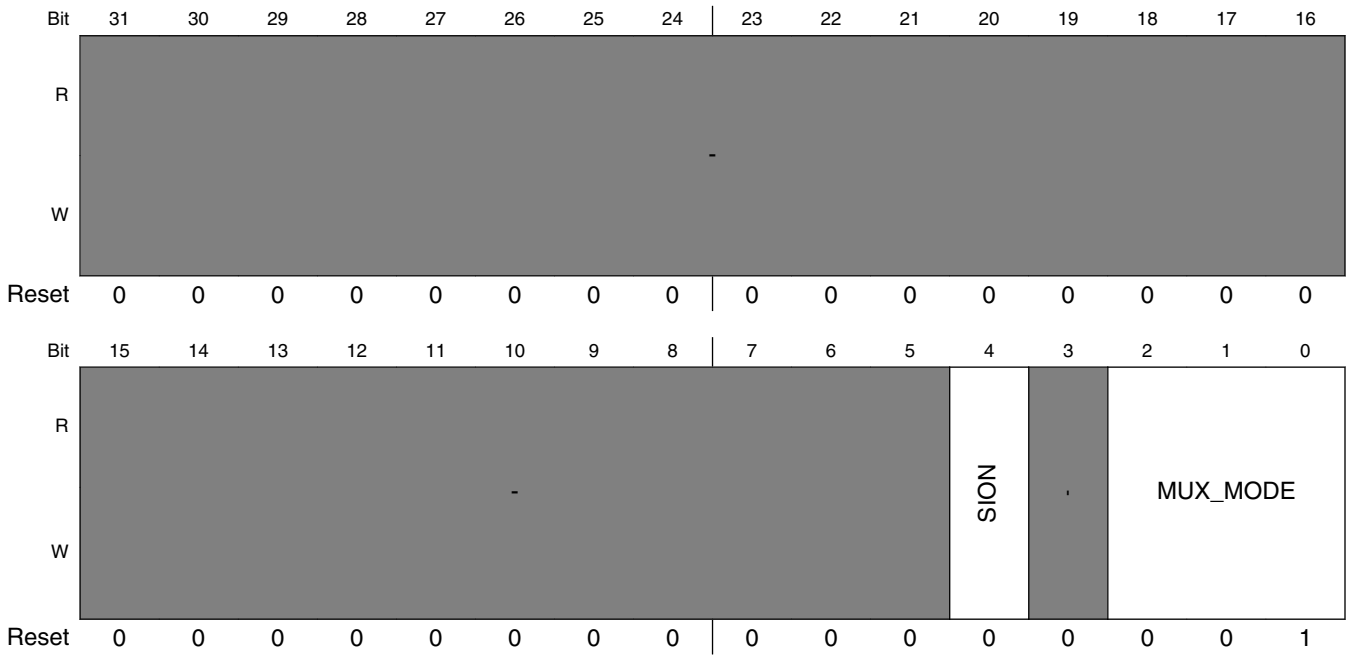
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_D6 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D6. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EPDC_D6. **NOTE:** Pad EPDC_D6 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_6_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[6] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[6] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[6] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_EVT_CHN_LINES[6] of instance: sdma. 111 Select mux mode: ALT7 mux port: DATAOUT[6] of instance: usbphy2.

35.4.116 SW_MUX_CTL_PAD_EPDC_D7 (IOMUXC_SMUXC_PEPDC_D7)

Address: IOMUXC_SMUXC_PEPDC_D7 is 53FA_8000h base + 1CCh offset = 53FA_81CCh

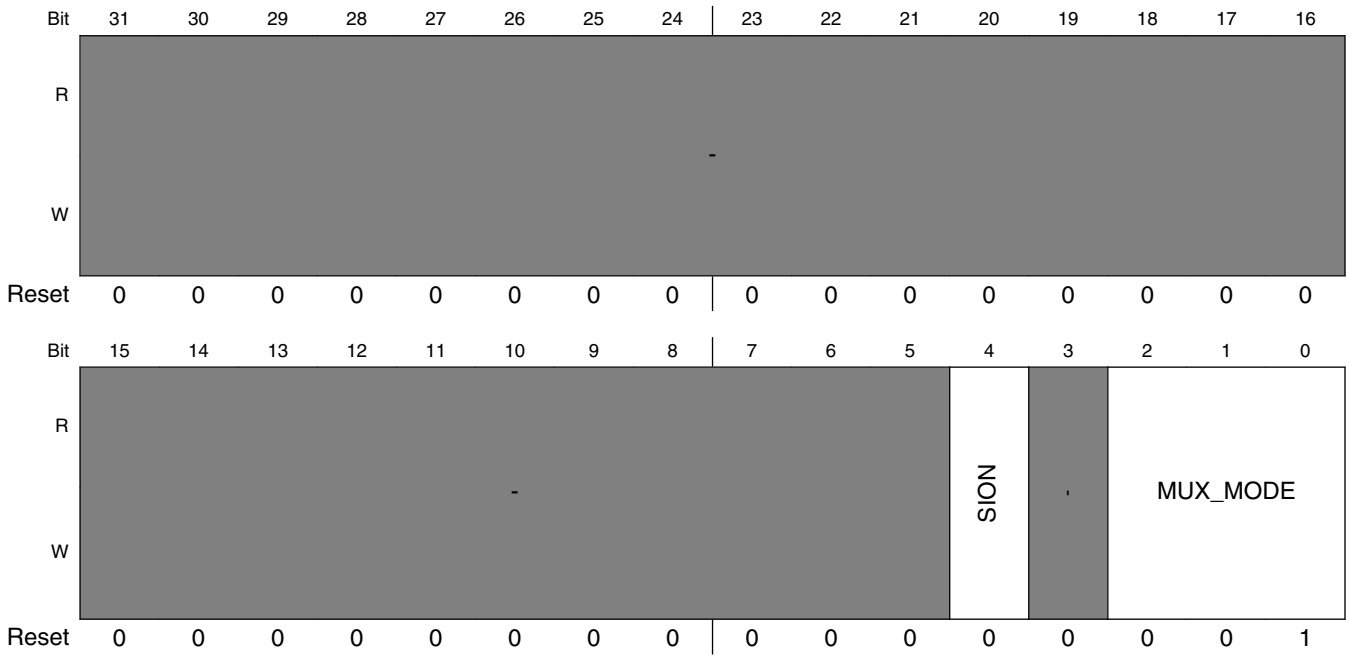


IOMUXC_SMUXC_PEPDC_D7 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D7. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EPDC_D7. **NOTE:** Pad EPDC_D7 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_7_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[7] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[7] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[7] of instance: weimv2. 110 Select mux mode: ALT6 mux port: DEBUG_EVT_CHN_LINES[7] of instance: sdma. 111 Select mux mode: ALT7 mux port: DATAOUT[7] of instance: usbphy2.

35.4.117 SW_MUX_CTL_PAD_EPDC_D8 (IOMUXC_SMUXC_PEPDC_D8)

Address: IOMUXC_SMUXC_PEPDC_D8 is 53FA_8000h base + 1D0h offset = 53FA_81D0h

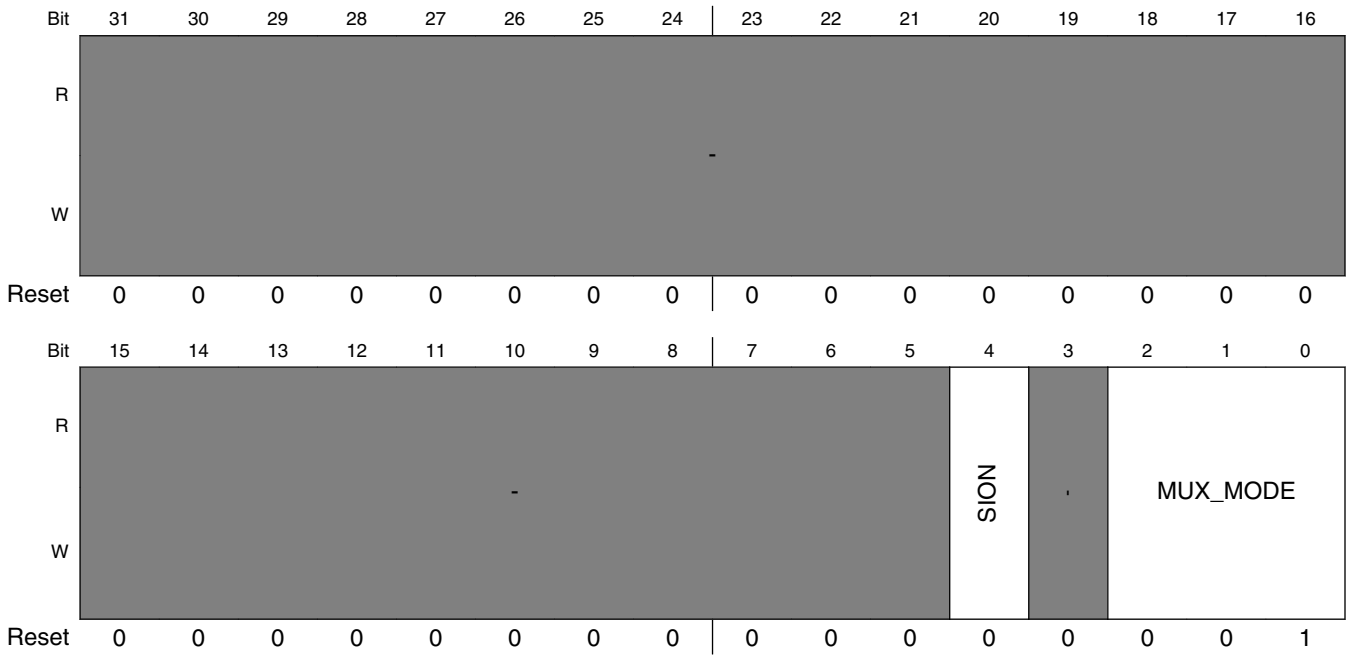


IOMUXC_SMUXC_PEPDC_D8 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D8. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EPDC_D8. **NOTE:** Pad EPDC_D8 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_8_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[8] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[8] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[8] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[24] of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_MATCHED_DMBUS of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[0] of instance: usbphy2.

35.4.118 SW_MUX_CTL_PAD_EPDC_D9 (IOMUXC_SMUXC_PEPDC_D9)

Address: IOMUXC_SMUXC_PEPDC_D9 is 53FA_8000h base + 1D4h offset = 53FA_81D4h



IOMUXC_SMUXC_PEPDC_D9 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D9. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EPDC_D9. **NOTE:** Pad EPDC_D9 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_9_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[9] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[9] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[9] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[25] of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVENT_CHANNEL_SEL of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[1] of instance: usbphy2.

35.4.119 SW_MUX_CTL_PAD_EPDC_D10 (IOMUXC_SMUXC_PEPDC_D10)

Address: IOMUXC_SMUXC_PEPDC_D10 is 53FA_8000h base + 1D8h offset = 53FA_81D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

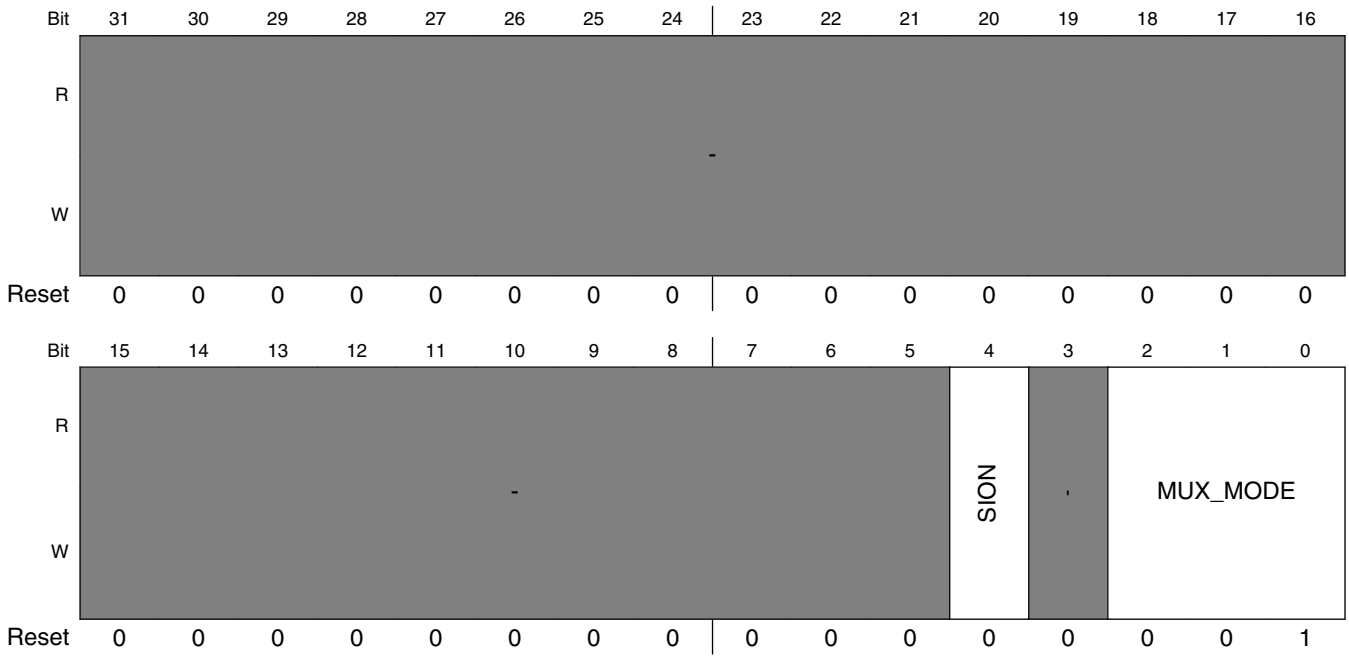
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_D10 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D10. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EPDC_D10. **NOTE:** Pad EPDC_D10 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_10_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[10] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[10] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[10] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[26] of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVENT_CHANNEL[0] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[2] of instance: usbphy2.

35.4.120 SW_MUX_CTL_PAD_EPDC_D11 (IOMUXC_SMUXC_PEPDC_D11)

Address: IOMUXC_SMUXC_PEPDC_D11 is 53FA_8000h base + 1DCh offset = 53FA_81DCh



IOMUXC_SMUXC_PEPDC_D11 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D11. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EPDC_D11. **NOTE:** Pad EPDC_D11 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_11_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[11] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[11] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[11] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[27] of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVENT_CHANNEL[1] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[3] of instance: usbphy2.

35.4.121 SW_MUX_CTL_PAD_EPDC_D12 (IOMUXC_SMUXC_PEPDC_D12)

Address: IOMUXC_SMUXC_PEPDC_D12 is 53FA_8000h base + 1E0h offset = 53FA_81E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

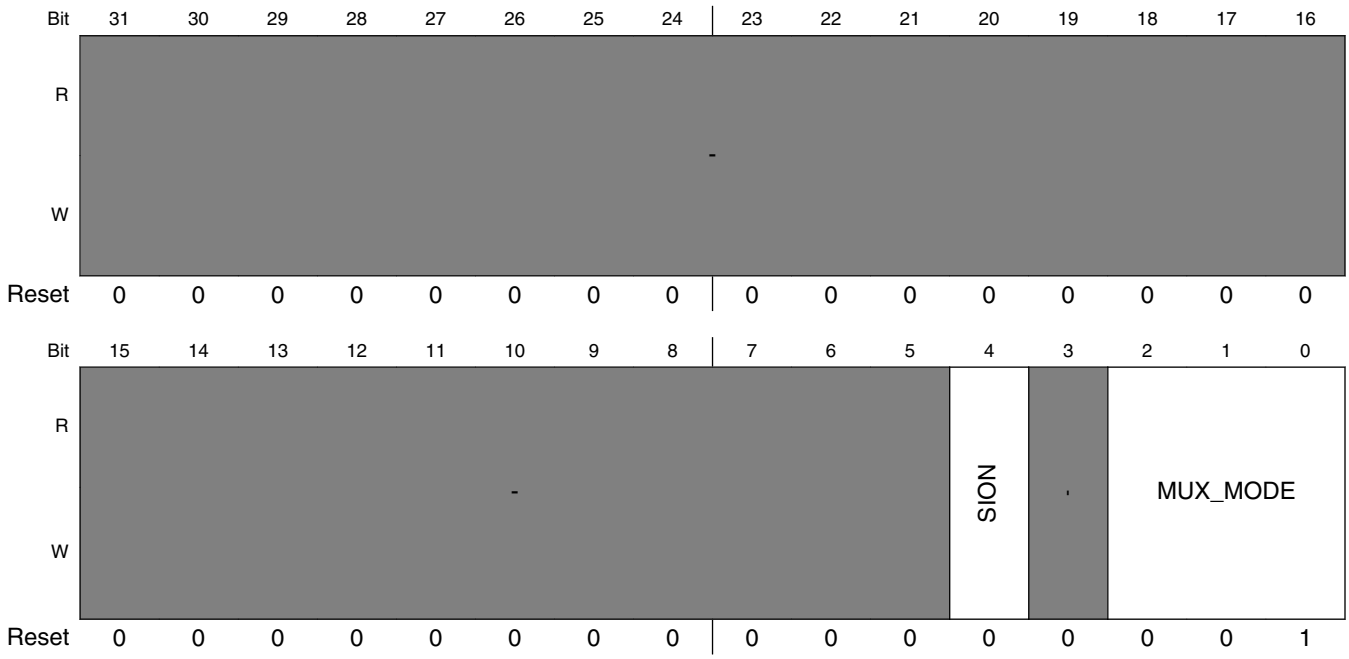
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_D12 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D12. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EPDC_D12. **NOTE:** Pad EPDC_D12 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_12_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[12] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[12] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[12] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[28] of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVENT_CHANNEL[2] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[4] of instance: usbphy2.

35.4.122 SW_MUX_CTL_PAD_EPDC_D13 (IOMUXC_SMUXC_PEPDC_D13)

Address: IOMUXC_SMUXC_PEPDC_D13 is 53FA_8000h base + 1E4h offset = 53FA_81E4h

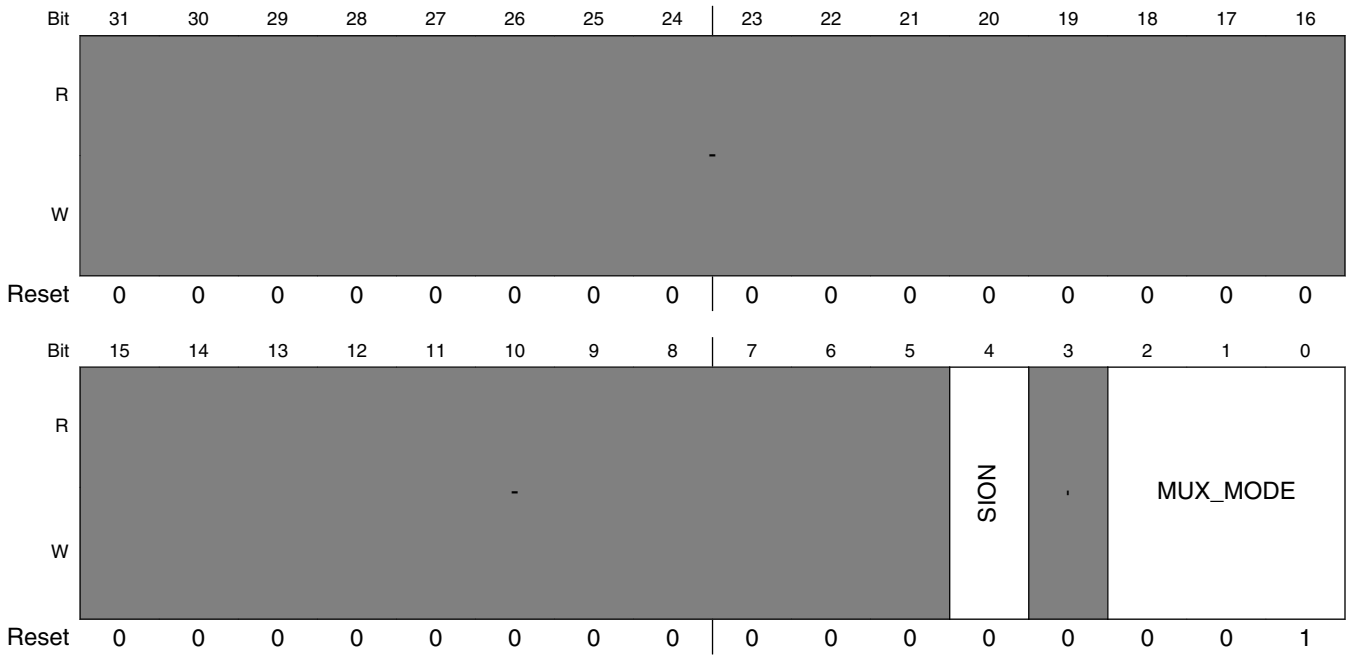


IOMUXC_SMUXC_PEPDC_D13 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D13. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EPDC_D13. **NOTE:** Pad EPDC_D13 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_13_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[13] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[13] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[13] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[29] of instance: elcdif. 110 Select mux mode: ALT6 mux port: DEBUG_EVENT_CHANNEL[3] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[5] of instance: usbphy2.

35.4.123 SW_MUX_CTL_PAD_EPDC_D14 (IOMUXC_SMUXC_PEPDC_D14)

Address: IOMUXC_SMUXC_PEPDC_D14 is 53FA_8000h base + 1E8h offset = 53FA_81E8h

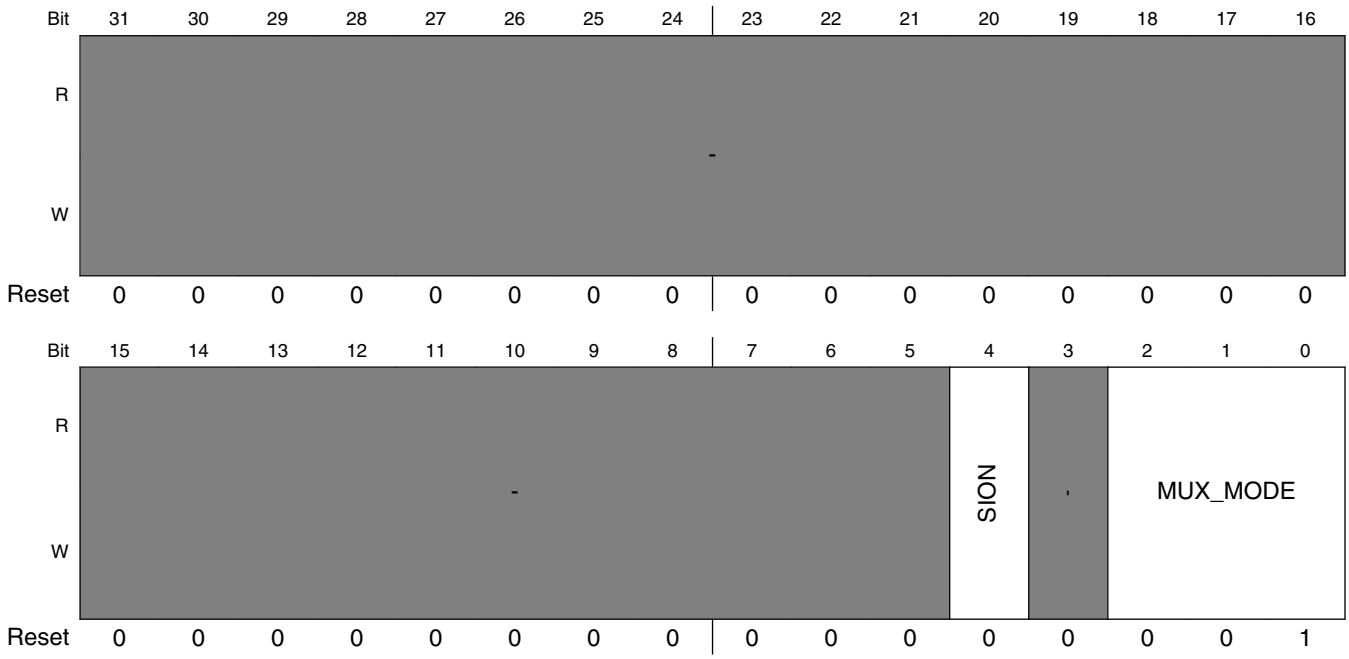


IOMUXC_SMUXC_PEPDC_D14 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D14. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_D14. **NOTE:** Pad EPDC_D14 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_14_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[14] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[14] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[14] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[30] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD6_TXD of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_EVENT_CHANNEL[4] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[6] of instance: usbphy2.

35.4.124 SW_MUX_CTL_PAD_EPDC_D15 (IOMUXC_SMUXC_PEPDC_D15)

Address: IOMUXC_SMUXC_PEPDC_D15 is 53FA_8000h base + 1ECh offset = 53FA_81ECh

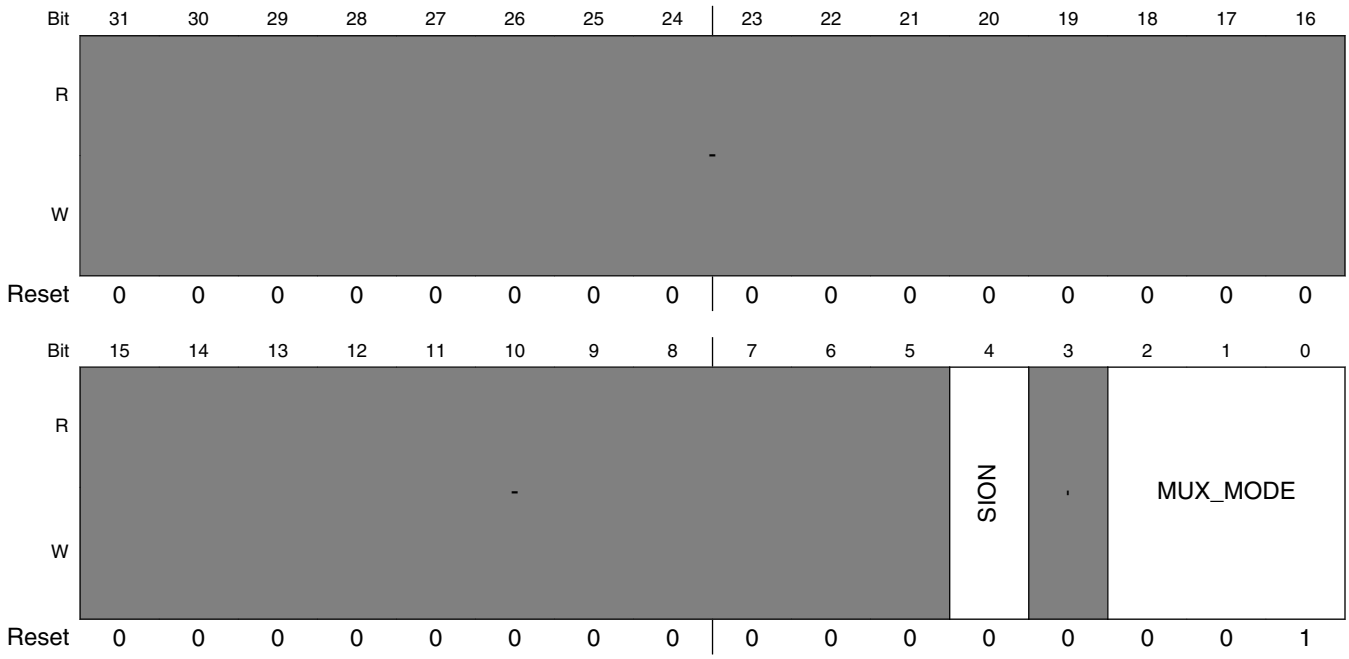


IOMUXC_SMUXC_PEPDC_D15 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_D15. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_D15. **NOTE:** Pad EPDC_D15 is involved in Daisy Chain. • Config Register IOMUXC_WEIMV2_IPP_IND_READ_DATA_15_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: SDDO[15] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[15] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[15] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[31] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD6_TXC of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_EVENT_CHANNEL[5] of instance: sdma. 111 Select mux mode: ALT7 mux port: VSTATUS[7] of instance: usbphy2.

35.4.125 SW_MUX_CTL_PAD_EPDC_GDCLK (IOMUXC_SMUXC_PEPDC_GDCLK)

Address: IOMUXC_SMUXC_PEPDC_GDCLK is 53FA_8000h base + 1F0h offset = 53FA_81F0h

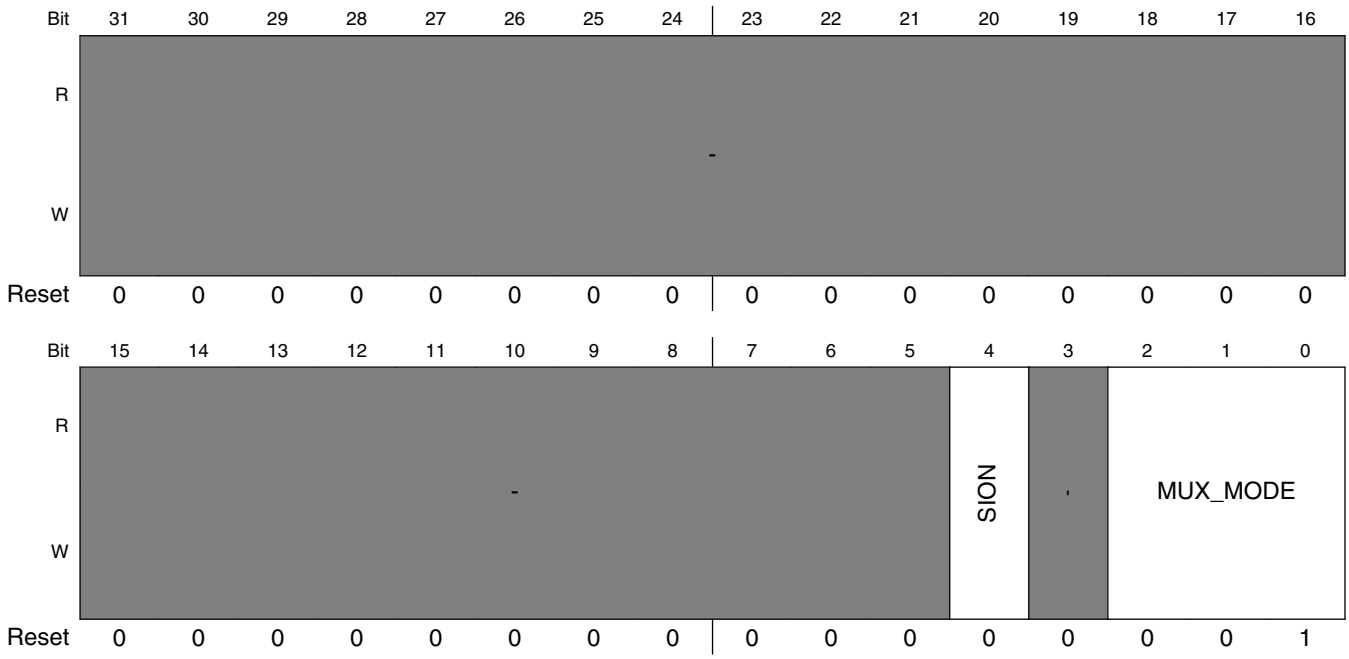


IOMUXC_SMUXC_PEPDC_GDCLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_GDCLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_GDCLK. 000 Select mux mode: ALT0 mux port: GDCLK of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[16] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[16] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[16] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD6_TXFS of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_CORE_STATE[0] of instance: sdma. 111 Select mux mode: ALT7 mux port: BISTOK of instance: usbphy2.

35.4.126 SW_MUX_CTL_PAD_EPDC_GDSP (IOMUXC_SMUXC_PEPDC_GDSP)

Address: IOMUXC_SMUXC_PEPDC_GDSP is 53FA_8000h base + 1F4h offset = 53FA_81F4h

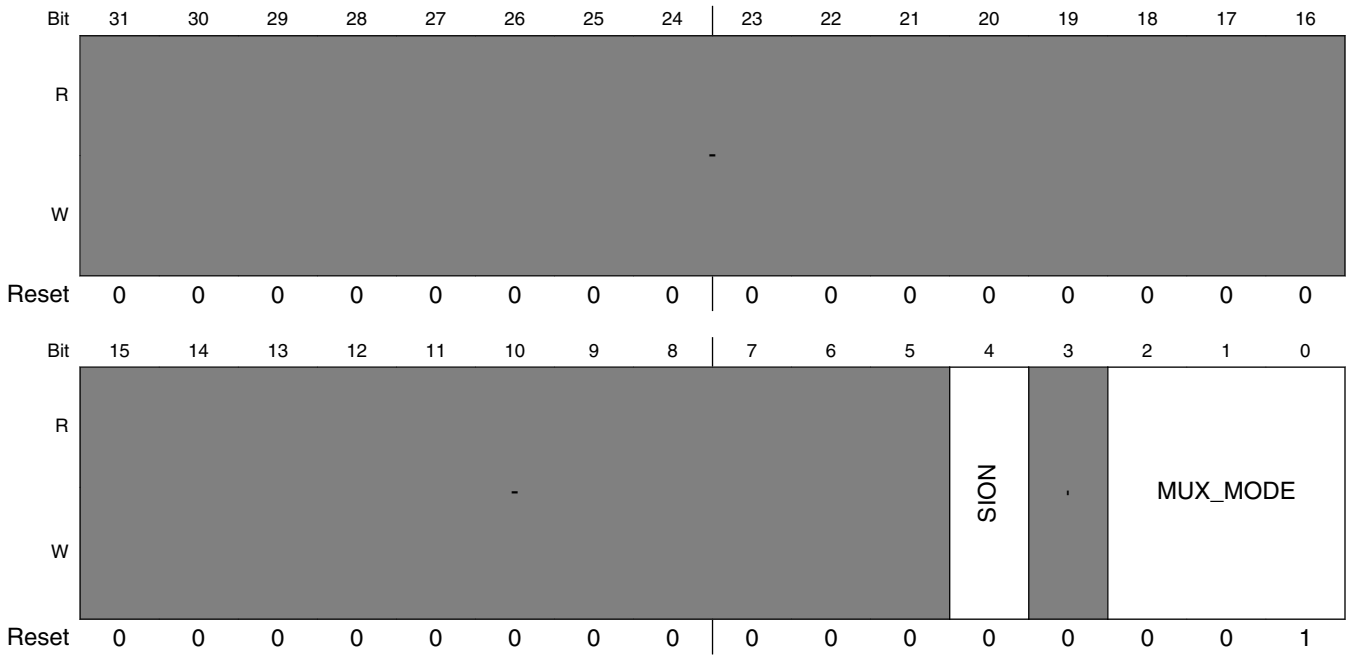


IOMUXC_SMUXC_PEPDC_GDSP field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_GDSP. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_GDSP. 000 Select mux mode: ALT0 mux port: GDSP of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[17] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[17] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[17] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD6_RXD of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_CORE_STATE[1] of instance: sdma. 111 Select mux mode: ALT7 mux port: BVALID of instance: usbphy2.

35.4.127 SW_MUX_CTL_PAD_EPDC_GDOE (IOMUXC_SMUXC_PEPDC_GDOE)

Address: IOMUXC_SMUXC_PEPDC_GDOE is 53FA_8000h base + 1F8h offset = 53FA_81F8h

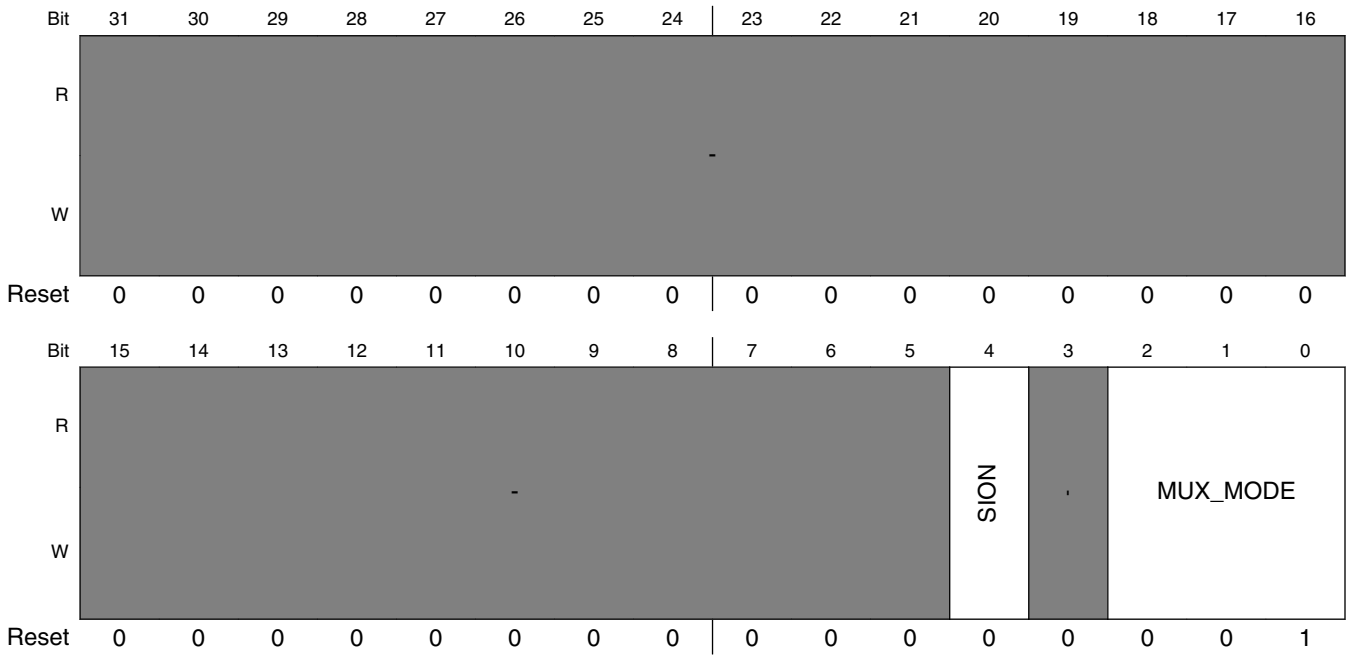


IOMUXC_SMUXC_PEPDC_GDOE field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_GDOE. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_GDOE. 000 Select mux mode: ALT0 mux port: GDOE of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[18] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[18] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[18] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD6_RXC of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_CORE_STATE[2] of instance: sdma. 111 Select mux mode: ALT7 mux port: ENDSESSION of instance: usbphy2.

35.4.128 SW_MUX_CTL_PAD_EPDC_GDRL (IOMUXC_SMUXC_PEPDC_GDRL)

Address: IOMUXC_SMUXC_PEPDC_GDRL is 53FA_8000h base + 1FCh offset = 53FA_81FCh

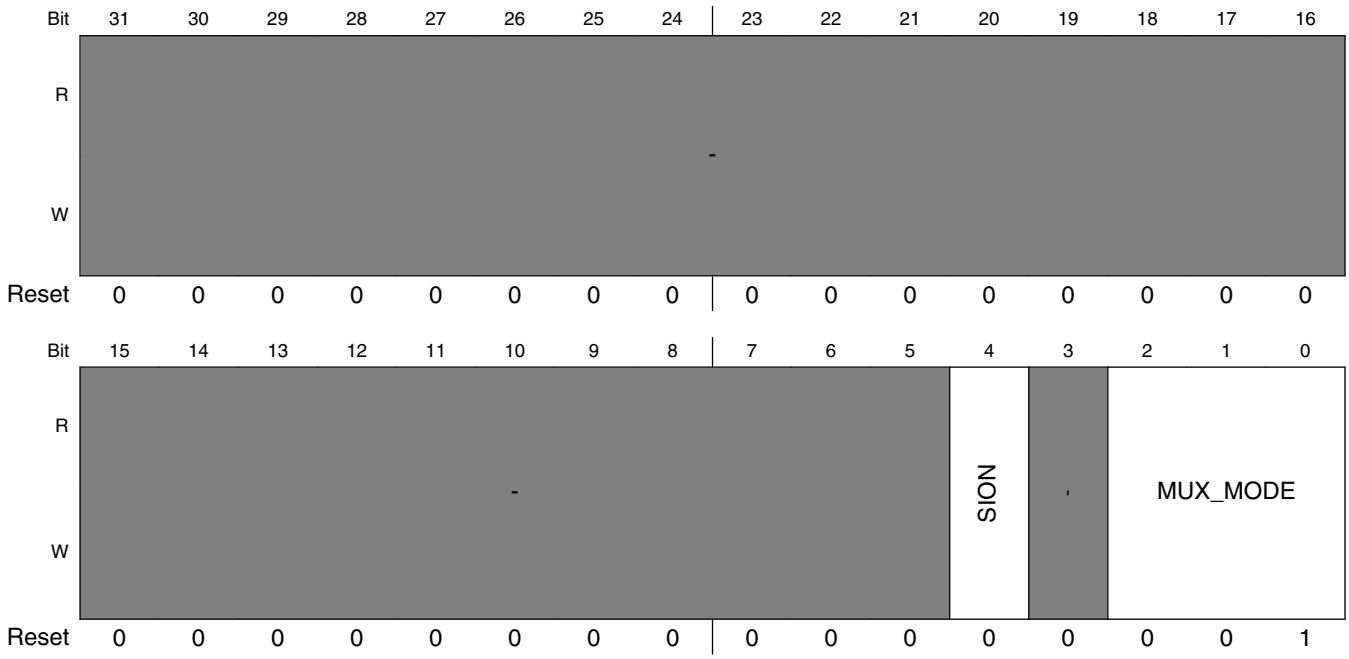


IOMUXC_SMUXC_PEPDC_GDRL field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_GDRL. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_GDRL. 000 Select mux mode: ALT0 mux port: GDRL of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[19] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[19] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[19] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD6_RXFS of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_CORE_STATE[3] of instance: sdma. 111 Select mux mode: ALT7 mux port: IDDIG of instance: usbphy2.

35.4.129 SW_MUX_CTL_PAD_EPDC_SDCLK (IOMUXC_SMUXC_PEPDC_SDCLK)

Address: IOMUXC_SMUXC_PEPDC_SDCLK is 53FA_8000h base + 200h offset = 53FA_8200h

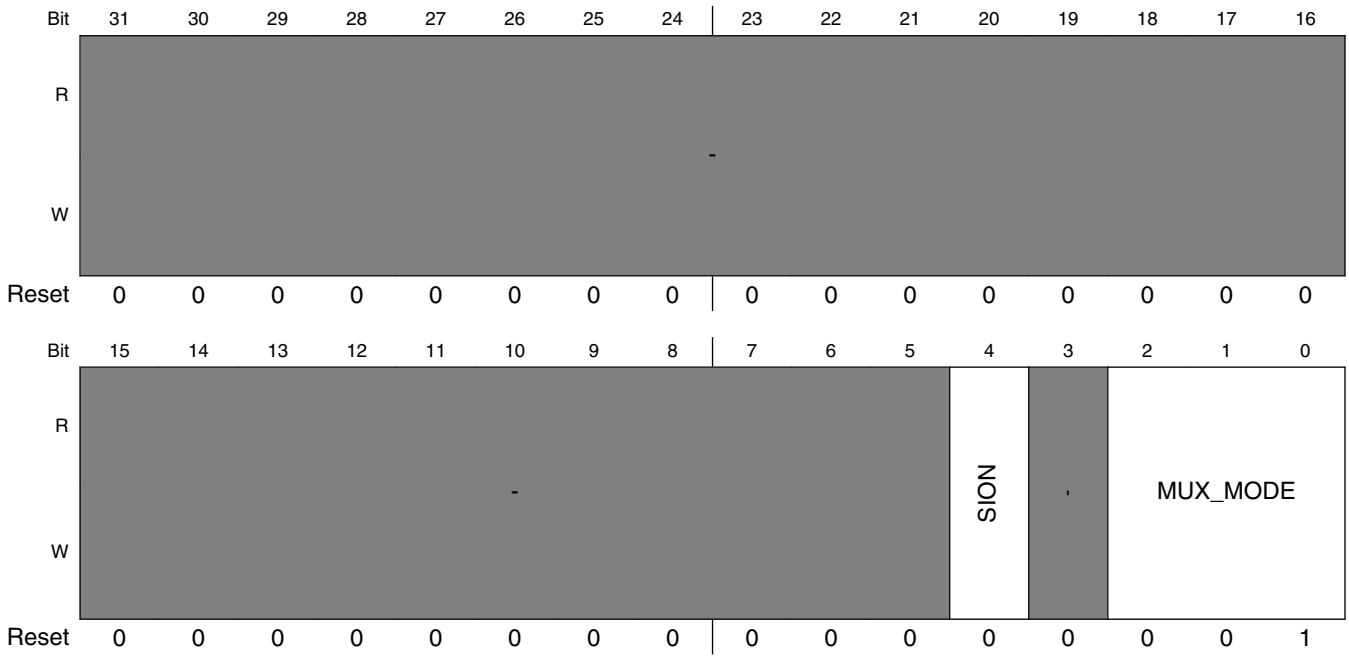


IOMUXC_SMUXC_PEPDC_SDCLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDCLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_SDCLK. 000 Select mux mode: ALT0 mux port: SDCLK of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[20] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[20] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[20] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD5_TXD of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_BUS_DEVICE[0] of instance: sdma. 111 Select mux mode: ALT7 mux port: HOSTDISCONNECT of instance: usbphy2.

35.4.130 SW_MUX_CTL_PAD_EPDC_SDOEZ (IOMUXC_SMUXC_PEPDC_SDOEZ)

Address: IOMUXC_SMUXC_PEPDC_SDOEZ is 53FA_8000h base + 204h offset = 53FA_8204h



IOMUXC_SMUXC_PEPDC_SDOEZ field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDOEZ. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_SDOEZ. 000 Select mux mode: ALT0 mux port: SDOEZ of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[21] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[21] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[21] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD5_TXC of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_BUS_DEVICE[1] of instance: sdma. 111 Select mux mode: ALT7 mux port: TXREADY of instance: usbphy2.

35.4.131 SW_MUX_CTL_PAD_EPDC_SDOED (IOMUXC_SMUXC_PEPDC_SDOED)

Address: IOMUXC_SMUXC_PEPDC_SDOED is 53FA_8000h base + 208h offset = 53FA_8208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

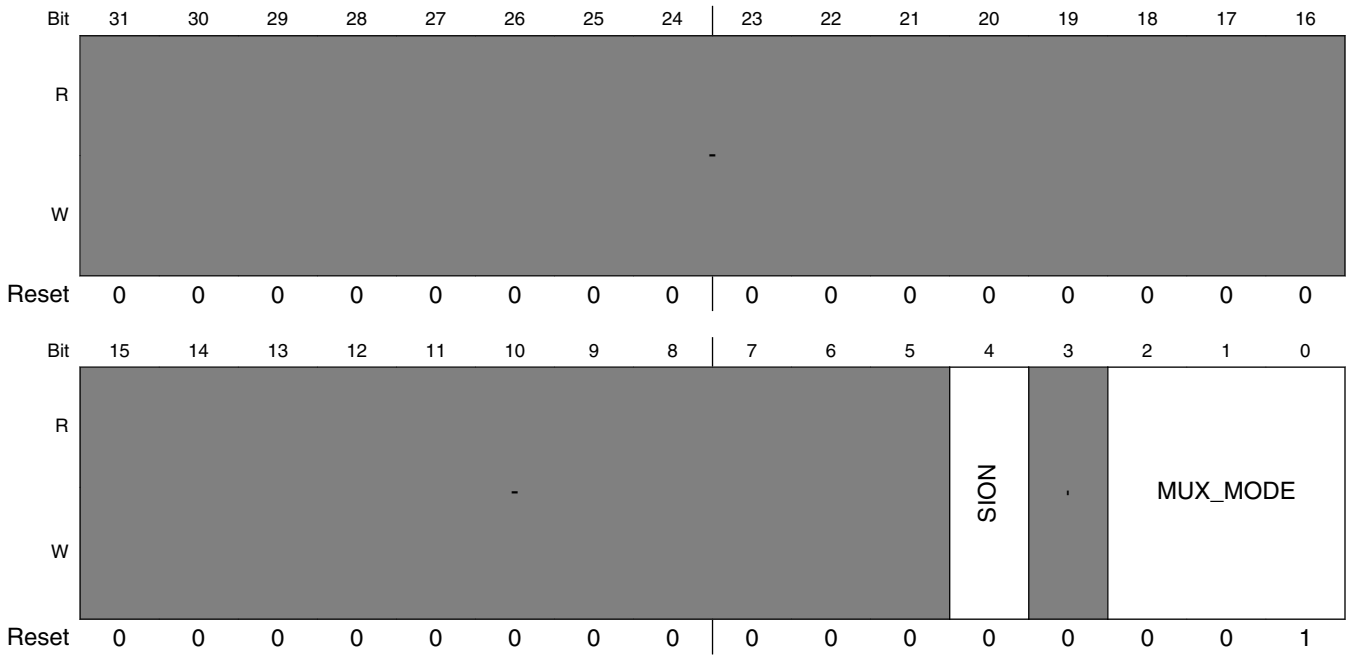
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_SDOED field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDOED. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_SDOED. 000 Select mux mode: ALT0 mux port: SDOED of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[22] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[22] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[22] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD5_TXFS of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_BUS_DEVICE[2] of instance: sdma. 111 Select mux mode: ALT7 mux port: RXVALID of instance: usbphy2.

35.4.132 OMUXC_SW_MUX_CTL_PAD_EPDC_SDOE (IOMUXC_OMUXC_SMUXC_PEPDC_SDOE)

Address: IOMUXC_OMUXC_SMUXC_PEPDC_SDOE is 53FA_8000h base + 20Ch offset = 53FA_820Ch



IOMUXC_OMUXC_SMUXC_PEPDC_SDOE field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDOE. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_SDOE. 000 Select mux mode: ALT0 mux port: SDOE of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[23] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[23] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[23] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD5_RXD of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_BUS_DEVICE[3] of instance: sdma. 111 Select mux mode: ALT7 mux port: RXACTIVE of instance: usbphy2.

35.4.133 SW_MUX_CTL_PAD_EPDC_SDLE (IOMUXC_SMUXC_PEPDC_SDLE)

Address: IOMUXC_SMUXC_PEPDC_SDLE is 53FA_8000h base + 210h offset = 53FA_8210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

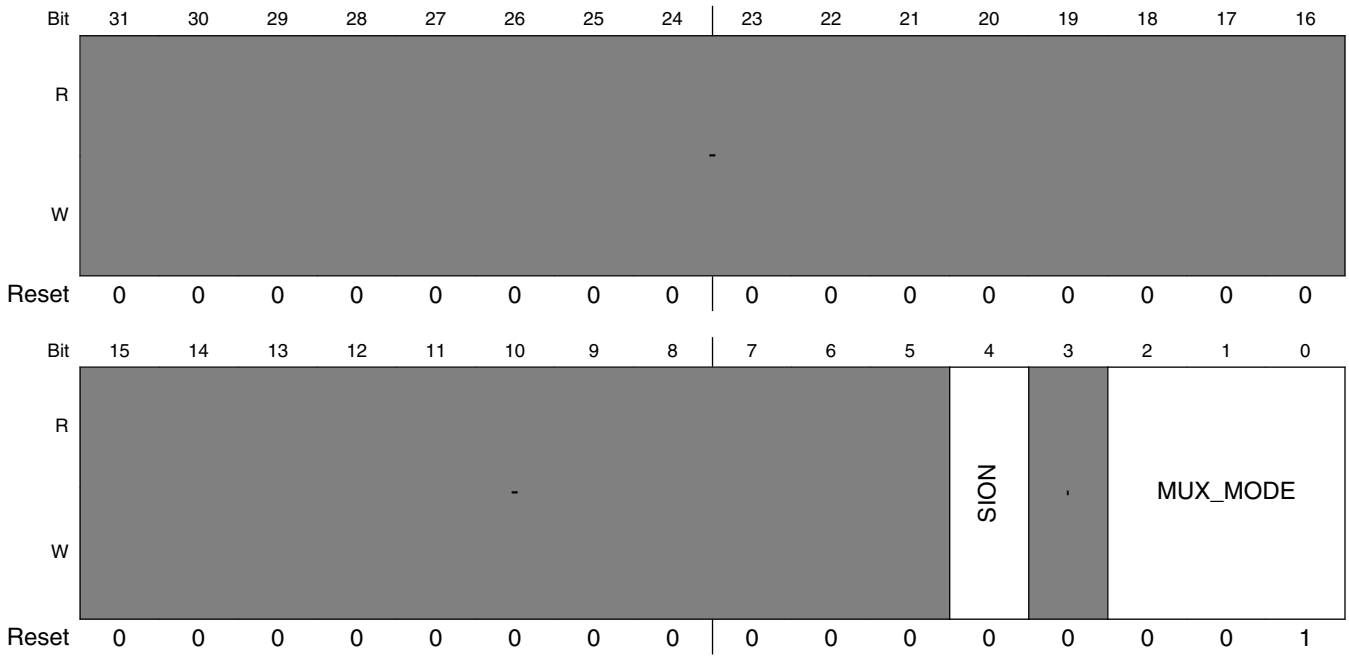
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_SDLE field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDLE. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_SDLE. **NOTE:** Pad EPDC_SDLE is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_8_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: SDLE of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[24] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[24] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[8] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD5_RXC of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_BUS_DEVICE[4] of instance: sdma. 111 Select mux mode: ALT7 mux port: RXERROR of instance: usbphy2.

35.4.134 SW_MUX_CTL_PAD_EPDC_SDCLKN (IOMUXC_SMUXC_PEPDC_SDCLKN)

Address: IOMUXC_SMUXC_PEPDC_SDCLKN is 53FA_8000h base + 214h offset = 53FA_8214h



IOMUXC_SMUXC_PEPDC_SDCLKN field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDCLKN. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_SDCLKN. **NOTE:** Pad EPDC_SDCLKN is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_9_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: SDCLKN of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[25] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[25] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[9] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD5_RXFS of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_BUS_ERROR of instance: sdma. 111 Select mux mode: ALT7 mux port: SIECLOCK of instance: usbphy2.

35.4.135 SW_MUX_CTL_PAD_EPDC_SDSHR (IOMUXC_SMUXC_PEPDC_SDSHR)

Address: IOMUXC_SMUXC_PEPDC_SDSHR is 53FA_8000h base + 218h offset = 53FA_8218h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

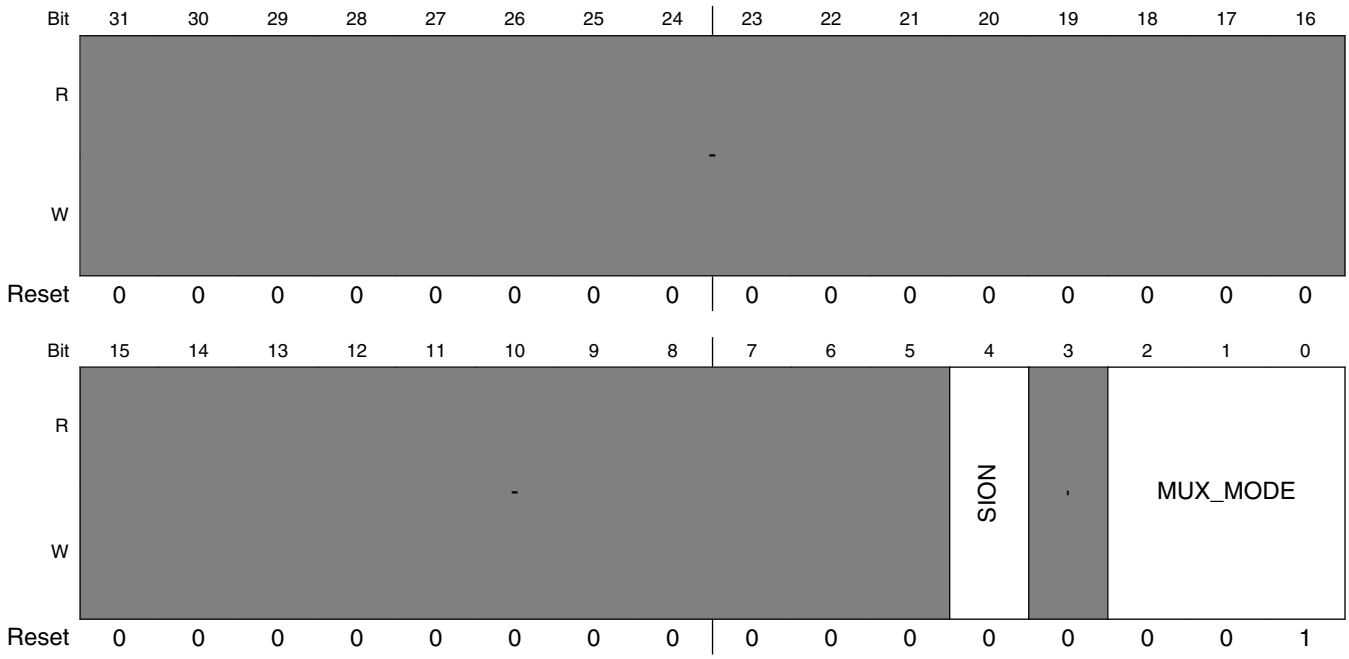
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_SDSHR field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDSHR. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_SDSHR. **NOTE:** Pad EPDC_SDSHR is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_10_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: SDSHR of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[26] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[26] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[10] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD4_TXD of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_BUS_RWB of instance: sdma. 111 Select mux mode: ALT7 mux port: LINESTATE[0] of instance: usbphy2.

35.4.136 SW_MUX_CTL_PAD_EPDC_PWRCOM (IOMUXC_SMUXC_PEPDC_PWRCOM)

Address: IOMUXC_SMUXC_PEPDC_PWRCOM is 53FA_8000h base + 21Ch offset = 53FA_821Ch

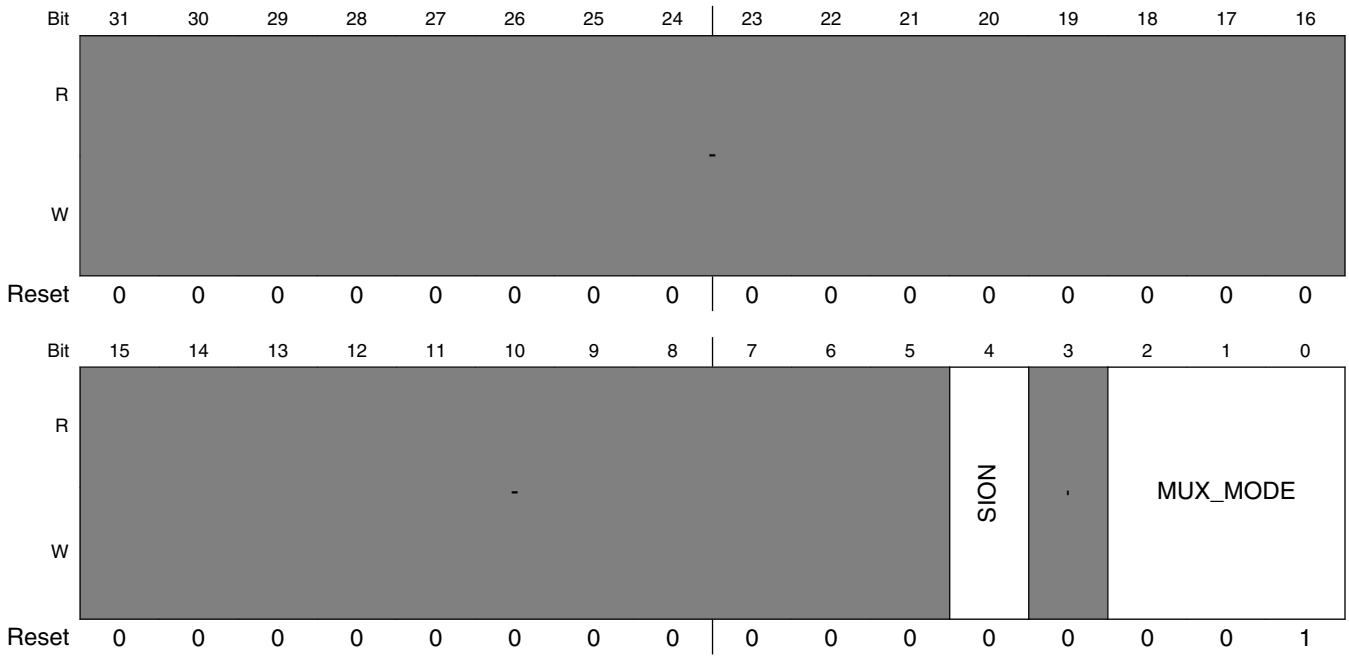


IOMUXC_SMUXC_PEPDC_PWRCOM field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_PWRCOM. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_PWRCOM. **NOTE:** Pad EPDC_PWRCOM is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_11_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: PWRCOM of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[27] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[27] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[11] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD4_TXC of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_CORE_RUN of instance: sdma. 111 Select mux mode: ALT7 mux port: LINESTATE[1] of instance: usbphy2.

35.4.137 SW_MUX_CTL_PAD_EPDC_PWRSTAT (IOMUXC_SMUXC_PEPDC_PWRSTAT)

Address: IOMUXC_SMUXC_PEPDC_PWRSTAT is 53FA_8000h base + 220h offset = 53FA_8220h



IOMUXC_SMUXC_PEPDC_PWRSTAT field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_PWRSTAT. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_PWRSTAT. **NOTE:** Pad EPDC_PWRSTAT is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_12_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: PWRSTAT of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[28] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[28] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[12] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD4_TXFS of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_MODE of instance: sdma. 111 Select mux mode: ALT7 mux port: VBUSVALID of instance: usbphy2.

35.4.138 SW_MUX_CTL_PAD_EPDC_PWRCTRL0 (IOMUXC_SMUXC_PEPDC_PWRCTRL0)

Address: IOMUXC_SMUXC_PEPDC_PWRCTRL0 is 53FA_8000h base + 224h offset = 53FA_8224h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_PWRCTRL0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_PWRCTRL0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_PWRCTRL0. **NOTE:** Pad EPDC_PWRCTRL0 is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_13_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: PWRCTRL[0] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[29] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[29] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[13] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD4_RXD of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_RTBUFFER_WRITE of instance: sdma. 111 Select mux mode: ALT7 mux port: AVALID of instance: usbphy2.

35.4.139 OMUXC_SW_MUX_CTL_PAD_EPDC_PWRCTRL1 (IOMUXC_OMUXC_SMUXC_PEPDC_PWRCTRL1)

Address: IOMUXC_OMUXC_SMUXC_PEPDC_PWRCTRL1 is 53FA_8000h base + 228h offset = 53FA_8228h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

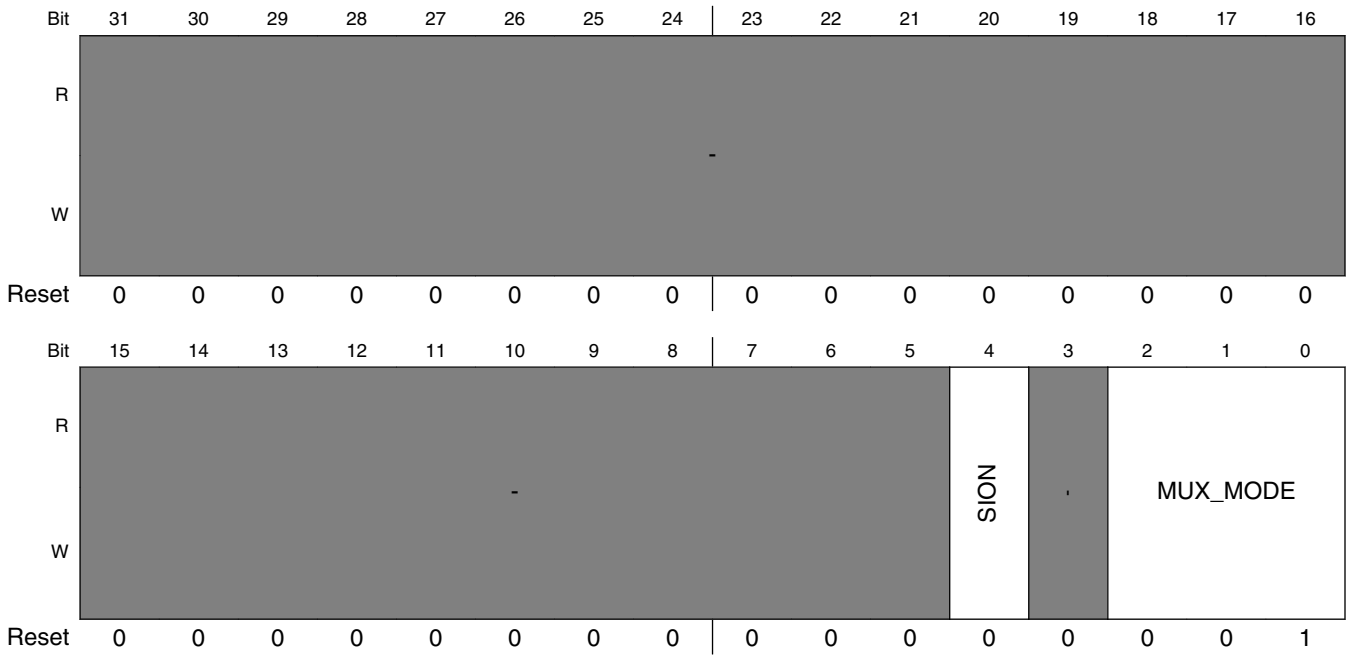
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_OMUXC_SMUXC_PEPDC_PWRCTRL1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_PWRCTRL1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_PWRCTRL1. **NOTE:** Pad EPDC_PWRCTRL1 is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_14_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: PWRCTRL[1] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[30] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[30] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[14] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD4_RXC of instance: audmux. 110 Select mux mode: ALT6 mux port: DEBUG_YIELD of instance: sdma. 111 Select mux mode: ALT7 mux port: ONBIST of instance: usbphy1.

35.4.140 SW_MUX_CTL_PAD_EPDC_PWRCTRL2 (IOMUXC_SMUXC_PEPDC_PWRCTRL2)

Address: IOMUXC_SMUXC_PEPDC_PWRCTRL2 is 53FA_8000h base + 22Ch offset = 53FA_822Ch

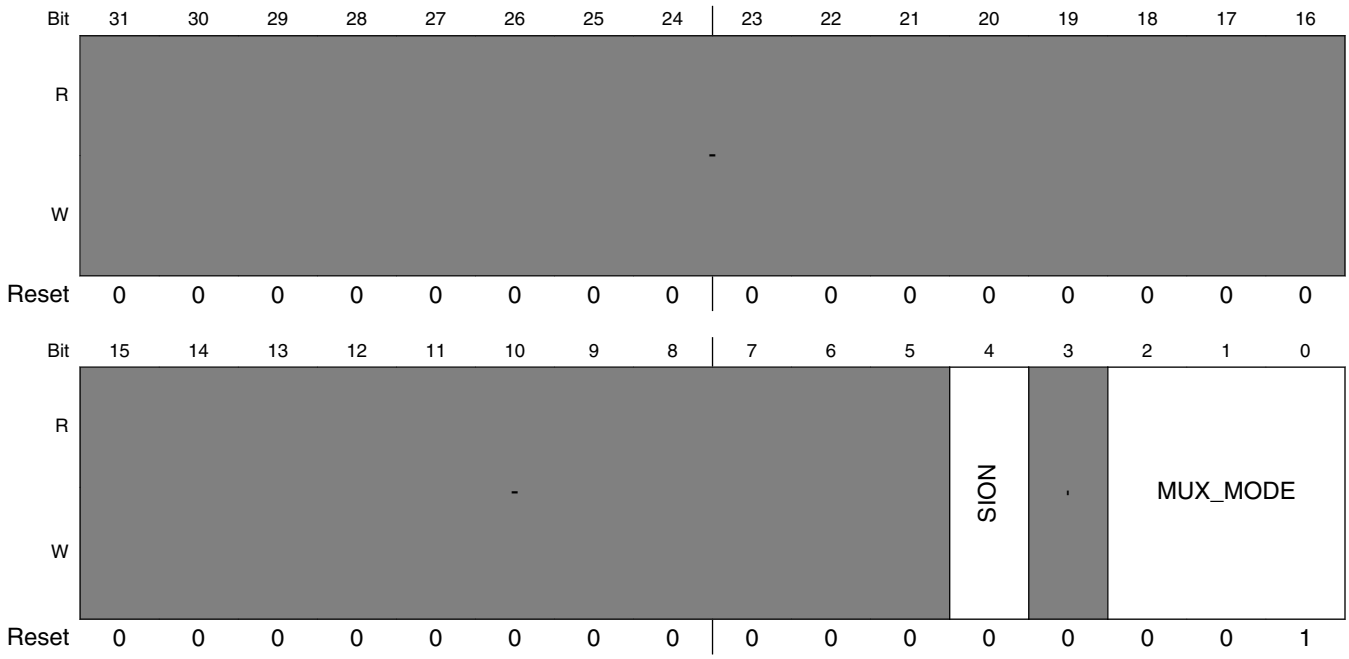


IOMUXC_SMUXC_PEPDC_PWRCTRL2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_PWRCTRL2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: EPDC_PWRCTRL2. **NOTE:** Pad EPDC_PWRCTRL2 is involved in Daisy Chain. • Config Register IOMUXC_AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT for mode ALT4. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_15_SELECT_INPUT for mode ALT3. • Config Register IOMUXC_SDMA_EVENTS_14_SELECT_INPUT for mode ALT6. 000 Select mux mode: ALT0 mux port: PWRCTRL[2] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[31] of instance: gpio3. 010 Select mux mode: ALT2 mux port: D[31] of instance: weimv2. 011 Select mux mode: ALT3 mux port: DAT[15] of instance: elcdif. 100 Select mux mode: ALT4 mux port: AUD4_RXFS of instance: audmux. 110 Select mux mode: ALT6 mux port: SDMA_EXT_EVENT[0] of instance: sdma. 111 Select mux mode: ALT7 mux port: ONBIST of instance: usbphy2.

35.4.141 SW_MUX_CTL_PAD_EPDC_PWRCTRL3 (IOMUXC_SMUXC_PEPDC_PWRCTRL3)

Address: IOMUXC_SMUXC_PEPDC_PWRCTRL3 is 53FA_8000h base + 230h offset = 53FA_8230h

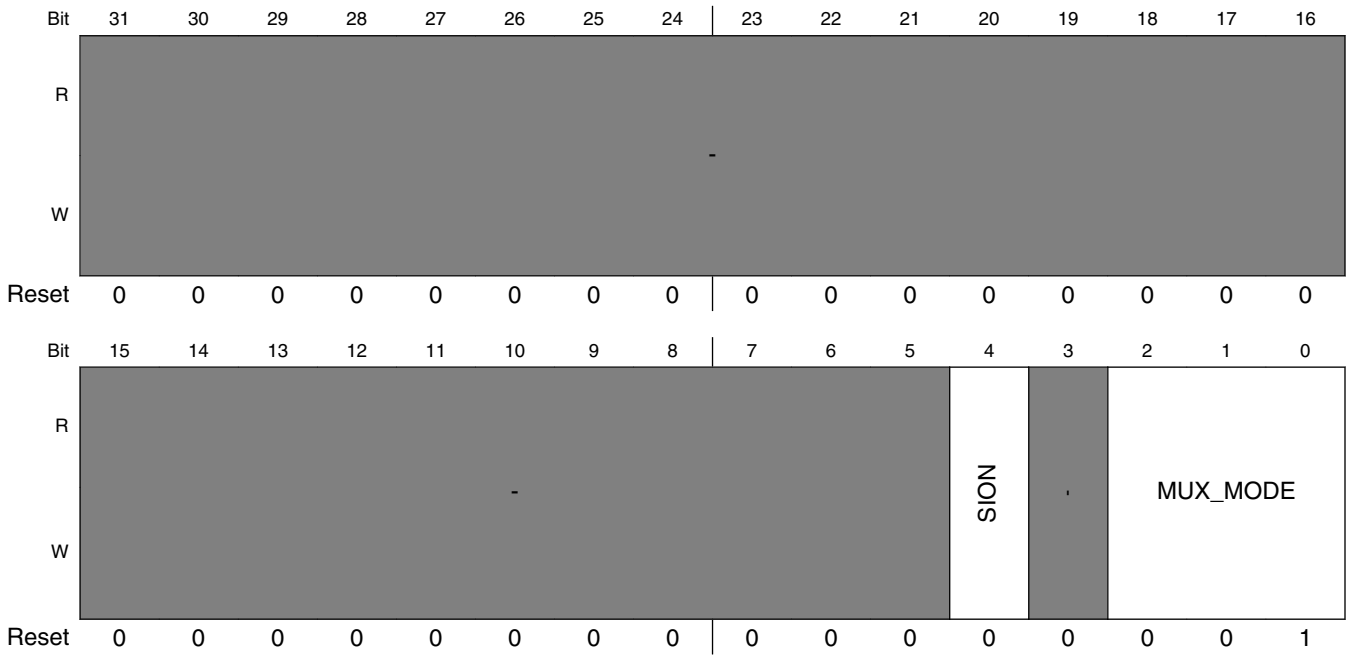


IOMUXC_SMUXC_PEPDC_PWRCTRL3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_PWRCTRL3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EPDC_PWRCTRL3. **NOTE:** Pad EPDC_PWRCTRL3 is involved in Daisy Chain. • Config Register IOMUXC_SDMA_EVENTS_15_SELECT_INPUT for mode ALT6. 000 Select mux mode: ALT0 mux port: PWRCTRL[3] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[20] of instance: gpio4. 010 Select mux mode: ALT2 mux port: EB[2] of instance: weimv2. 110 Select mux mode: ALT6 mux port: SDMA_EXT_EVENT[1] of instance: sdma. 111 Select mux mode: ALT7 mux port: BISTOK of instance: usbphy1.

35.4.142 SW_MUX_CTL_PAD_EPDC_VCOM0 (IOMUXC_SMUXC_PEPDC_VCOM0)

Address: IOMUXC_SMUXC_PEPDC_VCOM0 is 53FA_8000h base + 234h offset = 53FA_8234h



IOMUXC_SMUXC_PEPDC_VCOM0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_VCOM0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: EPDC_VCOM0. 000 Select mux mode: ALT0 mux port: VCOM[0] of instance: epdc. 001 Select mux mode: ALT1 mux port: GPIO[21] of instance: gpio4. 010 Select mux mode: ALT2 mux port: EB[3] of instance: weimv2. 111 Select mux mode: ALT7 mux port: BISTOK of instance: usbphy2.

35.4.143 SW_MUX_CTL_PAD_EPDC_VCOM1 (IOMUXC_SMUXC_PEPDC_VCOM1)

Address: IOMUXC_SMUXC_PEPDC_VCOM1 is 53FA_8000h base + 238h offset = 53FA_8238h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

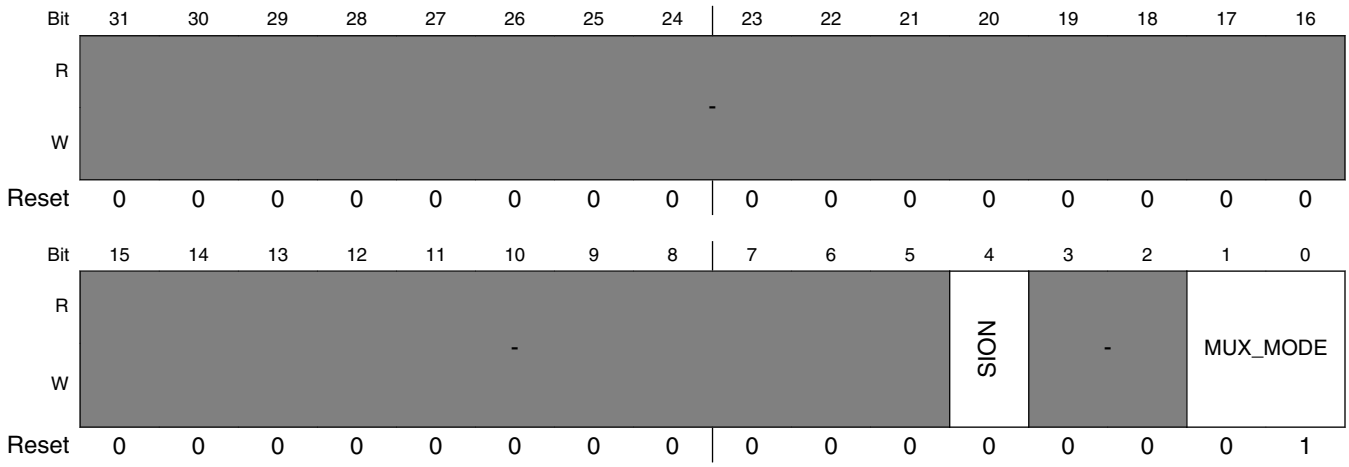
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_VCOM1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_VCOM1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_VCOM1. 00 Select mux mode: ALT0 mux port: VCOM[1] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[22] of instance: gpio4. 10 Select mux mode: ALT2 mux port: CS[3] of instance: weimv2.

35.4.144 SW_MUX_CTL_PAD_EPDC_BDR0 (IOMUXC_SMUXC_PEPDC_BDR0)

Address: IOMUXC_SMUXC_PEPDC_BDR0 is 53FA_8000h base + 23Ch offset = 53FA_823Ch



IOMUXC_SMUXC_PEPDC_BDR0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_BDR0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_BDR0. **NOTE:** Pad EPDC_BDR0 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_7_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: BDR[0] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[23] of instance: gpio4. 11 Select mux mode: ALT3 mux port: DAT[7] of instance: elcdif.

35.4.145 SW_MUX_CTL_PAD_EPDC_BDR1 (IOMUXC_SMUXC_PEPDC_BDR1)

Address: IOMUXC_SMUXC_PEPDC_BDR1 is 53FA_8000h base + 240h offset = 53FA_8240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

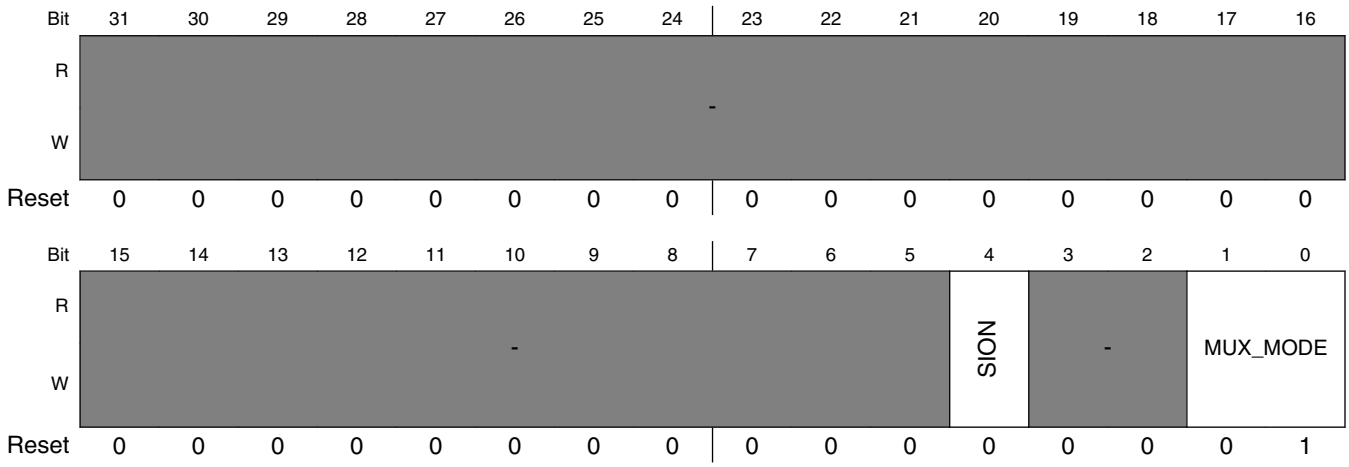
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_BDR1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_BDR1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_BDR1. **NOTE:** Pad EPDC_BDR1 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_6_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: BDR[1] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[24] of instance: gpio4. 11 Select mux mode: ALT3 mux port: DAT[6] of instance: elcdif.

35.4.146 SW_MUX_CTL_PAD_EPDC_SDCE0 (IOMUXC_SMUXC_PEPDC_SDCE0)

Address: IOMUXC_SMUXC_PEPDC_SDCE0 is 53FA_8000h base + 244h offset = 53FA_8244h

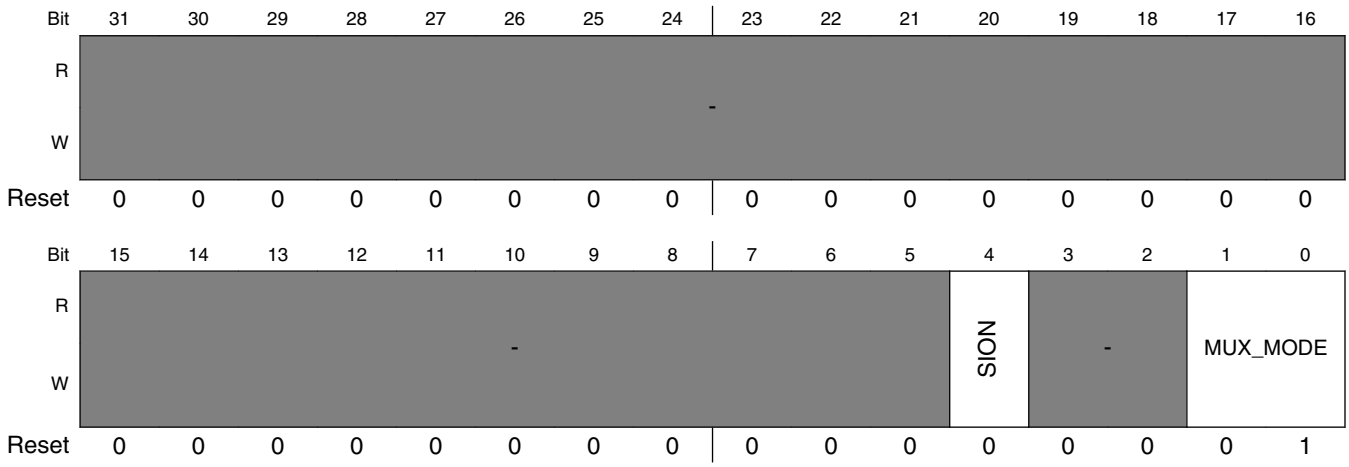


IOMUXC_SMUXC_PEPDC_SDCE0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDCE0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_SDCE0. **NOTE:** Pad EPDC_SDCE0 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_5_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: SDCE[0] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[25] of instance: gpio4. 11 Select mux mode: ALT3 mux port: DAT[5] of instance: elcdif.

35.4.147 SW_MUX_CTL_PAD_EPDC_SDCE1 (IOMUXC_SMUXC_PEPDC_SDCE1)

Address: IOMUXC_SMUXC_PEPDC_SDCE1 is 53FA_8000h base + 248h offset = 53FA_8248h



IOMUXC_SMUXC_PEPDC_SDCE1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDCE1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_SDCE1. **NOTE:** Pad EPDC_SDCE1 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_4_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: SDCE[1] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[26] of instance: gpio4. 11 Select mux mode: ALT3 mux port: DAT[4] of instance: elcdif.

35.4.148 SW_MUX_CTL_PAD_EPDC_SDCE2 (IOMUXC_SMUXC_PEPDC_SDCE2)

Address: IOMUXC_SMUXC_PEPDC_SDCE2 is 53FA_8000h base + 24Ch offset = 53FA_824Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

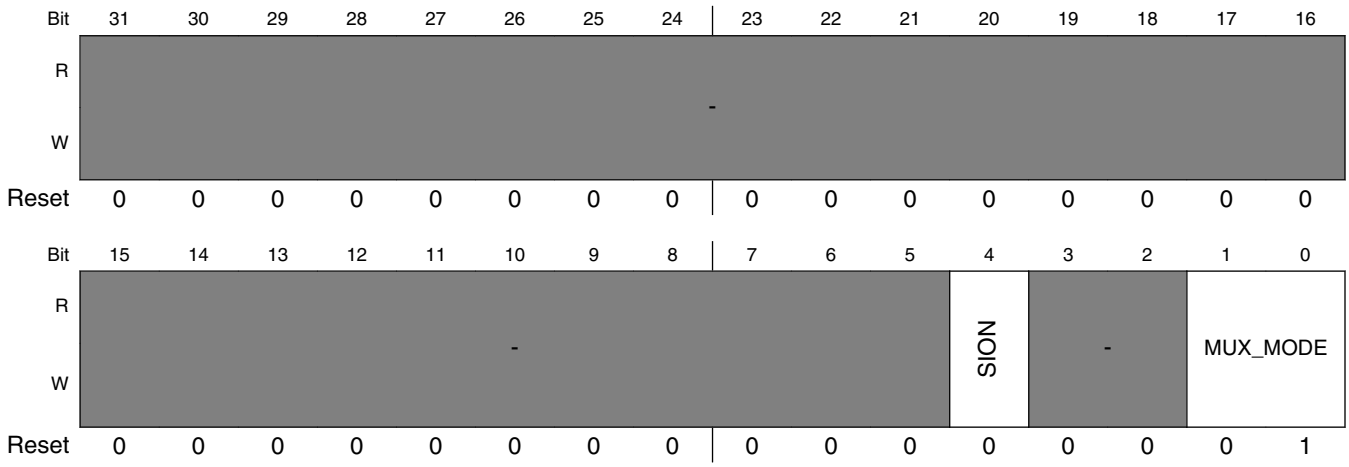
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_SDCE2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDCE2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_SDCE2. **NOTE:** Pad EPDC_SDCE2 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_3_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: SDCE[2] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[27] of instance: gpio4. 11 Select mux mode: ALT3 mux port: DAT[3] of instance: elcdif.

35.4.149 SW_MUX_CTL_PAD_EPDC_SDCE3 (IOMUXC_SMUXC_PEPDC_SDCE3)

Address: IOMUXC_SMUXC_PEPDC_SDCE3 is 53FA_8000h base + 250h offset = 53FA_8250h



IOMUXC_SMUXC_PEPDC_SDCE3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDCE3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_SDCE3. **NOTE:** Pad EPDC_SDCE3 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_2_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: SDCE[3] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[28] of instance: gpio4. 11 Select mux mode: ALT3 mux port: DAT[2] of instance: elcdif.

35.4.150 SW_MUX_CTL_PAD_EPDC_SDCE4 (IOMUXC_SMUXC_PEPDC_SDCE4)

Address: IOMUXC_SMUXC_PEPDC_SDCE4 is 53FA_8000h base + 254h offset = 53FA_8254h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

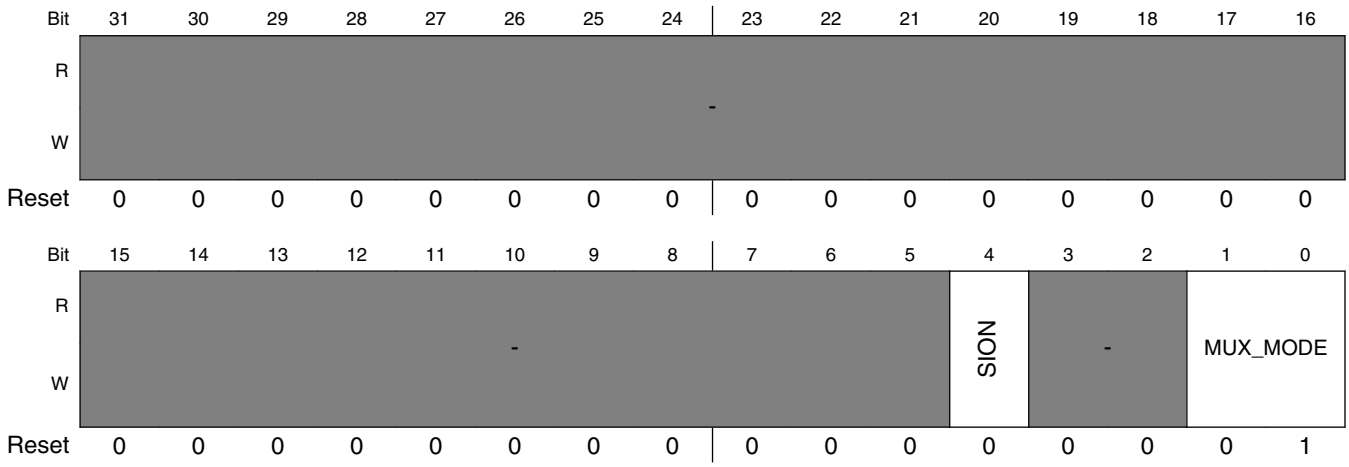
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-		MUX_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IOMUXC_SMUXC_PEPDC_SDCE4 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDCE4. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_SDCE4. **NOTE:** Pad EPDC_SDCE4 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_1_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: SDCE[4] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[29] of instance: gpio4. 11 Select mux mode: ALT3 mux port: DAT[1] of instance: elcdif.

35.4.151 SW_MUX_CTL_PAD_EPDC_SDCE5 (IOMUXC_SMUXC_PEPDC_SDCE5)

Address: IOMUXC_SMUXC_PEPDC_SDCE5 is 53FA_8000h base + 258h offset = 53FA_8258h

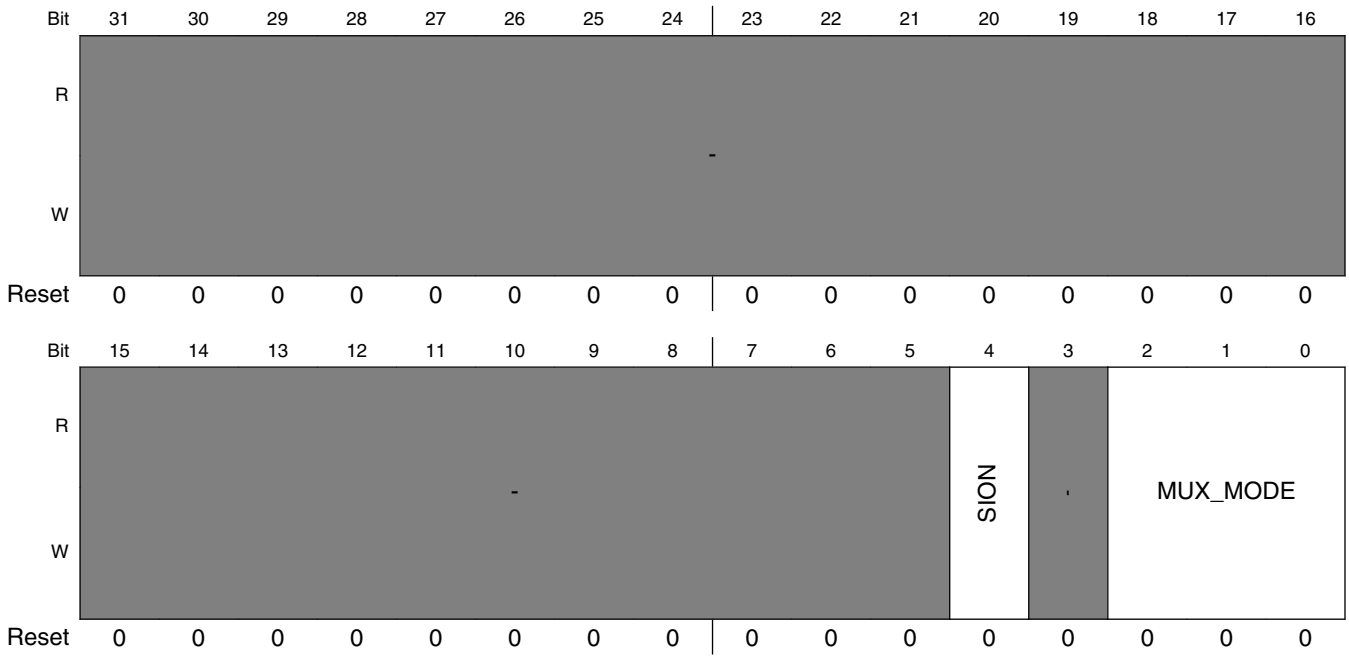


IOMUXC_SMUXC_PEPDC_SDCE5 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EPDC_SDCE5. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–2 -	Reserved
1–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EPDC_SDCE5. **NOTE:** Pad EPDC_SDCE5 is involved in Daisy Chain. • Config Register IOMUXC_ELCDIF_LCDIF_RXDATA_0_SELECT_INPUT for mode ALT3. 00 Select mux mode: ALT0 mux port: SDCE[5] of instance: epdc. 01 Select mux mode: ALT1 mux port: GPIO[30] of instance: gpio4. 11 Select mux mode: ALT3 mux port: DAT[0] of instance: elcdif.

35.4.152 SW_MUX_CTL_PAD_EIM_DA0 (IOMUXC_SMUXC_PEIM_DA0)

Address: IOMUXC_SMUXC_PEIM_DA0 is 53FA_8000h base + 25Ch offset = 53FA_825Ch

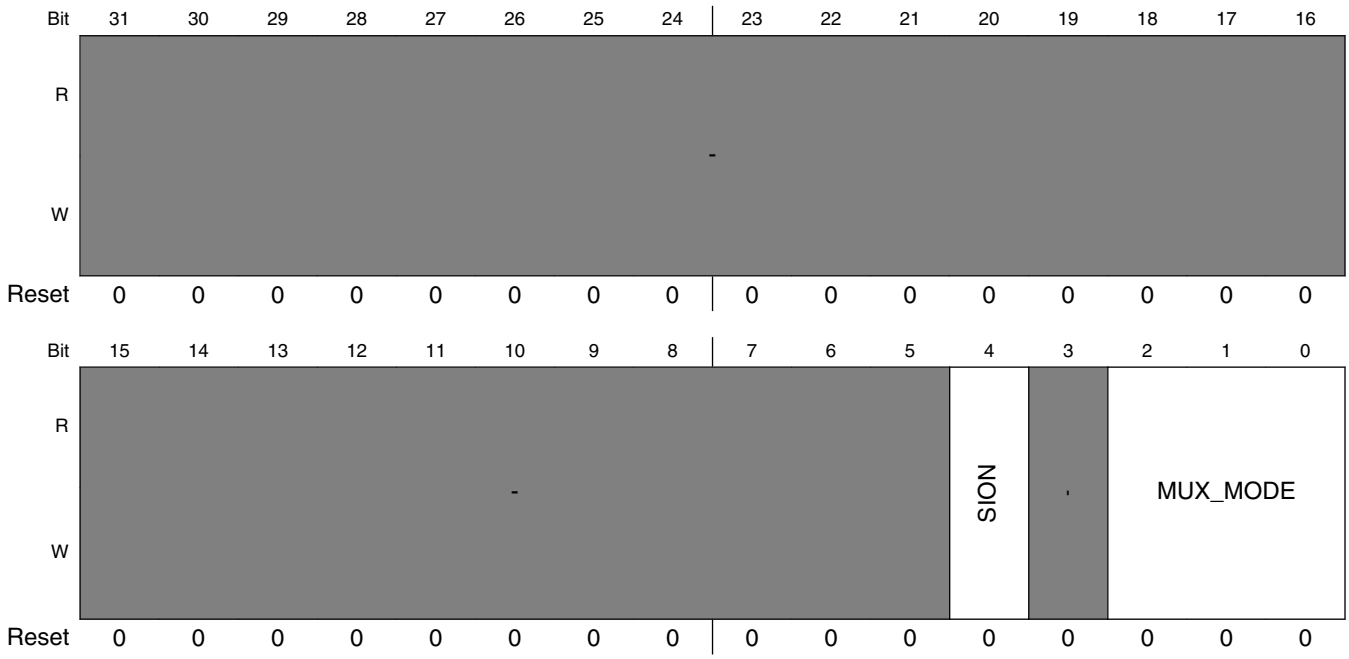


IOMUXC_SMUXC_PEIM_DA0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA0. **NOTE:** Pad EIM_DA0 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: A[0] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[0] of instance: gpio1. 011 Select mux mode: ALT3 mux port: COL[4] of instance: kpp. 110 Select mux mode: ALT6 mux port: TRACE[0] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG1[0] of instance: src.

35.4.153 SW_MUX_CTL_PAD_EIM_DA1 (IOMUXC_SMUXC_PEIM_DA1)

Address: IOMUXC_SMUXC_PEIM_DA1 is 53FA_8000h base + 260h offset = 53FA_8260h

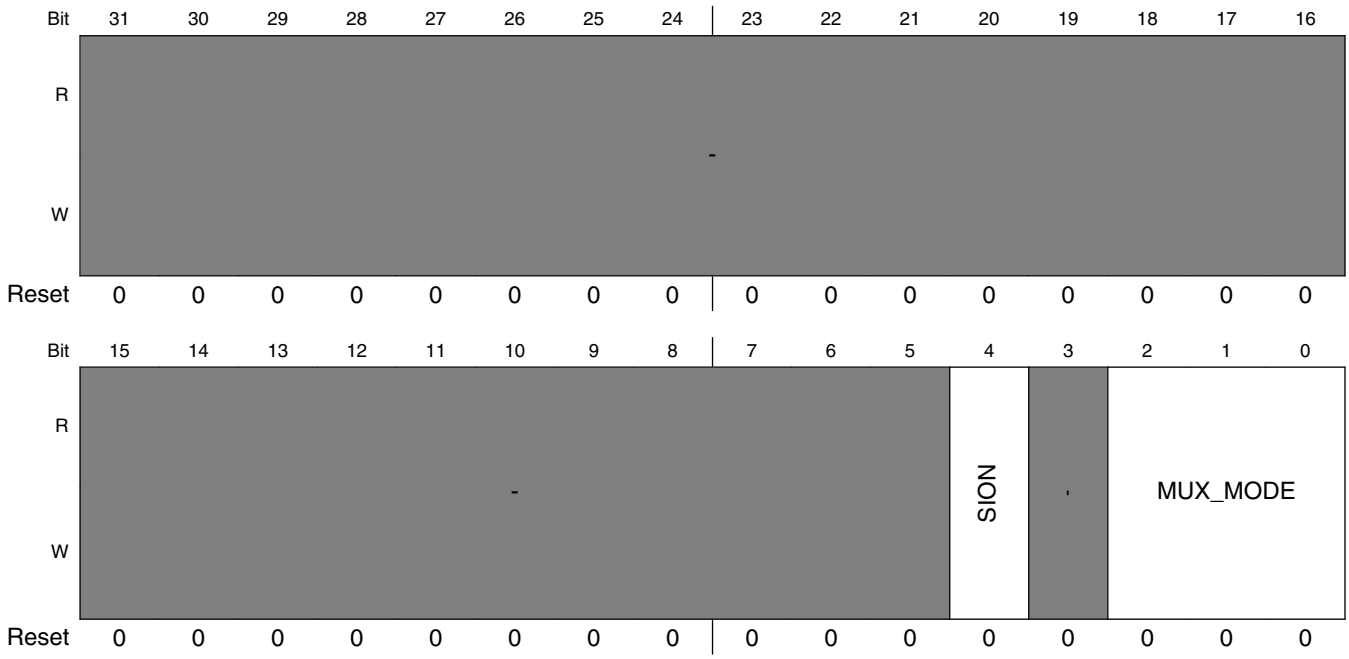


IOMUXC_SMUXC_PEIM_DA1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA1. **NOTE:** Pad EIM_DA1 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: A[1] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[1] of instance: gpio1. 011 Select mux mode: ALT3 mux port: ROW[4] of instance: kpp. 110 Select mux mode: ALT6 mux port: TRACE[1] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG1[1] of instance: src.

35.4.154 SW_MUX_CTL_PAD_EIM_DA2 (IOMUXC_SMUXC_PEIM_DA2)

Address: IOMUXC_SMUXC_PEIM_DA2 is 53FA_8000h base + 264h offset = 53FA_8264h

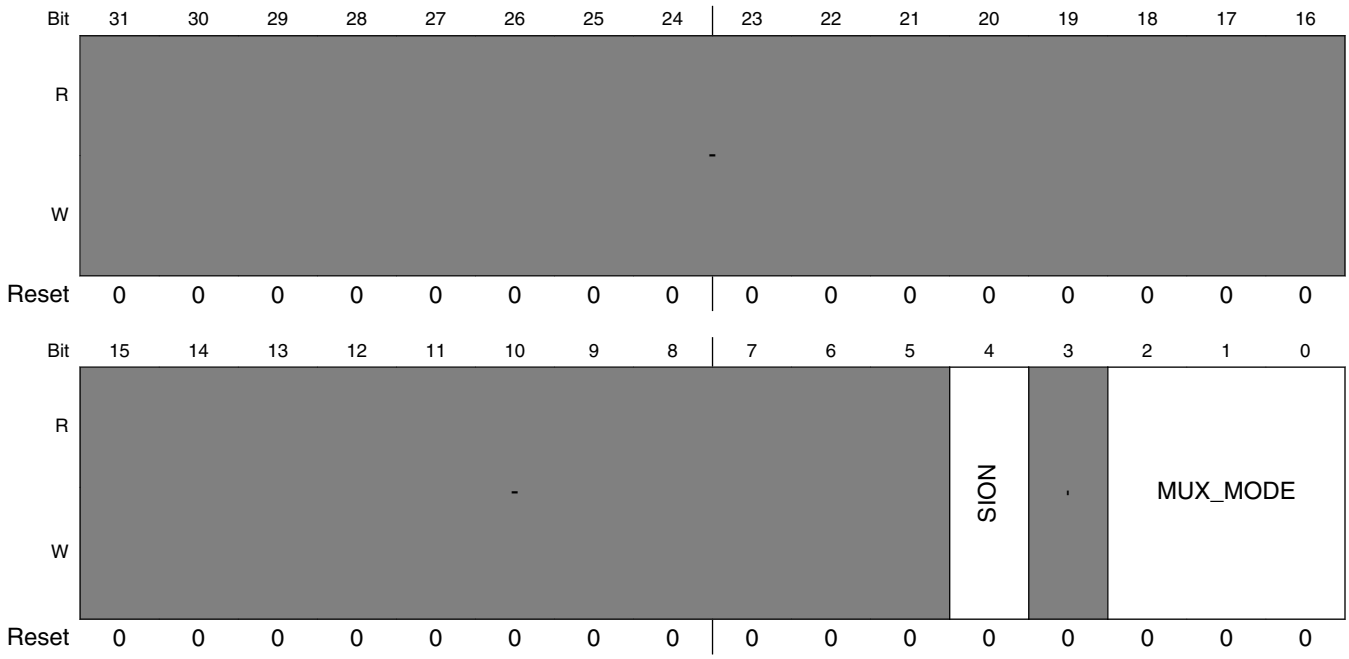


IOMUXC_SMUXC_PEIM_DA2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA2. **NOTE:** Pad EIM_DA2 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: A[2] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[2] of instance: gpio1. 011 Select mux mode: ALT3 mux port: COL[5] of instance: kpp. 110 Select mux mode: ALT6 mux port: TRACE[2] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG1[2] of instance: src.

35.4.155 SW_MUX_CTL_PAD_EIM_DA3 (IOMUXC_SMUXC_PEIM_DA3)

Address: IOMUXC_SMUXC_PEIM_DA3 is 53FA_8000h base + 268h offset = 53FA_8268h

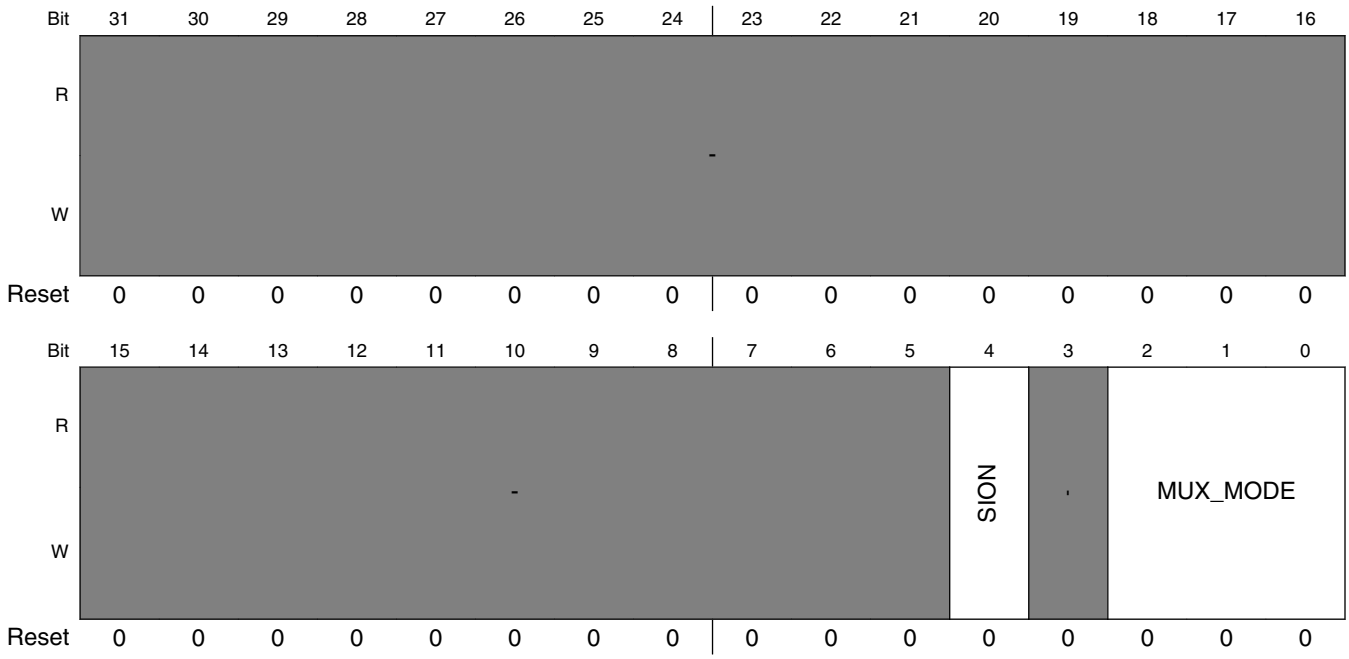


IOMUXC_SMUXC_PEIM_DA3 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA3. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA3. **NOTE:** Pad EIM_DA3 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: A[3] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[3] of instance: gpio1. 011 Select mux mode: ALT3 mux port: ROW[5] of instance: kpp. 110 Select mux mode: ALT6 mux port: TRACE[3] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG1[3] of instance: src.

35.4.156 SW_MUX_CTL_PAD_EIM_DA4 (IOMUXC_SMUXC_PEIM_DA4)

Address: IOMUXC_SMUXC_PEIM_DA4 is 53FA_8000h base + 26Ch offset = 53FA_826Ch

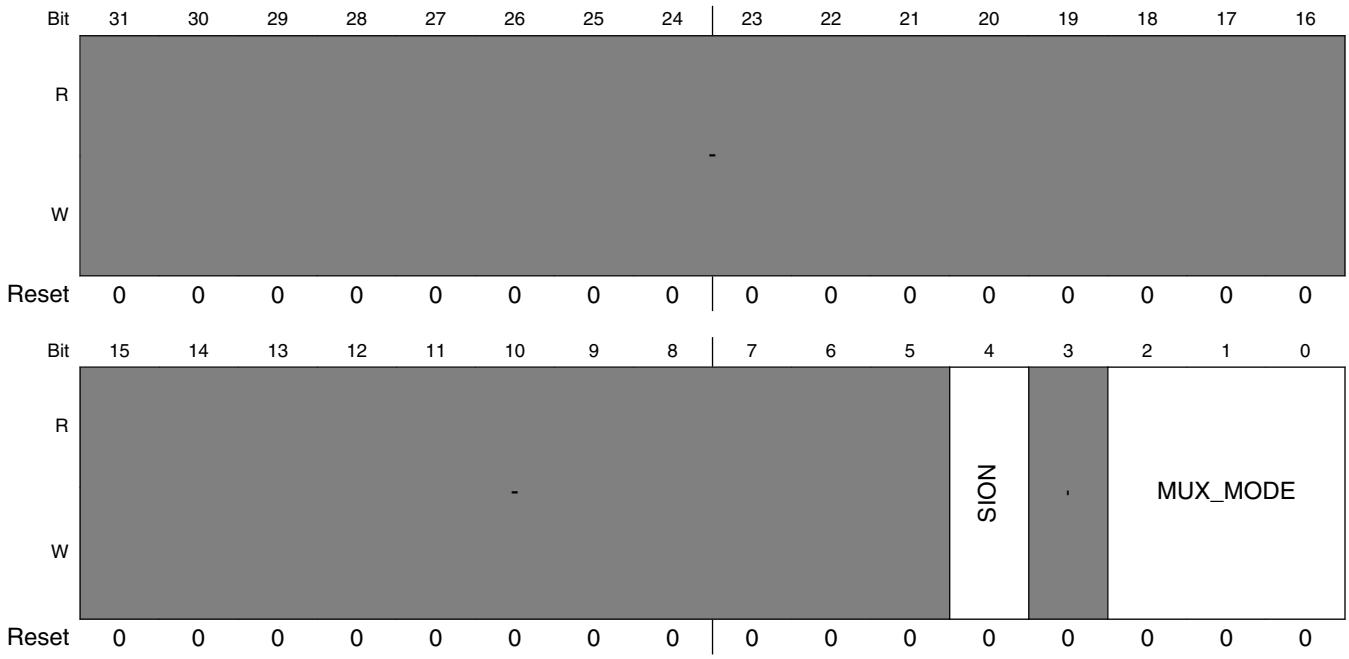


IOMUXC_SMUXC_PEIM_DA4 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA4. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA4. **NOTE:** Pad EIM_DA4 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: A[4] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[4] of instance: gpio1. 011 Select mux mode: ALT3 mux port: COL[6] of instance: kpp. 110 Select mux mode: ALT6 mux port: TRACE[4] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG1[4] of instance: src.

35.4.157 SW_MUX_CTL_PAD_EIM_DA5 (IOMUXC_SMUXC_PEIM_DA5)

Address: IOMUXC_SMUXC_PEIM_DA5 is 53FA_8000h base + 270h offset = 53FA_8270h

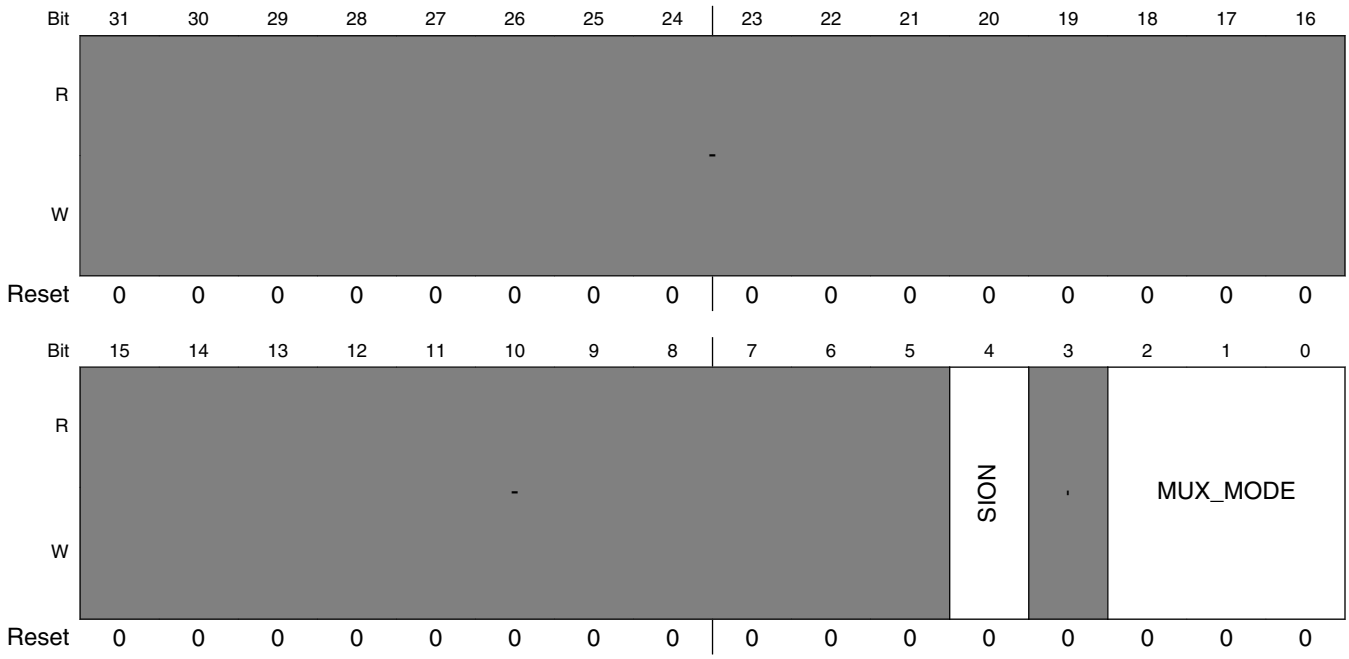


IOMUXC_SMUXC_PEIM_DA5 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA5. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA5. **NOTE:** Pad EIM_DA5 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: A[5] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[5] of instance: gpio1. 011 Select mux mode: ALT3 mux port: ROW[6] of instance: kpp. 110 Select mux mode: ALT6 mux port: TRACE[5] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG1[5] of instance: src.

35.4.158 SW_MUX_CTL_PAD_EIM_DA6 (IOMUXC_SMUXC_PEIM_DA6)

Address: IOMUXC_SMUXC_PEIM_DA6 is 53FA_8000h base + 274h offset = 53FA_8274h

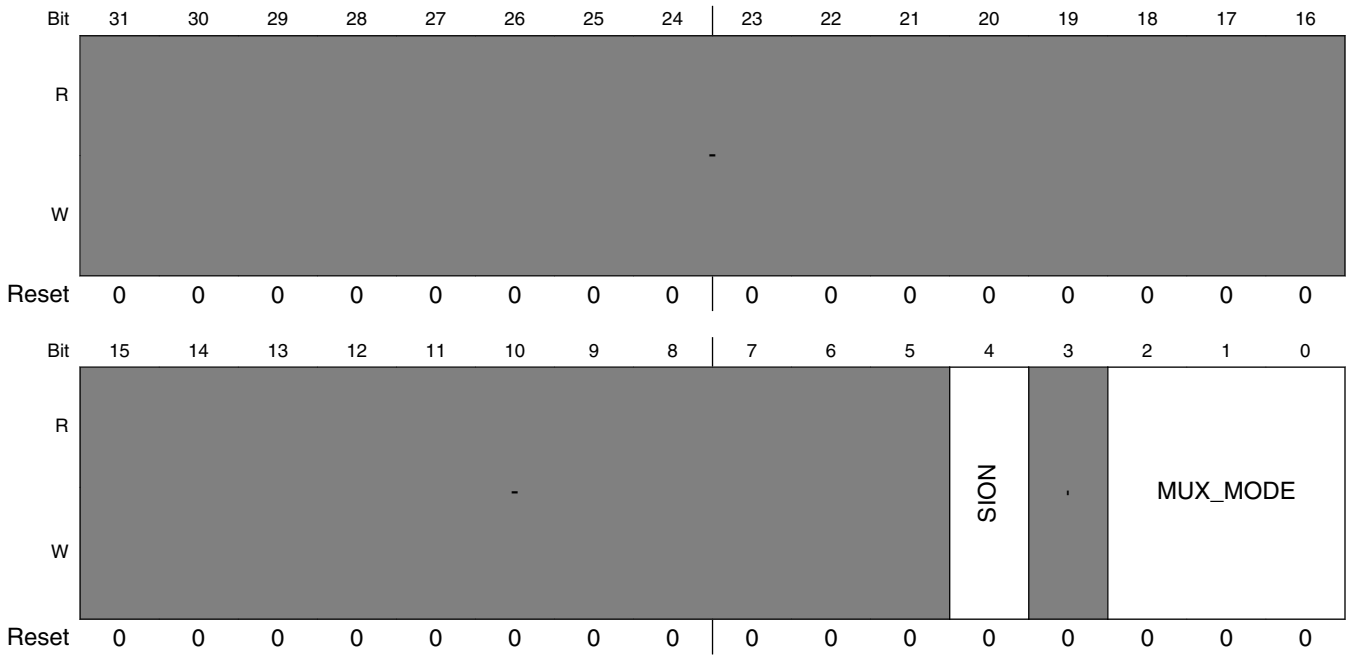


IOMUXC_SMUXC_PEIM_DA6 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA6. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA6. **NOTE:** Pad EIM_DA6 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: A[6] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[6] of instance: gpio1. 011 Select mux mode: ALT3 mux port: COL[7] of instance: kpp. 110 Select mux mode: ALT6 mux port: TRACE[6] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG1[6] of instance: src.

35.4.159 SW_MUX_CTL_PAD_EIM_DA7 (IOMUXC_SMUXC_PEIM_DA7)

Address: IOMUXC_SMUXC_PEIM_DA7 is 53FA_8000h base + 278h offset = 53FA_8278h

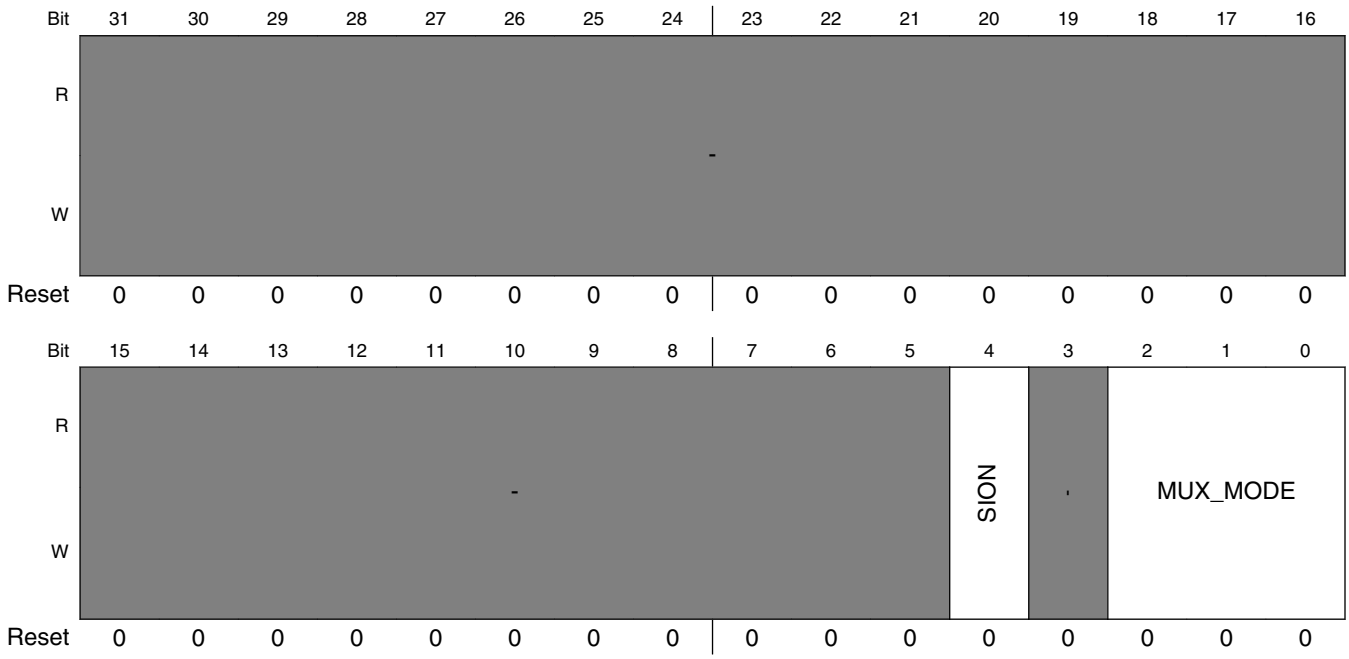


IOMUXC_SMUXC_PEIM_DA7 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA7. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA7. **NOTE:** Pad EIM_DA7 is involved in Daisy Chain. • Config Register IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT for mode ALT3. 000 Select mux mode: ALT0 mux port: A[7] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[7] of instance: gpio1. 011 Select mux mode: ALT3 mux port: ROW[7] of instance: kpp. 110 Select mux mode: ALT6 mux port: TRACE[7] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG1[7] of instance: src.

35.4.160 SW_MUX_CTL_PAD_EIM_DA8 (IOMUXC_SMUXC_PEIM_DA8)

Address: IOMUXC_SMUXC_PEIM_DA8 is 53FA_8000h base + 27Ch offset = 53FA_827Ch

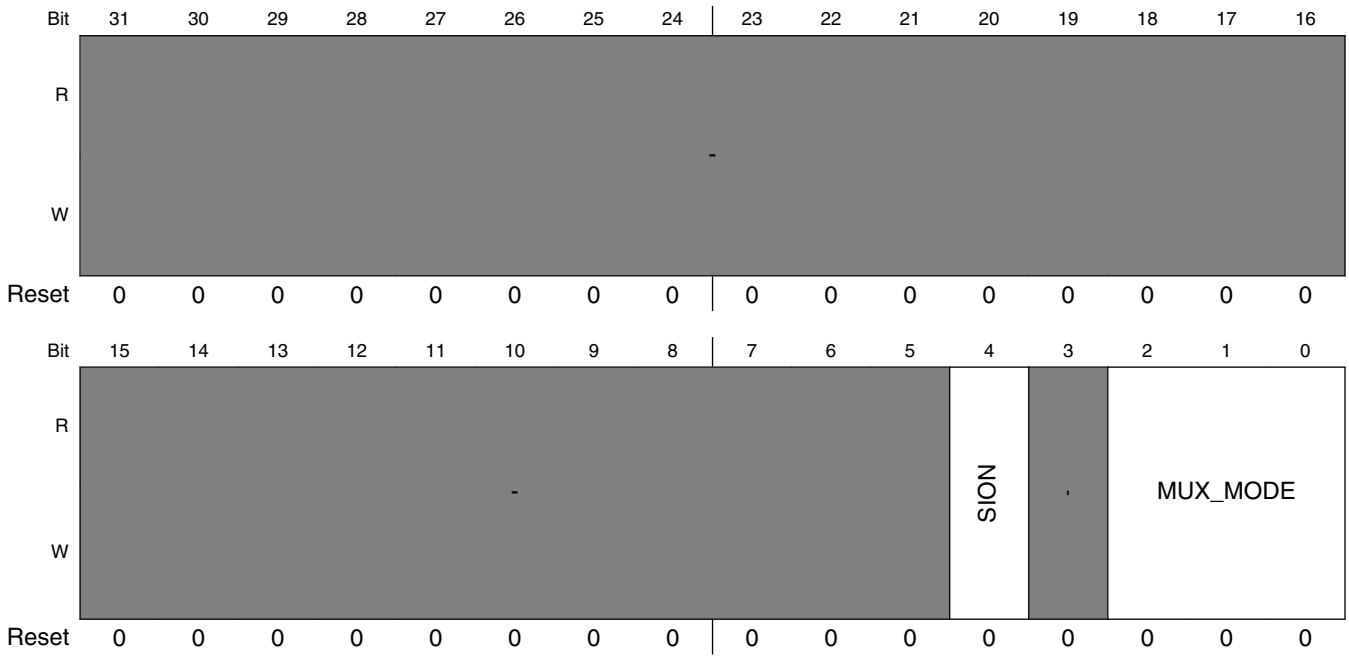


IOMUXC_SMUXC_PEIM_DA8 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA8. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA8. 000 Select mux mode: ALT0 mux port: A[8] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[8] of instance: gpio1. 010 Select mux mode: ALT2 mux port: CLE of instance: rawnand. 110 Select mux mode: ALT6 mux port: TRACE[8] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG2[0] of instance: src.

35.4.161 SW_MUX_CTL_PAD_EIM_DA9 (IOMUXC_SMUXC_PEIM_DA9)

Address: IOMUXC_SMUXC_PEIM_DA9 is 53FA_8000h base + 280h offset = 53FA_8280h

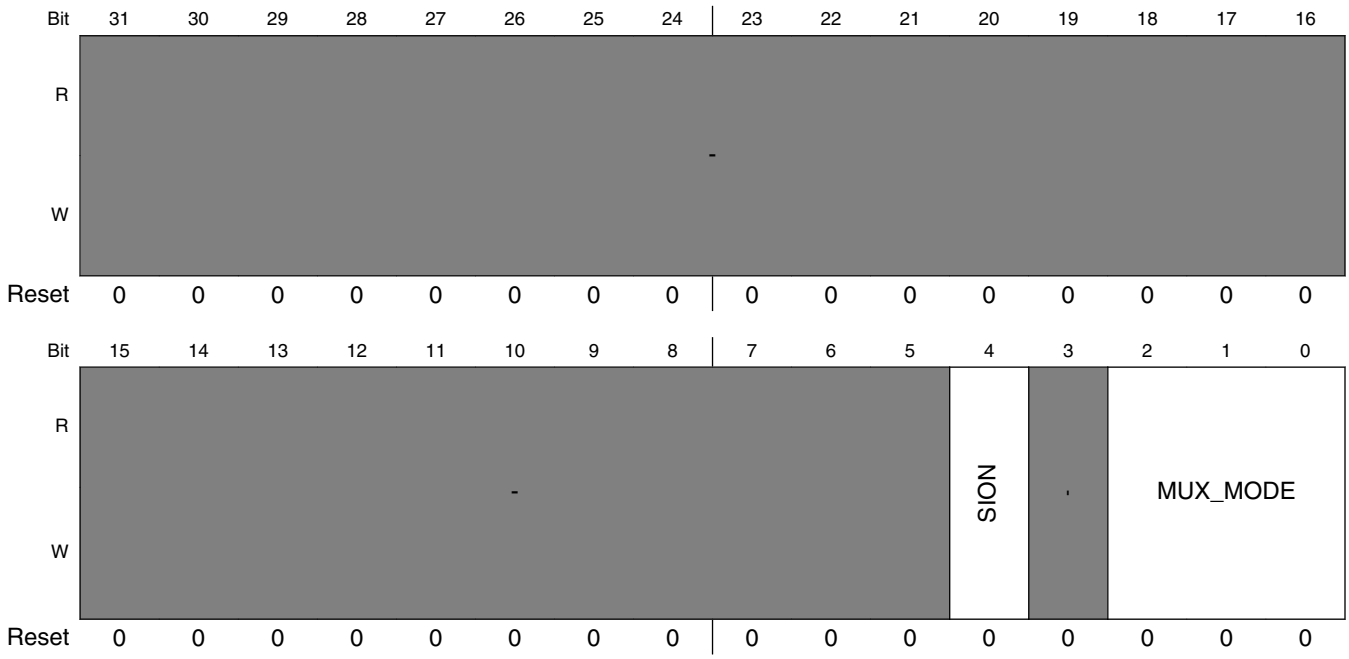


IOMUXC_SMUXC_PEIM_DA9 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA9. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA9. 000 Select mux mode: ALT0 mux port: A[9] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[9] of instance: gpio1. 010 Select mux mode: ALT2 mux port: ALE of instance: rawnand. 110 Select mux mode: ALT6 mux port: TRACE[9] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG2[1] of instance: src.

35.4.162 SW_MUX_CTL_PAD_EIM_DA10 (IOMUXC_SMUXC_PEIM_DA10)

Address: IOMUXC_SMUXC_PEIM_DA10 is 53FA_8000h base + 284h offset = 53FA_8284h

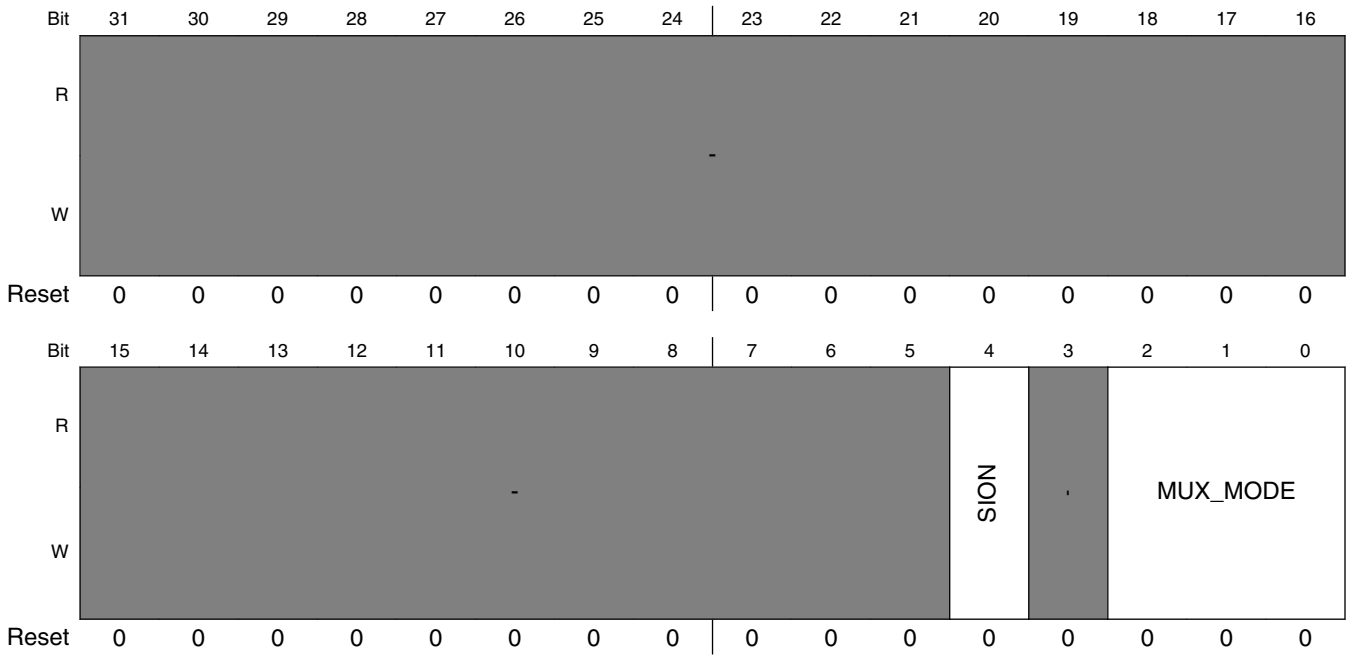


IOMUXC_SMUXC_PEIM_DA10 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA10. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA10. 000 Select mux mode: ALT0 mux port: A[10] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[10] of instance: gpio1. 010 Select mux mode: ALT2 mux port: CEN[0] of instance: rawnand. 110 Select mux mode: ALT6 mux port: TRACE[10] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG2[2] of instance: src.

35.4.163 SW_MUX_CTL_PAD_EIM_DA11 (IOMUXC_SMUXC_PEIM_DA11)

Address: IOMUXC_SMUXC_PEIM_DA11 is 53FA_8000h base + 288h offset = 53FA_8288h

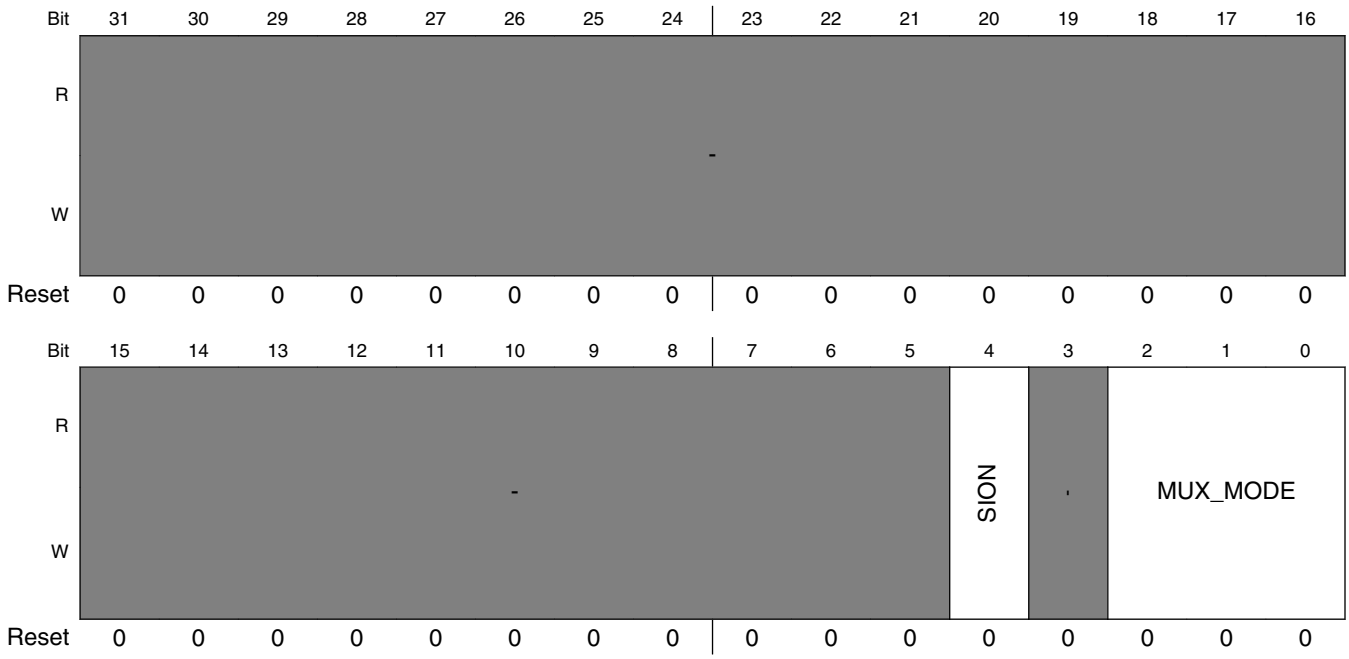


IOMUXC_SMUXC_PEIM_DA11 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA11. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_DA11. 000 Select mux mode: ALT0 mux port: A[11] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[11] of instance: gpio1. 010 Select mux mode: ALT2 mux port: CEN[1] of instance: rawnand. 110 Select mux mode: ALT6 mux port: TRACE[11] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG2[3] of instance: src.

35.4.164 SW_MUX_CTL_PAD_EIM_DA12 (IOMUXC_SMUXC_PEIM_DA12)

Address: IOMUXC_SMUXC_PEIM_DA12 is 53FA_8000h base + 28Ch offset = 53FA_828Ch

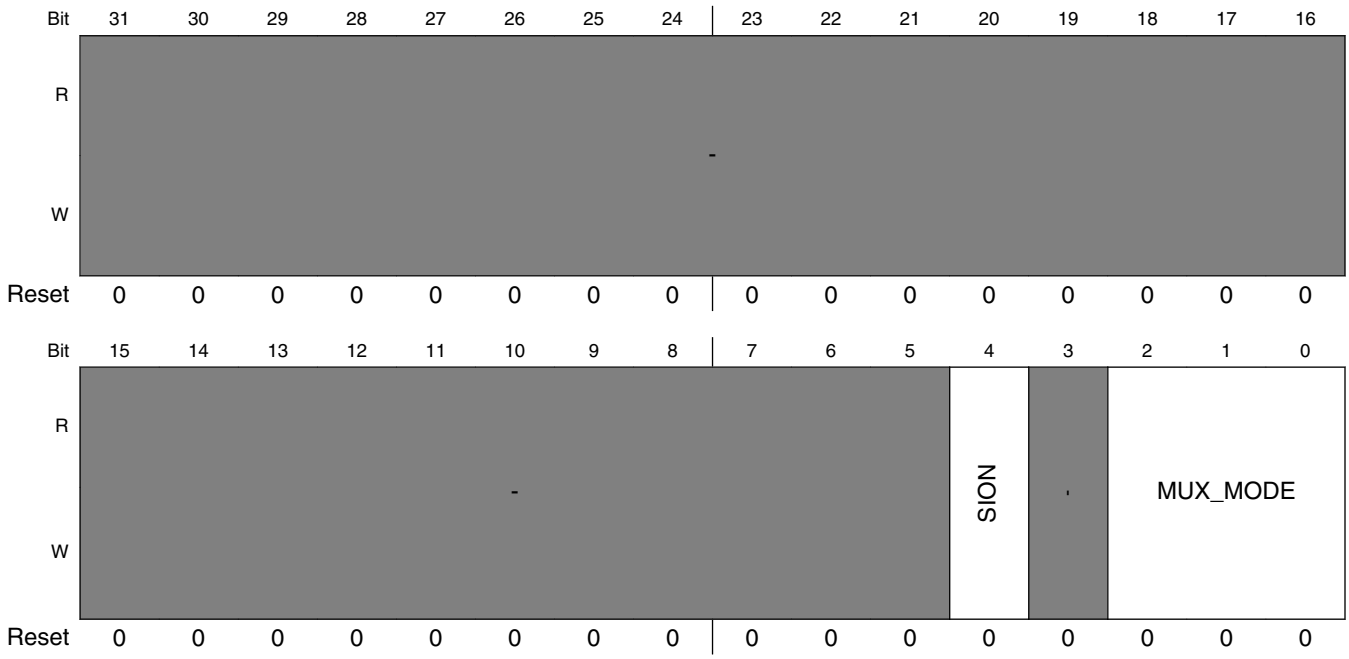


IOMUXC_SMUXC_PEIM_DA12 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA12. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EIM_DA12. 000 Select mux mode: ALT0 mux port: A[12] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[12] of instance: gpio1. 010 Select mux mode: ALT2 mux port: CEN[2] of instance: rawnand. 011 Select mux mode: ALT3 mux port: SDCE[6] of instance: epdc. 110 Select mux mode: ALT6 mux port: TRACE[12] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG2[4] of instance: src.

35.4.165 SW_MUX_CTL_PAD_EIM_DA13 (IOMUXC_SMUXC_PEIM_DA13)

Address: IOMUXC_SMUXC_PEIM_DA13 is 53FA_8000h base + 290h offset = 53FA_8290h

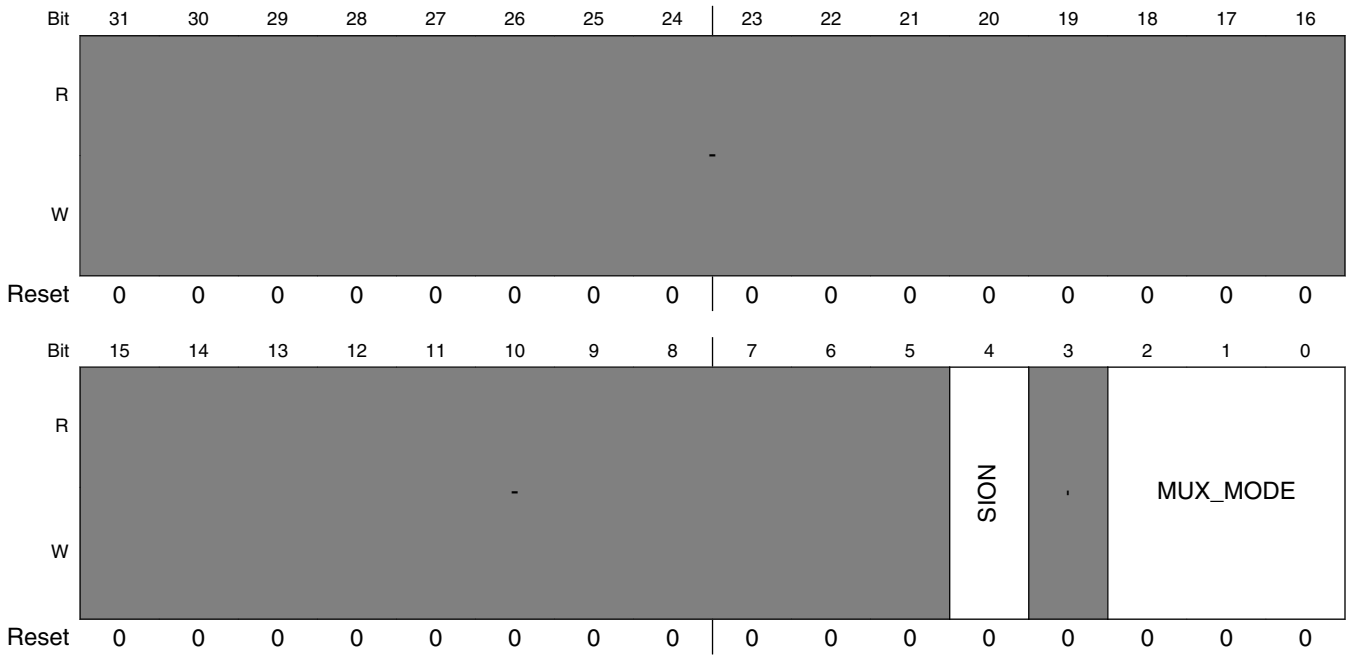


IOMUXC_SMUXC_PEIM_DA13 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA13. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EIM_DA13. 000 Select mux mode: ALT0 mux port: A[13] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[13] of instance: gpio1. 010 Select mux mode: ALT2 mux port: CEN[3] of instance: rawnand. 011 Select mux mode: ALT3 mux port: SDCE[7] of instance: epdc. 110 Select mux mode: ALT6 mux port: TRACE[13] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG2[5] of instance: src.

35.4.166 SW_MUX_CTL_PAD_EIM_DA14 (IOMUXC_SMUXC_PEIM_DA14)

Address: IOMUXC_SMUXC_PEIM_DA14 is 53FA_8000h base + 294h offset = 53FA_8294h

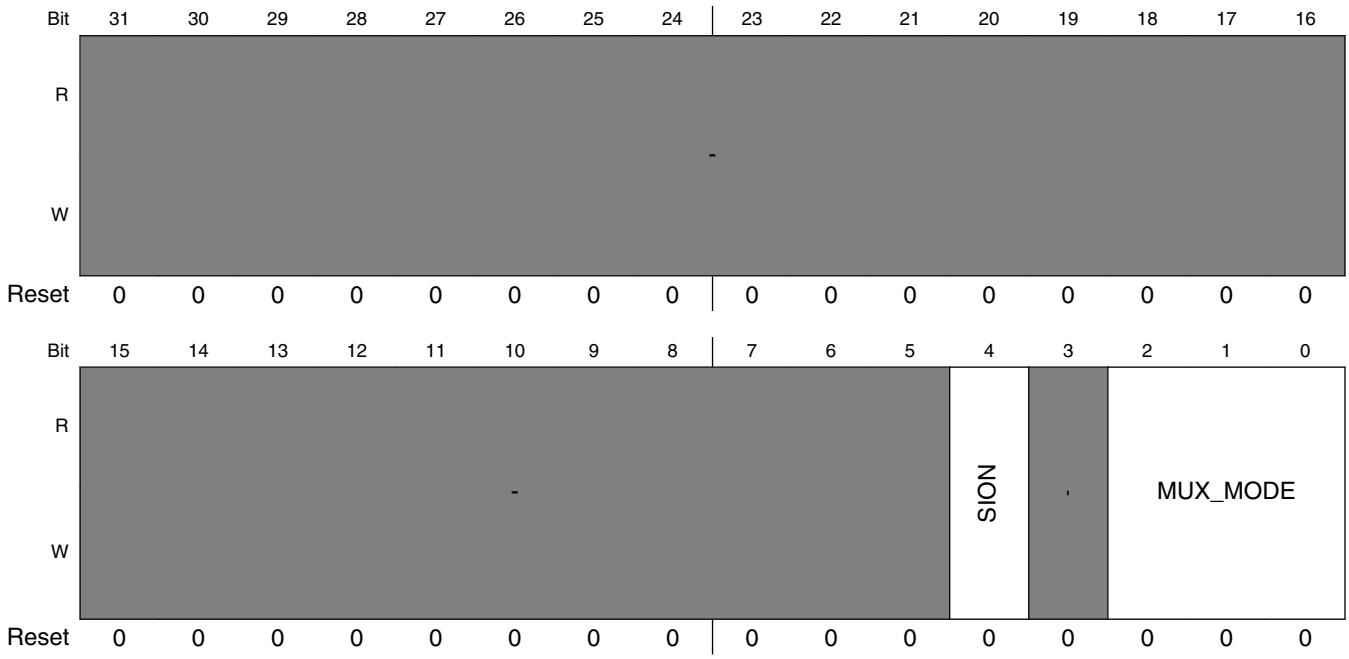


IOMUXC_SMUXC_PEIM_DA14 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA14. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EIM_DA14. **NOTE:** Pad EIM_DA14 is involved in Daisy Chain. • Config Register IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SELECT_INPUT for mode ALT2 000 Select mux mode: ALT0 mux port: A[14] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[14] of instance: gpio1. 010 Select mux mode: ALT2 mux port: READY0 of instance: rawnand. 011 Select mux mode: ALT3 mux port: SDCE[8] of instance: epdc. 110 Select mux mode: ALT6 mux port: TRACE[14] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG2[6] of instance: src.

35.4.167 SW_MUX_CTL_PAD_EIM_DA15 (IOMUXC_SMUXC_PEIM_DA15)

Address: IOMUXC_SMUXC_PEIM_DA15 is 53FA_8000h base + 298h offset = 53FA_8298h

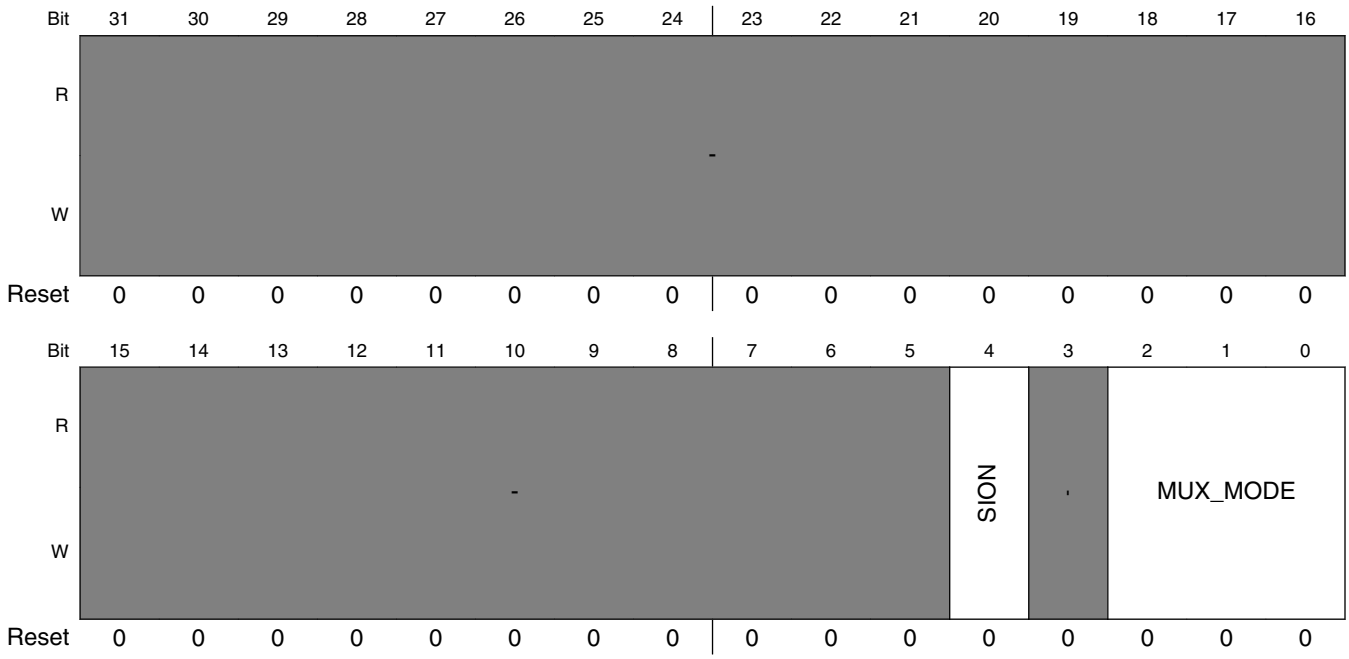


IOMUXC_SMUXC_PEIM_DA15 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_DA15. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: EIM_DA15. **NOTE:** Pad EIM_DA15 is involved in Daisy Chain. • Config Register IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SELECT_INPUT for mode ALT2. 000 Select mux mode: ALT0 mux port: A[15] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[15] of instance: gpio1. 010 Select mux mode: ALT2 mux port: DQS of instance: rawnand. 011 Select mux mode: ALT3 mux port: SDCE[9] of instance: epdc. 110 Select mux mode: ALT6 mux port: TRACE[15] of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG2[7] of instance: src.

35.4.168 SW_MUX_CTL_PAD_EIM_CS2 (IOMUXC_SMUXC_PEIM_CS2)

Address: IOMUXC_SMUXC_PEIM_CS2 is 53FA_8000h base + 29Ch offset = 53FA_829Ch

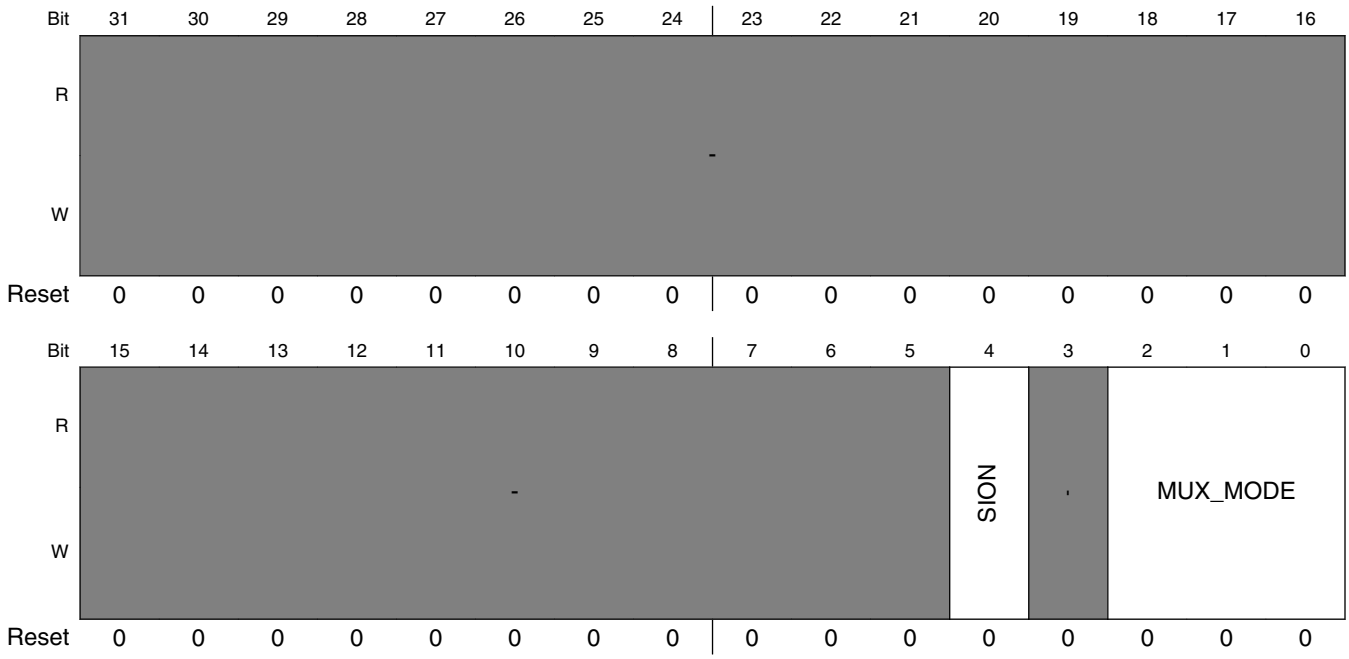


IOMUXC_SMUXC_PEIM_CS2 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_CS2. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: EIM_CS2. 000 Select mux mode: ALT0 mux port: CS[2] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[16] of instance: gpio1. 010 Select mux mode: ALT2 mux port: A[27] of instance: weimv2. 110 Select mux mode: ALT6 mux port: TRCLK of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG3[0] of instance: src.

35.4.169 SW_MUX_CTL_PAD_EIM_CS1 (IOMUXC_SMUXC_PEIM_CS1)

Address: IOMUXC_SMUXC_PEIM_CS1 is 53FA_8000h base + 2A0h offset = 53FA_82A0h

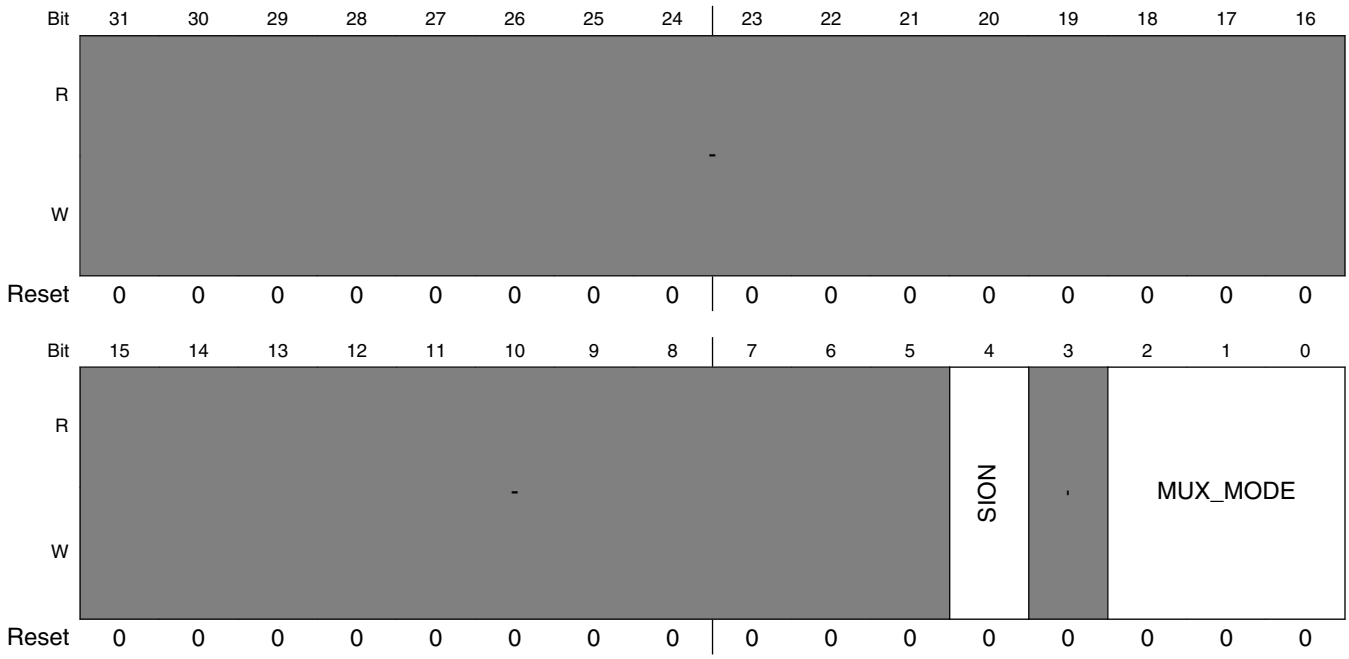


IOMUXC_SMUXC_PEIM_CS1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_CS1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: EIM_CS1. 000 Select mux mode: ALT0 mux port: CS[1] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[17] of instance: gpio1. 110 Select mux mode: ALT6 mux port: TRCTL of instance: tpiu. 111 Select mux mode: ALT7 mux port: BT_CFG3[1] of instance: src.

35.4.170 SW_MUX_CTL_PAD_EIM_CS0 (IOMUXC_SMUXC_PEIM_CS0)

Address: IOMUXC_SMUXC_PEIM_CS0 is 53FA_8000h base + 2A4h offset = 53FA_82A4h



IOMUXC_SMUXC_PEIM_CS0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_CS0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EIM_CS0. 000 Select mux mode: ALT0 mux port: CS[0] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[18] of instance: gpio1. 111 Select mux mode: ALT7 mux port: BT_CFG3[2] of instance: src.

35.4.171 SW_MUX_CTL_PAD_EIM_EB0 (IOMUXC_SMUXC_PEIM_EB0)

Address: IOMUXC_SMUXC_PEIM_EB0 is 53FA_8000h base + 2A8h offset = 53FA_82A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

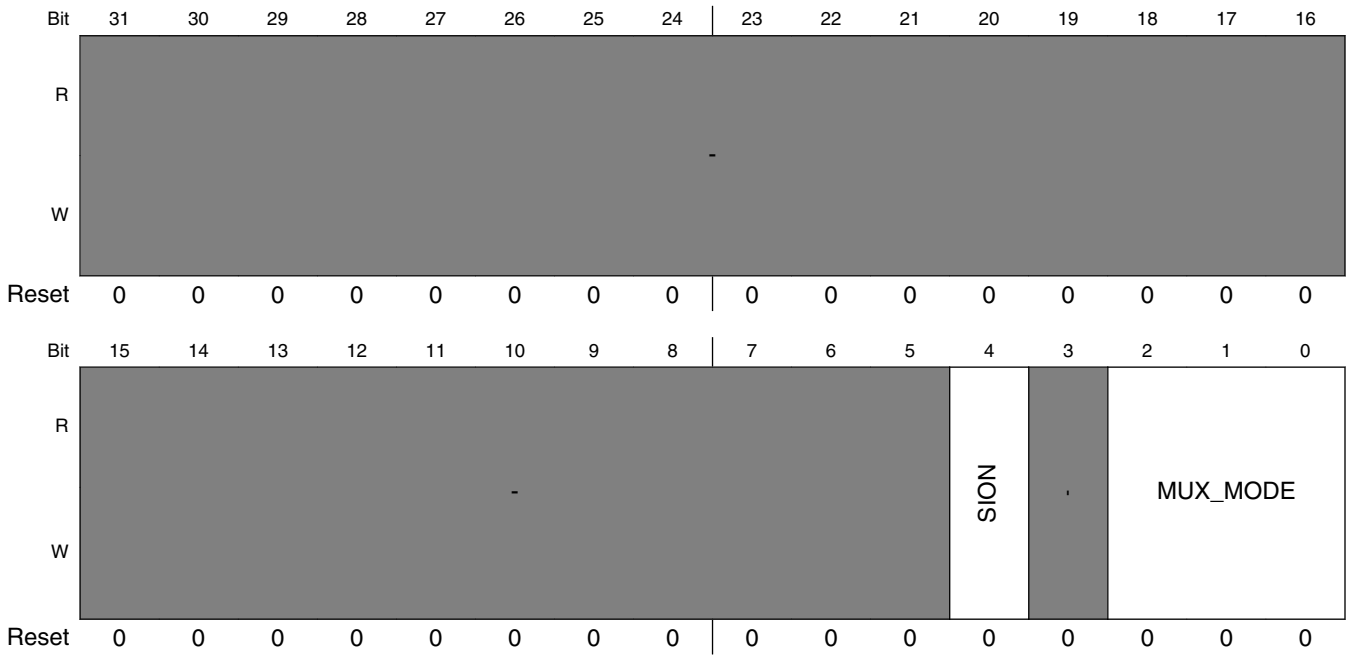
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-											SION	-	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SMUXC_PEIM_EB0 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_EB0. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EIM_EB0. 000 Select mux mode: ALT0 mux port: EB[0] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[19] of instance: gpio1. 111 Select mux mode: ALT7 mux port: BT_CFG3[3] of instance: src.

35.4.172 SW_MUX_CTL_PAD_EIM_EB1 (IOMUXC_SMUXC_PEIM_EB1)

Address: IOMUXC_SMUXC_PEIM_EB1 is 53FA_8000h base + 2ACh offset = 53FA_82ACh

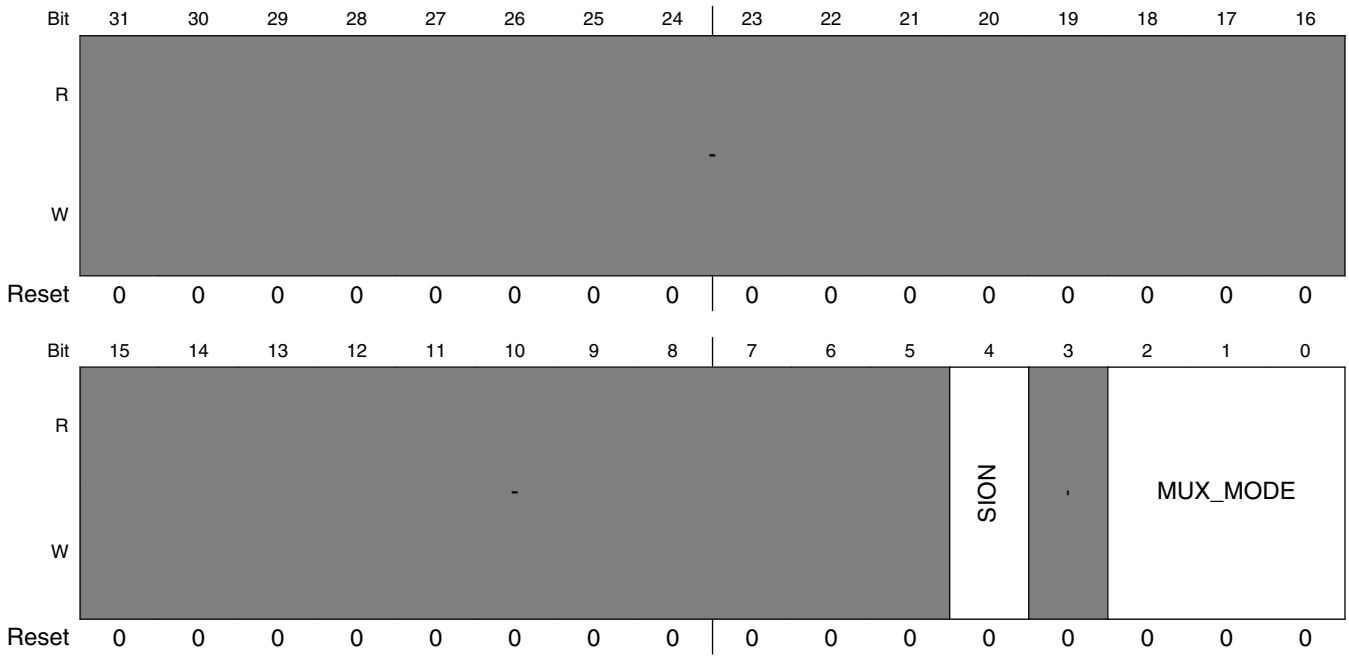


IOMUXC_SMUXC_PEIM_EB1 field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_EB1. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EIM_EB1. 000 Select mux mode: ALT0 mux port: EB[1] of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[20] of instance: gpio1. 111 Select mux mode: ALT7 mux port: BT_CFG3[4] of instance: src.

35.4.173 SW_MUX_CTL_PAD_EIM_WAIT (IOMUXC_SMUXC_PEIM_WAIT)

Address: IOMUXC_SMUXC_PEIM_WAIT is 53FA_8000h base + 2B0h offset = 53FA_82B0h

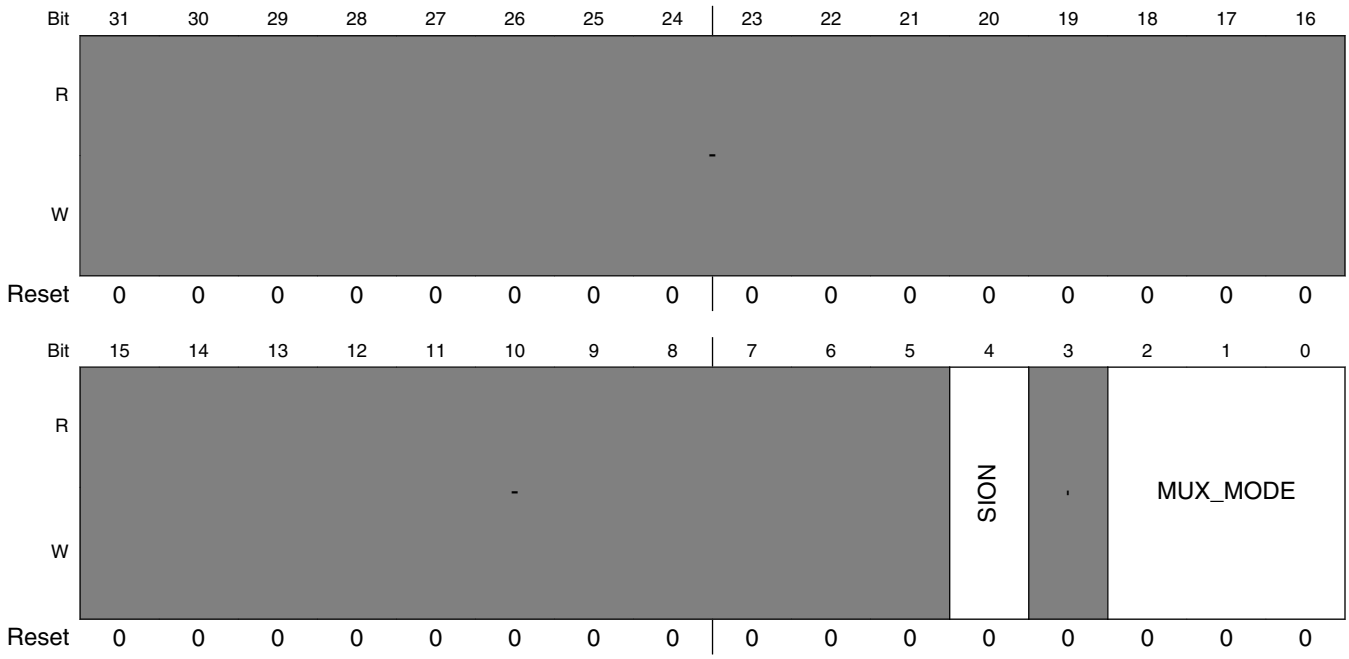


IOMUXC_SMUXC_PEIM_WAIT field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_WAIT. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: EIM_WAIT. 000 Select mux mode: ALT0 mux port: WAIT of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[21] of instance: gpio1. 010 Select mux mode: ALT2 mux port: DTACK_B of instance: weimv2. 111 Select mux mode: ALT7 mux port: BT_CFG3[5] of instance: src.

35.4.174 SW_MUX_CTL_PAD_EIM_BCLK (IOMUXC_SMUXC_PEIM_BCLK)

Address: IOMUXC_SMUXC_PEIM_BCLK is 53FA_8000h base + 2B4h offset = 53FA_82B4h

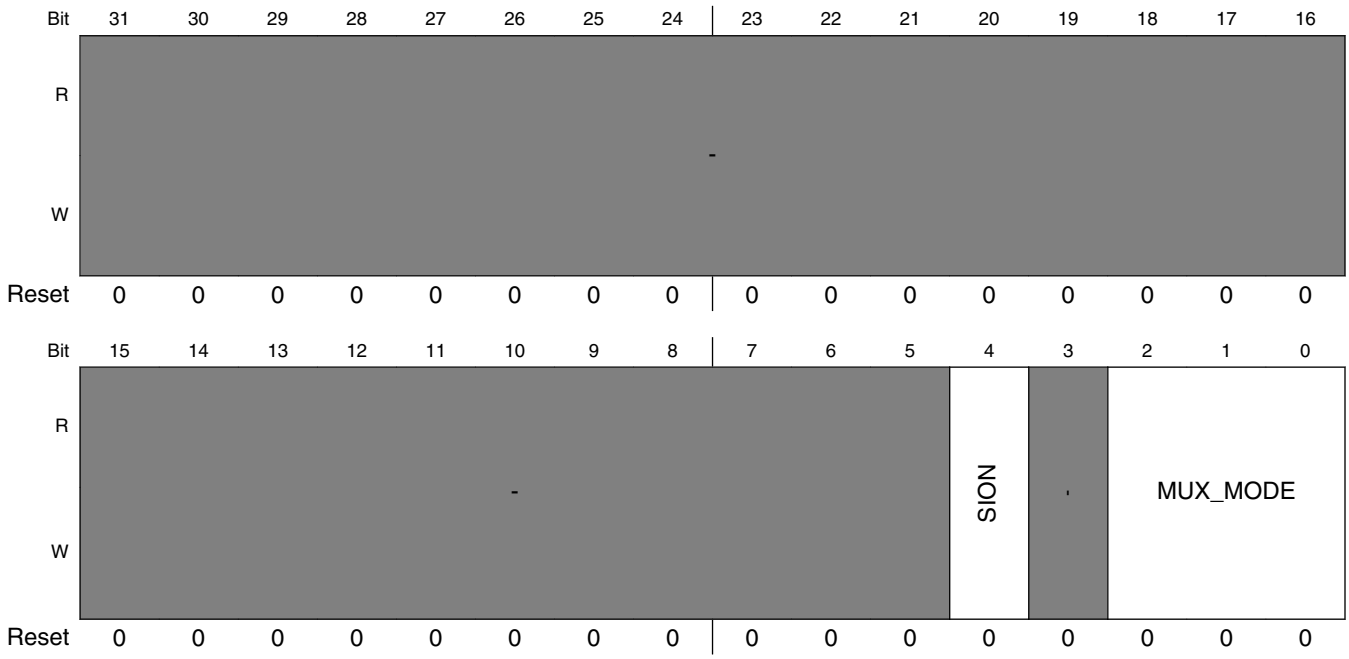


IOMUXC_SMUXC_PEIM_BCLK field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_BCLK. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EIM_BCLK. 000 Select mux mode: ALT0 mux port: BCLK of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[22] of instance: gpio1. 111 Select mux mode: ALT7 mux port: BT_CFG3[6] of instance: src.

35.4.175 SW_MUX_CTL_PAD_EIM_RDY (IOMUXC_SMUXC_PEIM_RDY)

Address: IOMUXC_SMUXC_PEIM_RDY is 53FA_8000h base + 2B8h offset = 53FA_82B8h

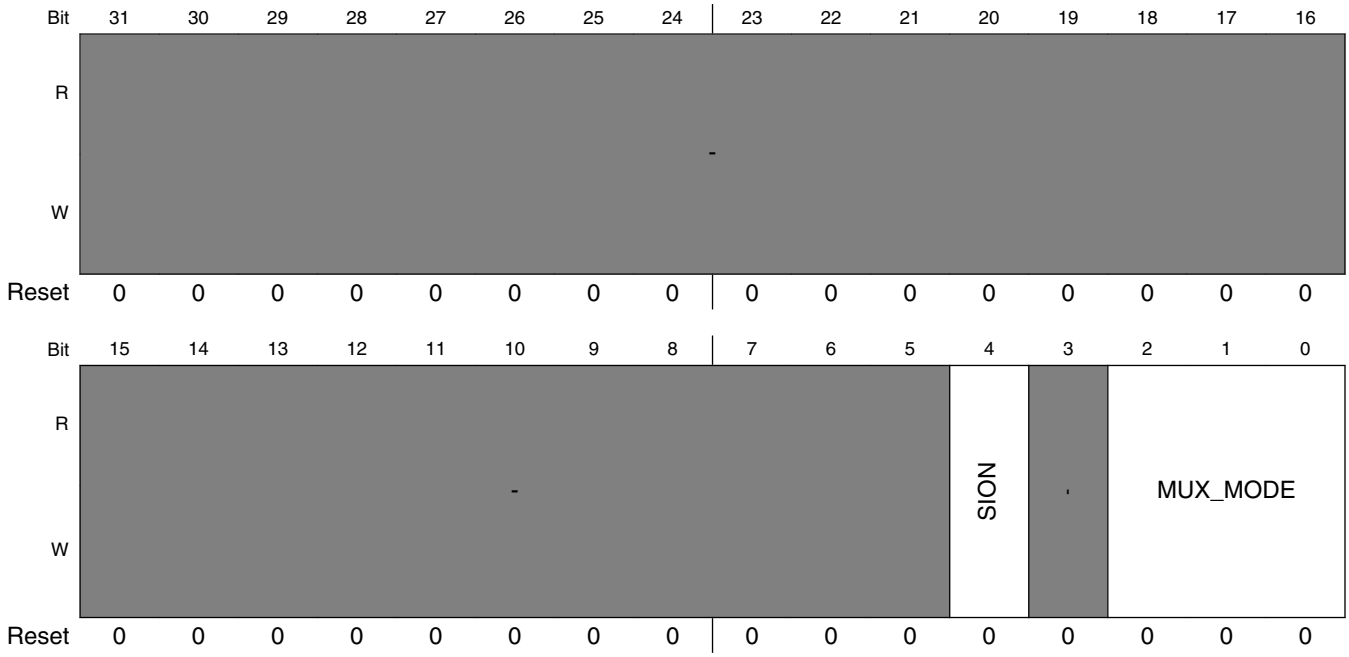


IOMUXC_SMUXC_PEIM_RDY field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_RDY. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EIM_RDY. 000 Select mux mode: ALT0 mux port: RDY of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[23] of instance: gpio1. 111 Select mux mode: ALT7 mux port: BT_CFG3[7] of instance: src.

35.4.176 SW_MUX_CTL_PAD_EIM_OE (IOMUXC_SMUXC_PEIM_OE)

Address: IOMUXC_SMUXC_PEIM_OE is 53FA_8000h base + 2BCh offset = 53FA_82BCh

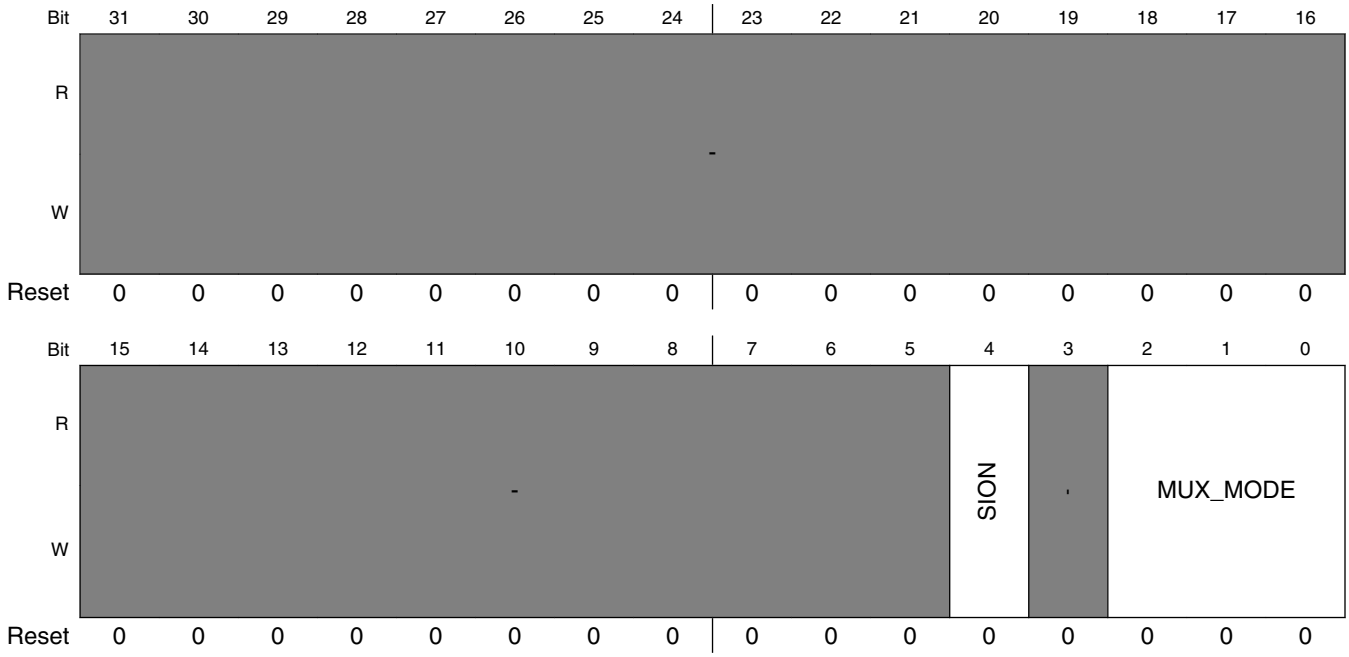


IOMUXC_SMUXC_PEIM_OE field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_OE. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EIM_OE. 000 Select mux mode: ALT0 mux port: OE of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[24] of instance: gpio1. 111 Select mux mode: ALT7 mux port: INT_BOOT of instance: src.

35.4.177 SW_MUX_CTL_PAD_EIM_RW (IOMUXC_SMUXC_PEIM_RW)

Address: IOMUXC_SMUXC_PEIM_RW is 53FA_8000h base + 2C0h offset = 53FA_82C0h

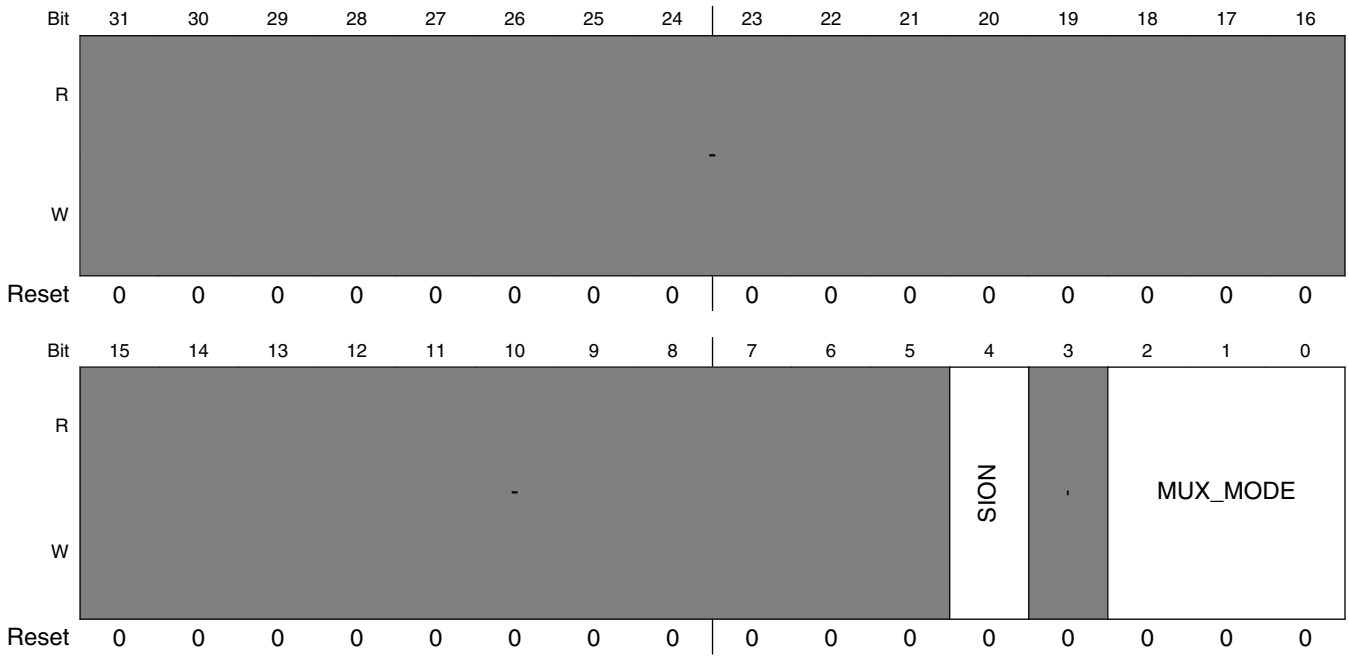


IOMUXC_SMUXC_PEIM_RW field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_RW. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EIM_RW. 000 Select mux mode: ALT0 mux port: RW of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[25] of instance: gpio1. 111 Select mux mode: ALT7 mux port: SYSTEM_RST of instance: src.

35.4.178 SW_MUX_CTL_PAD_EIM_LBA (IOMUXC_SMUXC_PEIM_LBA)

Address: IOMUXC_SMUXC_PEIM_LBA is 53FA_8000h base + 2C4h offset = 53FA_82C4h

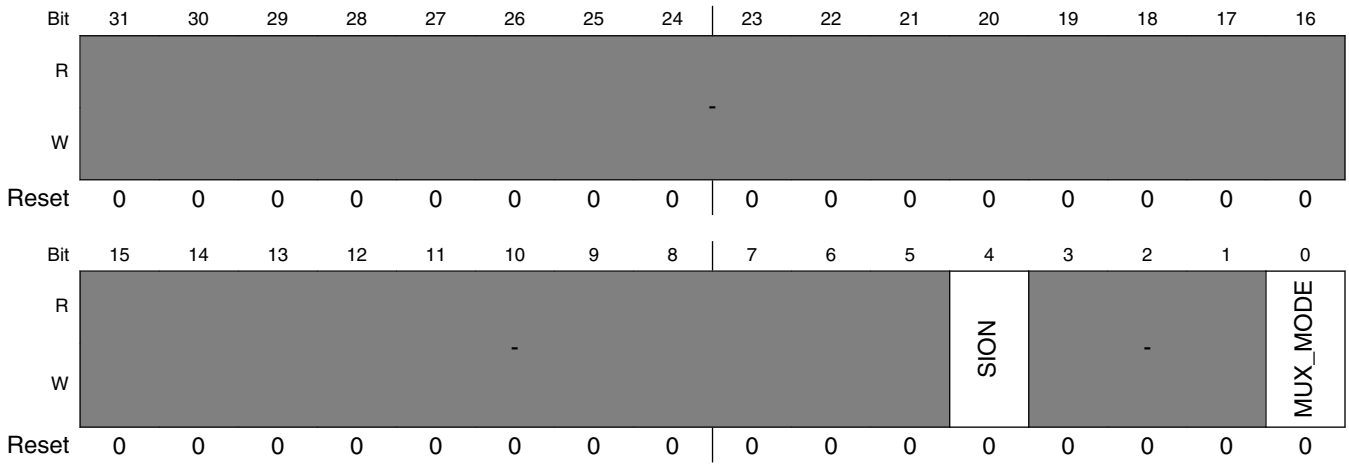


IOMUXC_SMUXC_PEIM_LBA field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_LBA. 0 Input Path is determined by functionality of the selected mux mode (regular).
3 -	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: EIM_LBA. 000 Select mux mode: ALT0 mux port: LBA of instance: weimv2. 001 Select mux mode: ALT1 mux port: GPIO[26] of instance: gpio1. 111 Select mux mode: ALT7 mux port: TESTER_ACK of instance: src.

35.4.179 SW_MUX_CTL_PAD_EIM_CRE (IOMUXC_SMUXC_PEIM_CRE)

Address: IOMUXC_SMUXC_PEIM_CRE is 53FA_8000h base + 2C8h offset = 53FA_82C8h

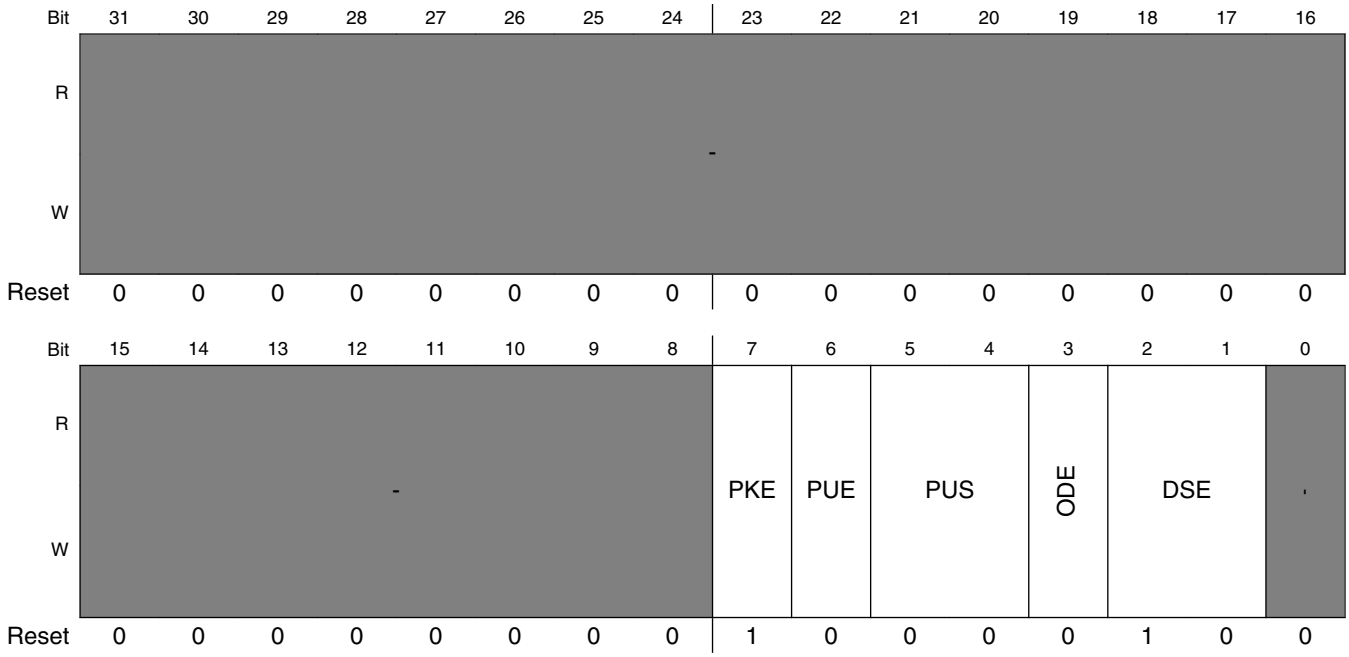


IOMUXC_SMUXC_PEIM_CRE field descriptions

Field	Description
31–5 -	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 Force input path of pad EIM_CRE. 0 Input Path is determined by functionality of the selected mux mode (regular).
3–1 -	Reserved
0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: EIM_CRE. 00 Select mux mode: ALT0 mux port: CRE of instance: weimv2. 01 Select mux mode: ALT1 mux port: GPIO[27] of instance: gpio1.

35.4.180 SW_PAD_CTL_PAD_KEY_COL0 (IOMUXC_SPADC_PKC0)

Address: IOMUXC_SPADC_PKC0 is 53FA_8000h base + 2CCh offset = 53FA_82CCh



IOMUXC_SPADC_PKC0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KEY_COL0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KEY_COL0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KEY_COL0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PKC0 field descriptions (continued)

Field	Description
	Select one out of next values for pad: KEY_COL0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: KEY_COL0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.181 SW_PAD_CTL_PAD_KEY_ROW0 (IOMUXC_SPADC_PKR0)

Address: IOMUXC_SPADC_PKR0 is 53FA_8000h base + 2D0h offset = 53FA_82D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PKR0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KEY_ROW0.

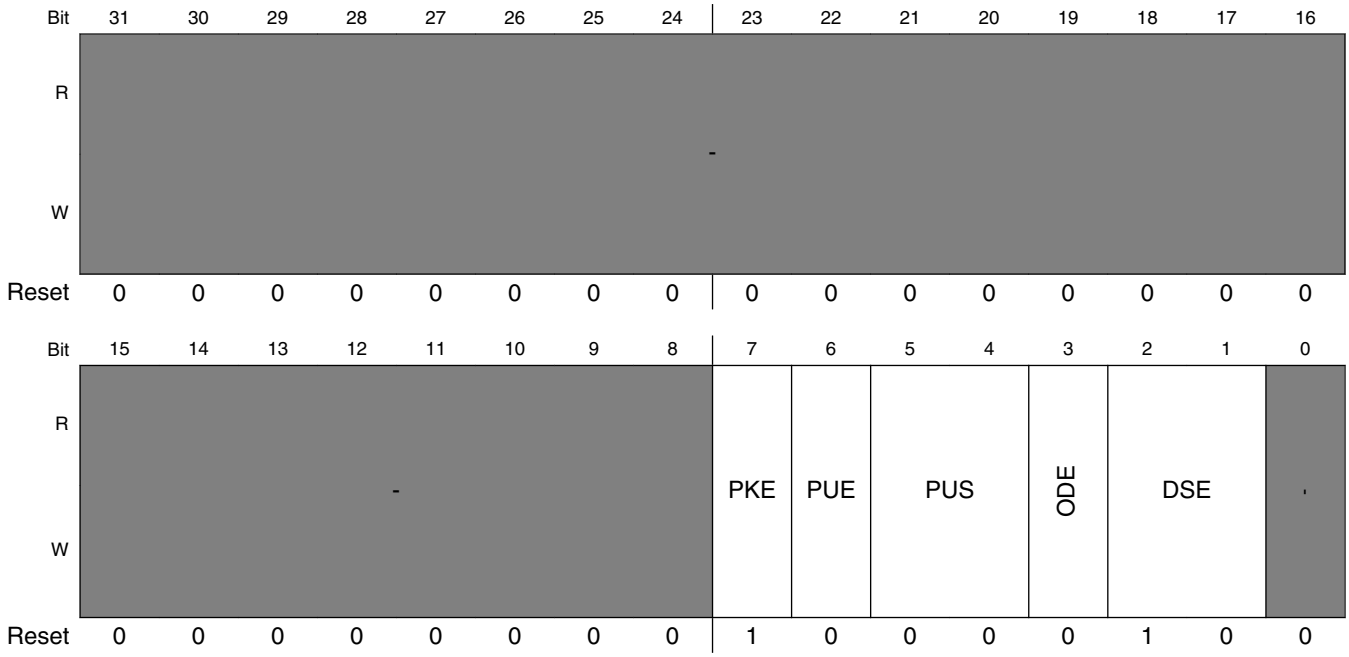
Table continues on the next page...

IOMUXC_SPADC_PKR0 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KEY_ROW0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KEY_ROW0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: KEY_ROW0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: KEY_ROW0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.182 SW_PAD_CTL_PAD_KEY_COL1 (IOMUXC_SPADC_PKC1)

Address: IOMUXC_SPADC_PKC1 is 53FA_8000h base + 2D4h offset = 53FA_82D4h



IOMUXC_SPADC_PKC1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KEY_COL1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KEY_COL1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KEY_COL1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PKC1 field descriptions (continued)

Field	Description
	Select one out of next values for pad: KEY_COL1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: KEY_COL1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.183 SW_PAD_CTL_PAD_KEY_ROW1 (IOMUXC_SPADC_PKR1)

Address: IOMUXC_SPADC_PKR1 is 53FA_8000h base + 2D8h offset = 53FA_82D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PKR1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KEY_ROW1.

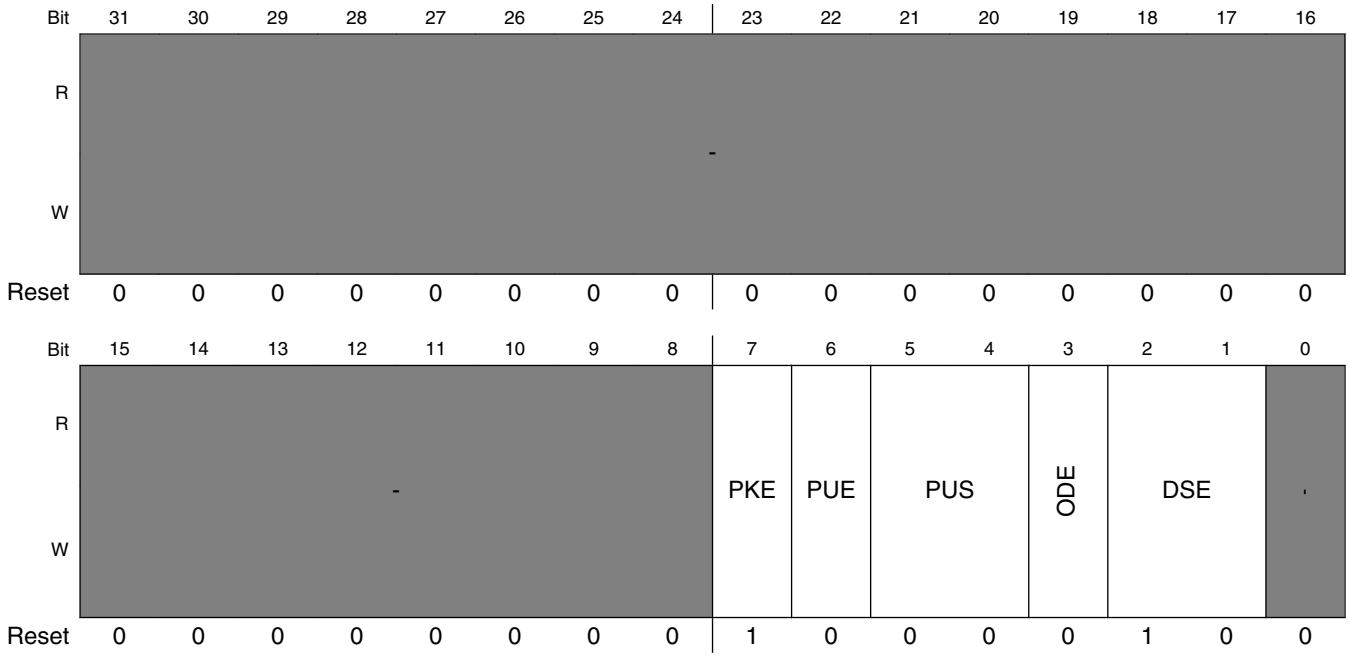
Table continues on the next page...

IOMUXC_SPADC_PKR1 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KEY_ROW1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KEY_ROW1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: KEY_ROW1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: KEY_ROW1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.184 SW_PAD_CTL_PAD_KEY_COL2 (IOMUXC_SPADC_PKC2)

Address: IOMUXC_SPADC_PKC2 is 53FA_8000h base + 2DCh offset = 53FA_82DCh



IOMUXC_SPADC_PKC2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KEY_COL2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KEY_COL2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KEY_COL2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

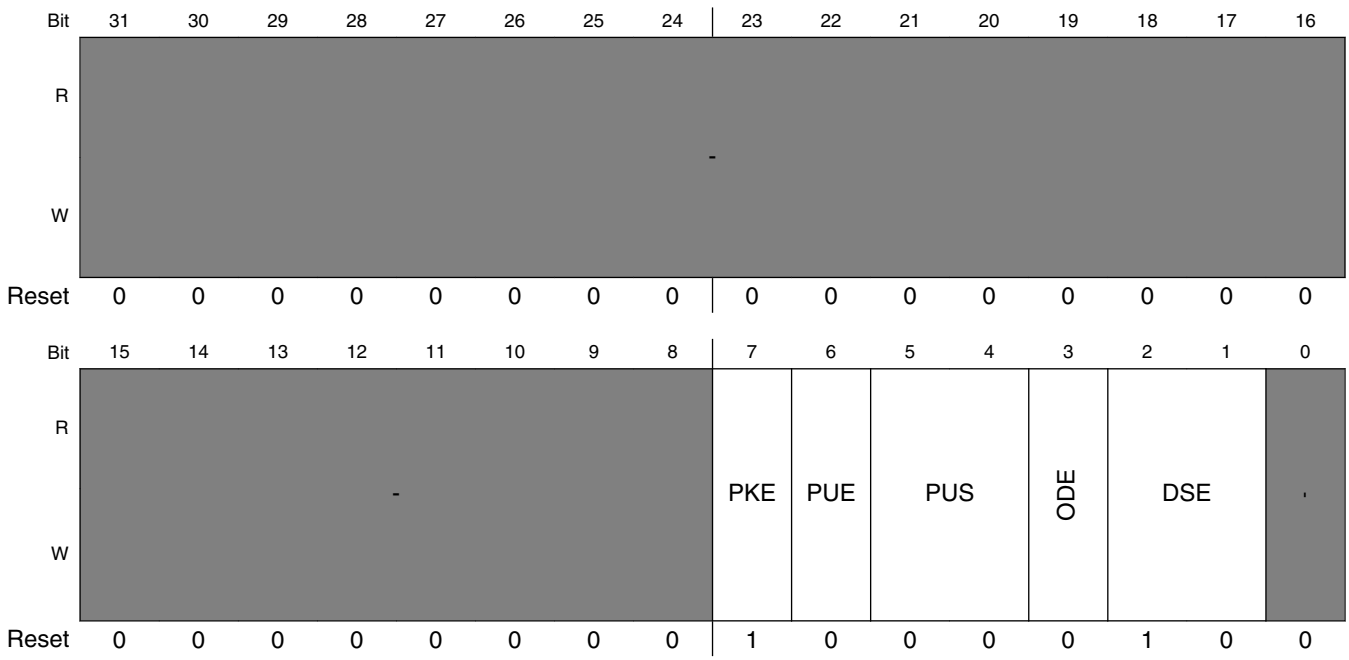
Table continues on the next page...

IOMUXC_SPADC_PKC2 field descriptions (continued)

Field	Description
	Select one out of next values for pad: KEY_COL2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: KEY_COL2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.185 SW_PAD_CTL_PAD_KEY_ROW2 (IOMUXC_SPADC_PKR2)

Address: IOMUXC_SPADC_PKR2 is 53FA_8000h base + 2E0h offset = 53FA_82E0h



IOMUXC_SPADC_PKR2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KEY_ROW2.

Table continues on the next page...

IOMUXC_SPADC_PKR2 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KEY_ROW2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KEY_ROW2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: KEY_ROW2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: KEY_ROW2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.186 SW_PAD_CTL_PAD_KEY_COL3 (IOMUXC_SPADC_PKC3)

Address: IOMUXC_SPADC_PKC3 is 53FA_8000h base + 2E4h offset = 53FA_82E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PKC3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KEY_COL3. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KEY_COL3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KEY_COL3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PKC3 field descriptions (continued)

Field	Description
	Select one out of next values for pad: KEY_COL3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: KEY_COL3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.187 SW_PAD_CTL_PAD_KEY_ROW3 (IOMUXC_SPADC_PKR3)

Address: IOMUXC_SPADC_PKR3 is 53FA_8000h base + 2E8h offset = 53FA_82E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PKR3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KEY_ROW3.

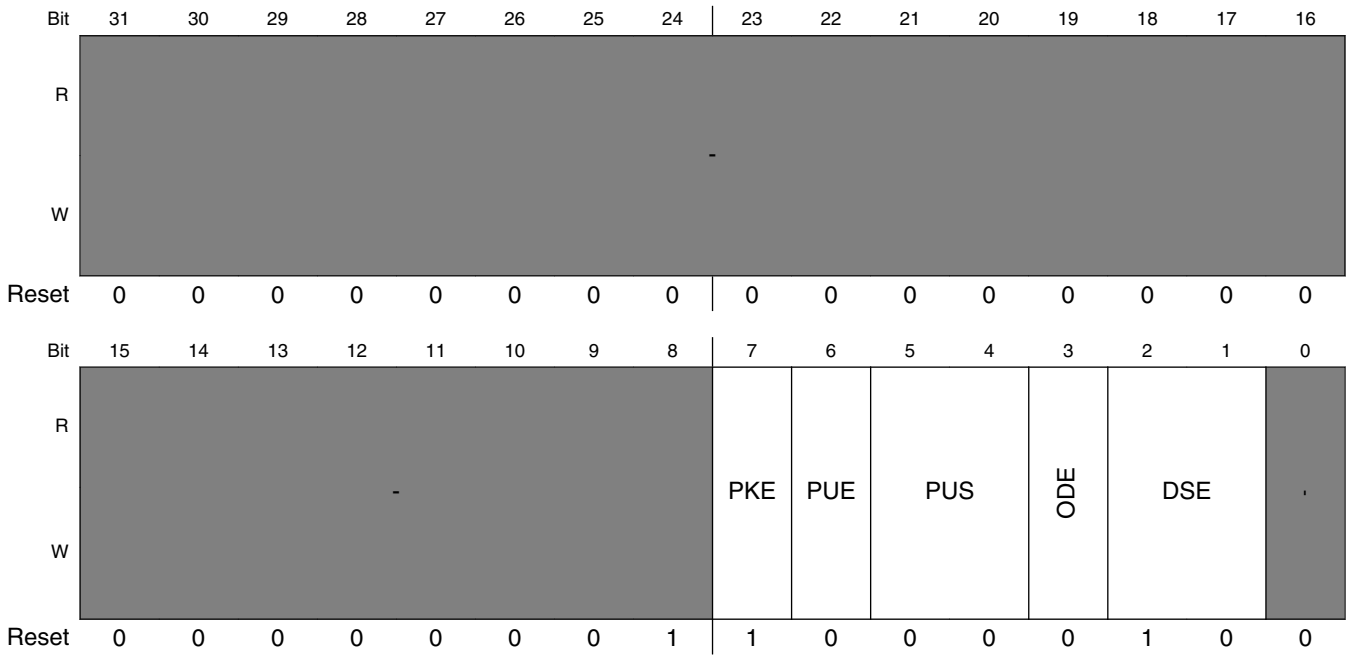
Table continues on the next page...

IOMUXC_SPADC_PKR3 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KEY_ROW3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KEY_ROW3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: KEY_ROW3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: KEY_ROW3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.188 SW_PAD_CTL_PAD_I2C1_SCL (IOMUXC_SPADC_PI2C1_SCL)

Address: IOMUXC_SPADC_PI2C1_SCL is 53FA_8000h base + 2ECh offset = 53FA_82ECh



IOMUXC_SPADC_PI2C1_SCL field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: I2C1_SCL. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: I2C1_SCL. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: I2C1_SCL. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PI2C1_SCL field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: I2C1_SCL. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: I2C1_SCL. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.189 SW_PAD_CTL_PAD_I2C1_SDA (IOMUXC_SPADC_PI2C1_SDA)

Address: IOMUXC_SPADC_PI2C1_SDA is 53FA_8000h base + 2F0h offset = 53FA_82F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PI2C1_SDA field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: I2C1_SDA. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: I2C1_SDA. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: I2C1_SDA. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: I2C1_SDA. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: I2C1_SDA. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.190 SW_PAD_CTL_PAD_I2C2_SCL (IOMUXC_SPADC_PI2C2_SCL)

Address: IOMUXC_SPADC_PI2C2_SCL is 53FA_8000h base + 2F4h offset = 53FA_82F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PI2C2_SCL field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: I2C2_SCL. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: I2C2_SCL. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: I2C2_SCL. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

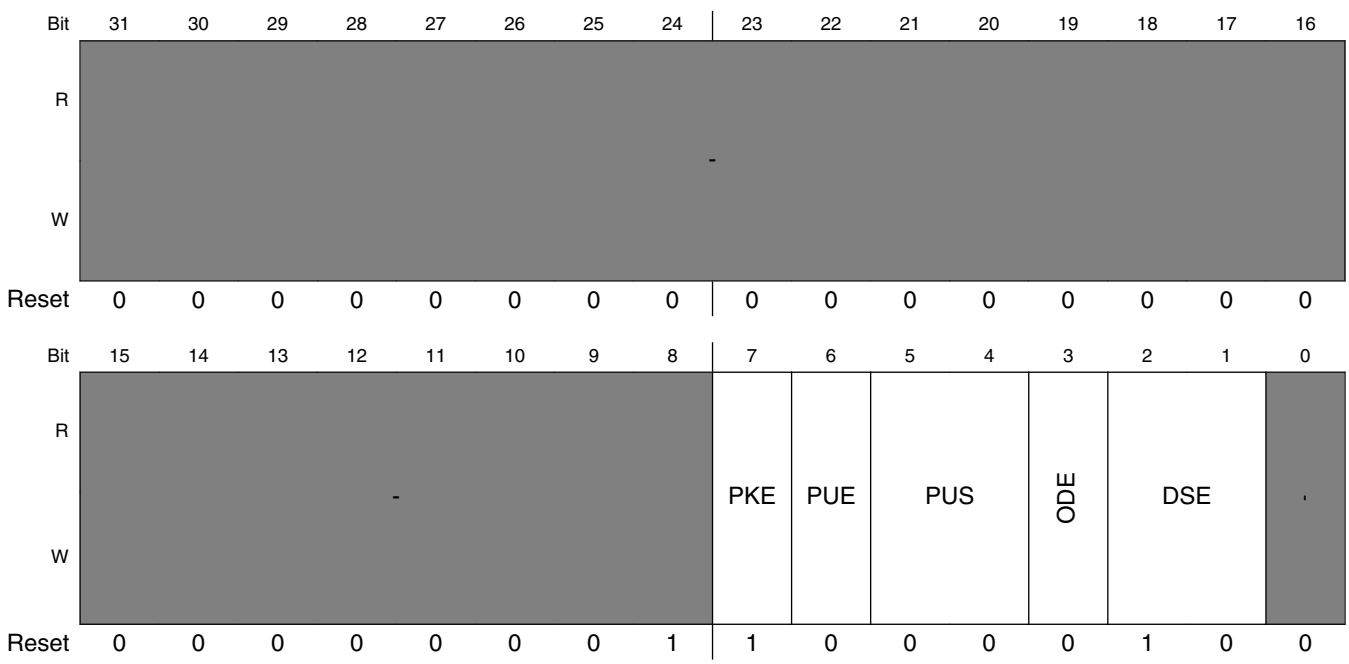
Table continues on the next page...

IOMUXC_SPADC_PI2C2_SCL field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: I2C2_SCL. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: I2C2_SCL. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.191 SW_PAD_CTL_PAD_I2C2_SDA (IOMUXC_SPADC_PI2C2_SDA)

Address: IOMUXC_SPADC_PI2C2_SDA is 53FA_8000h base + 2F8h offset = 53FA_82F8h

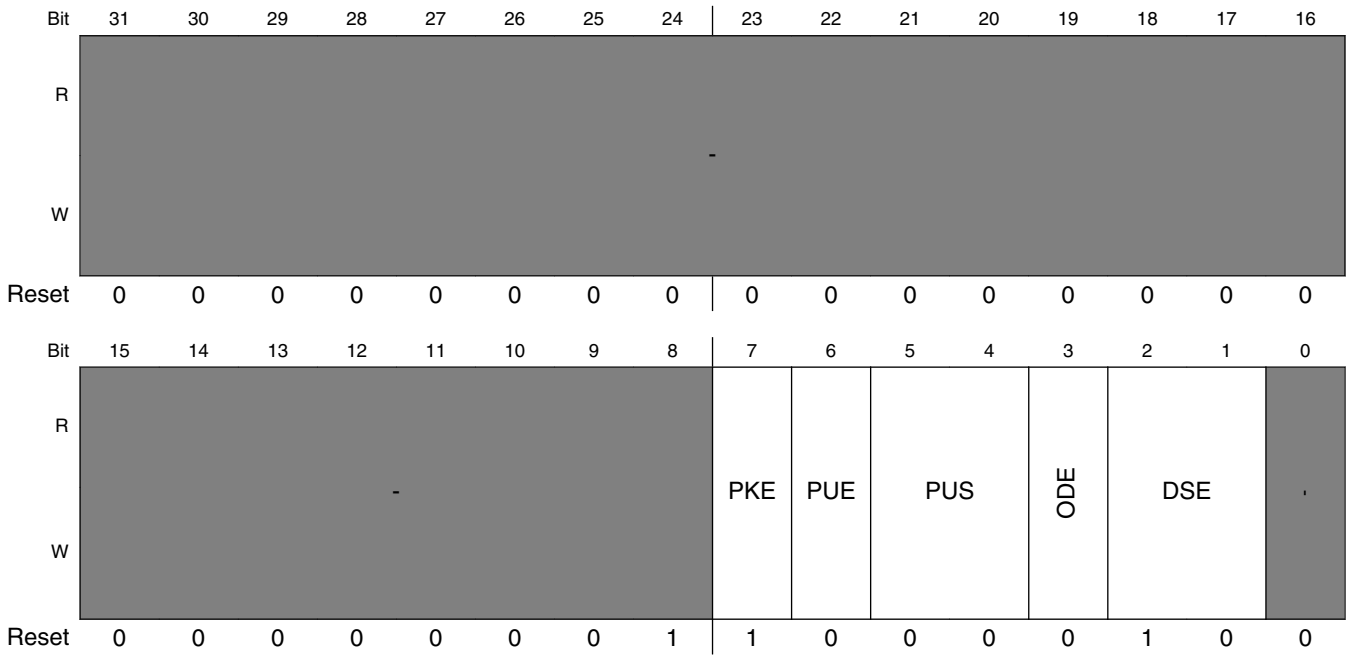


IOMUXC_SPADC_PI2C2_SDA field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: I2C2_SDA. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: I2C2_SDA. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: I2C2_SDA. 00 100 kΩ Pull Down 01 47 kΩPull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: I2C2_SDA. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: I2C2_SDA. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.192 SW_PAD_CTL_PAD_I2C3_SCL (IOMUXC_SPADC_PI2C3_SCL)

Address: IOMUXC_SPADC_PI2C3_SCL is 53FA_8000h base + 2FCh offset = 53FA_82FCh



IOMUXC_SPADC_PI2C3_SCL field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: I2C3_SCL. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: I2C3_SCL. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: I2C3_SCL. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PI2C3_SCL field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: I2C3_SCL. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: I2C3_SCL. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.193 SW_PAD_CTL_PAD_I2C3_SDA (IOMUXC_SPADC_PI2C3_SDA)

Address: IOMUXC_SPADC_PI2C3_SDA is 53FA_8000h base + 300h offset = 53FA_8300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

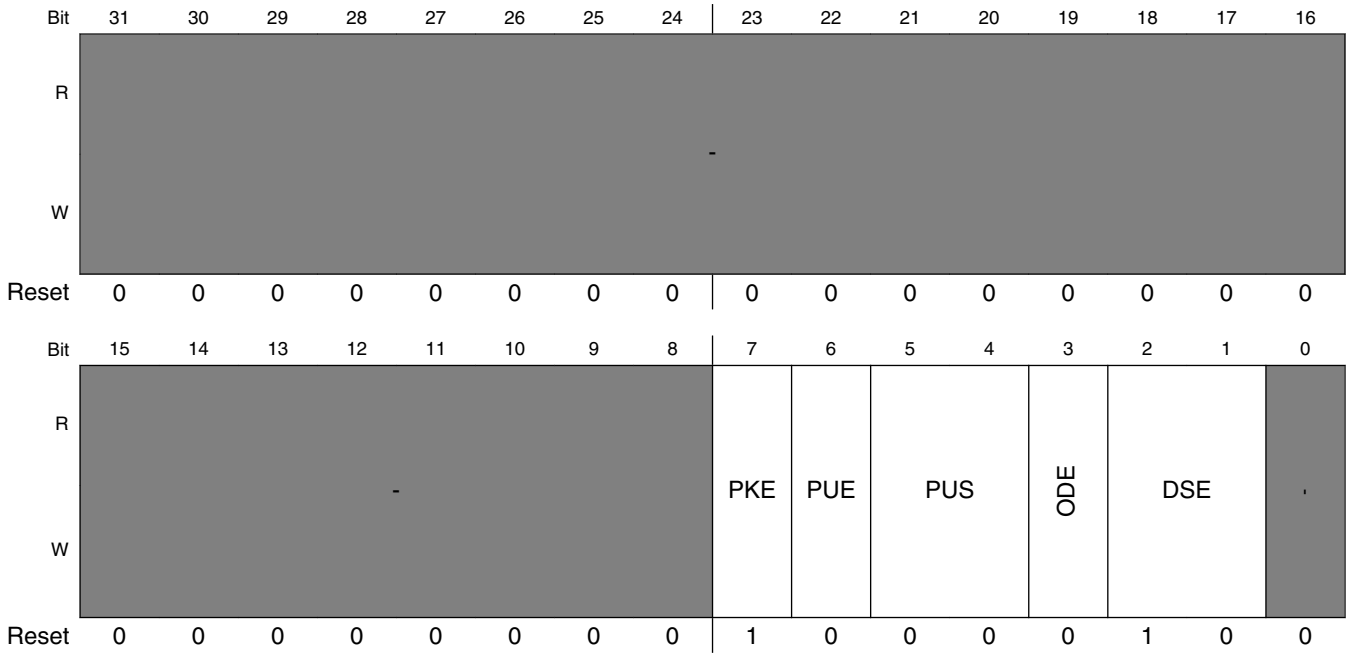
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PI2C3_SDA field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: I2C3_SDA. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: I2C3_SDA. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: I2C3_SDA. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: I2C3_SDA. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: I2C3_SDA. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.194 SW_PAD_CTL_PAD_PWM1 (IOMUXC_SPADC_PPWM1)

Address: IOMUXC_SPADC_PPWM1 is 53FA_8000h base + 304h offset = 53FA_8304h



IOMUXC_SPADC_PPWM1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: PWM1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: PWM1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: PWM1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PPWM1 field descriptions (continued)

Field	Description
	Select one out of next values for pad: PWM1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: PWM1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.195 SW_PAD_CTL_PAD_PWM2 (IOMUXC_SPADC_PPWM2)

Address: IOMUXC_SPADC_PPWM2 is 53FA_8000h base + 308h offset = 53FA_8308h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PPWM2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: PWM2.

Table continues on the next page...

IOMUXC_SPADC_PPWM2 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: PWM2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: PWM2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩPull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: PWM2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: PWM2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.196 SW_PAD_CTL_PAD_OWIRE (IOMUXC_SPADC_POWIRE)

Address: IOMUXC_SPADC_POWIRE is 53FA_8000h base + 30Ch offset = 53FA_830Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_POWIRE field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: OWIRE. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: OWIRE. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: OWIRE. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_POWIRE field descriptions (continued)

Field	Description
	Select one out of next values for pad: OWIRE. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: OWIRE. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.197 SW_PAD_CTL_PAD_EPITO (IOMUXC_SPADC_PEPITO)

Address: IOMUXC_SPADC_PEPITO is 53FA_8000h base + 310h offset = 53FA_8310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPITO field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPITO.

Table continues on the next page...

IOMUXC_SPADC_PEPITO field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPITO. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPITO. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPITO. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPITO. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.198 SW_PAD_CTL_PAD_WDOG (IOMUXC_SPADC_PWDOG)

Address: IOMUXC_SPADC_PWDOG is 53FA_8000h base + 314h offset = 53FA_8314h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PWDOG field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: WDOG. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: WDOG. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: WDOG. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PWDOG field descriptions (continued)

Field	Description
	Select one out of next values for pad: WDOG. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: WDOG. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.199 SW_PAD_CTL_PAD_SSI_TXFS (IOMUXC_SPADC_PSSI_TXFS)

Address: IOMUXC_SPADC_PSSI_TXFS is 53FA_8000h base + 318h offset = 53FA_8318h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSSI_TXFS field descriptions

Field	Description
31–8 -	Reserved

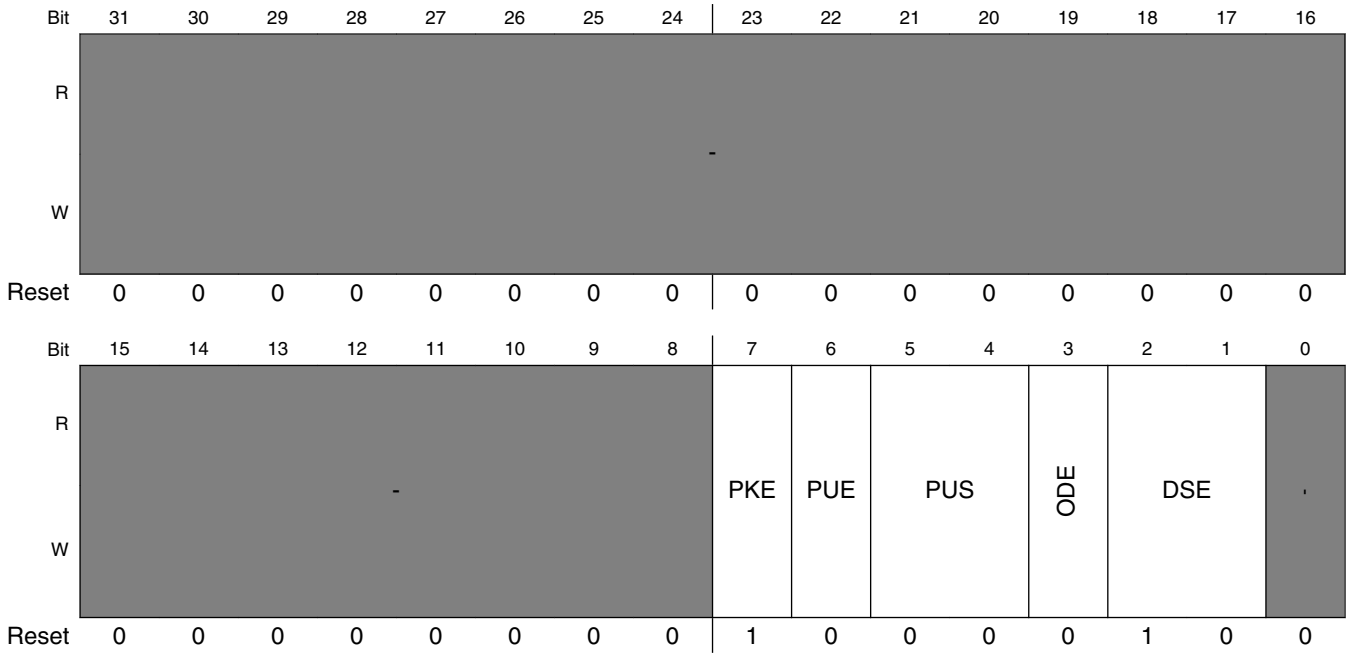
Table continues on the next page...

IOMUXC_SPADC_PSSI_TXFS field descriptions (continued)

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SSI_TXFS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SSI_TXFS. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SSI_TXFS. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SSI_TXFS. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SSI_TXFS. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.200 SW_PAD_CTL_PAD_SSI_TXC (IOMUXC_SPADC_PSSI_TXC)

Address: IOMUXC_SPADC_PSSI_TXC is 53FA_8000h base + 31Ch offset = 53FA_831Ch



IOMUXC_SPADC_PSSI_TXC field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SSI_TXC. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SSI_TXC. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SSI_TXC. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSSI_TXC field descriptions (continued)

Field	Description
	Select one out of next values for pad: SSI_TXC. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SSI_TXC. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.201 SW_PAD_CTL_PAD_SSI_TXD (IOMUXC_SPADC_PSSI_TXD)

Address: IOMUXC_SPADC_PSSI_TXD is 53FA_8000h base + 320h offset = 53FA_8320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSSI_TXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SSI_TXD.

Table continues on the next page...

IOMUXC_SPADC_PSSI_TXD field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SSI_TXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SSI_TXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SSI_TXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SSI_TXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.202 SW_PAD_CTL_PAD_SSI_RXD (IOMUXC_SPADC_PSSI_RXD)

Address: IOMUXC_SPADC_PSSI_RXD is 53FA_8000h base + 324h offset = 53FA_8324h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSSI_RXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SSI_RXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SSI_RXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SSI_RXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PSSI_RXD field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: SSI_RXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SSI_RXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.203 SW_PAD_CTL_PAD_SSI_RXFS (IOMUXC_SPADC_PSSI_RXFS)

Address: IOMUXC_SPADC_PSSI_RXFS is 53FA_8000h base + 328h offset = 53FA_8328h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSSI_RXFS field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SSI_RXFS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SSI_RXFS. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SSI_RXFS. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SSI_RXFS. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SSI_RXFS. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.204 SW_PAD_CTL_PAD_SSI_RXC (IOMUXC_SPADC_PSSI_RXC)

Address: IOMUXC_SPADC_PSSI_RXC is 53FA_8000h base + 32Ch offset = 53FA_832Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSSI_RXC field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SSI_RXC. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SSI_RXC. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SSI_RXC. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PSSI_RXC field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: SSI_RXC. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SSI_RXC. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.205 SW_PAD_CTL_PAD_UART1_TXD (IOMUXC_SPADC_PUART1_TXD)

Address: IOMUXC_SPADC_PUART1_TXD is 53FA_8000h base + 330h offset = 53FA_8330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PUART1_TXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_TXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_TXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_TXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART1_TXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART1_TXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.206 SW_PAD_CTL_PAD_UART1_RXD (IOMUXC_SPADC_PUART1_RXD)

Address: IOMUXC_SPADC_PUART1_RXD is 53FA_8000h base + 334h offset = 53FA_8334h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PUART1_RXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_RXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_RXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_RXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PUART1_RXD field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART1_RXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART1_RXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.207 SW_PAD_CTL_PAD_UART1_CTS (IOMUXC_SPADC_PUART1_CTS)

Address: IOMUXC_SPADC_PUART1_CTS is 53FA_8000h base + 338h offset = 53FA_8338h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

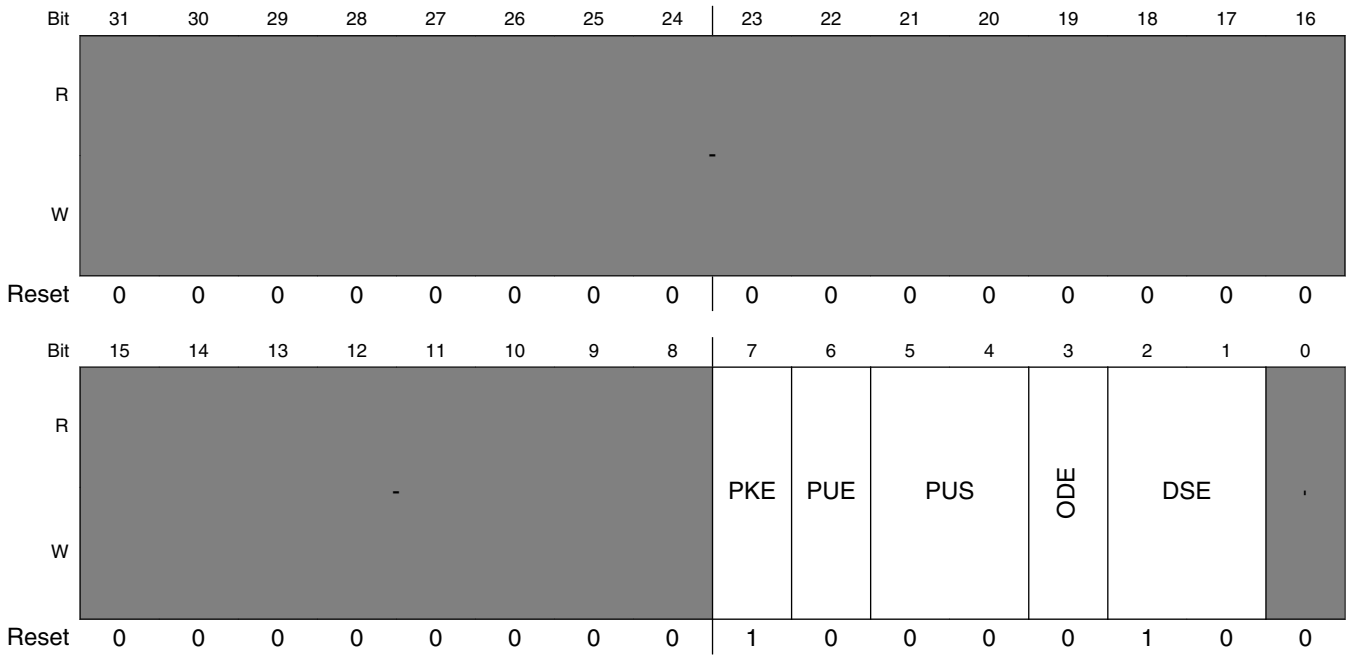
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PUART1_CTS field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_CTS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_CTS. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_CTS. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART1_CTS. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART1_CTS. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.208 SW_PAD_CTL_PAD_UART1_RTS (IOMUXC_SPADC_PUART1_RTS)

Address: IOMUXC_SPADC_PUART1_RTS is 53FA_8000h base + 33Ch offset = 53FA_833Ch



IOMUXC_SPADC_PUART1_RTS field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_RTS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_RTS. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_RTS. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PUART1_RTS field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART1_RTS. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART1_RTS. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.209 SW_PAD_CTL_PAD_UART2_TXD (IOMUXC_SPADC_PUART2_TXD)

Address: IOMUXC_SPADC_PUART2_TXD is 53FA_8000h base + 340h offset = 53FA_8340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

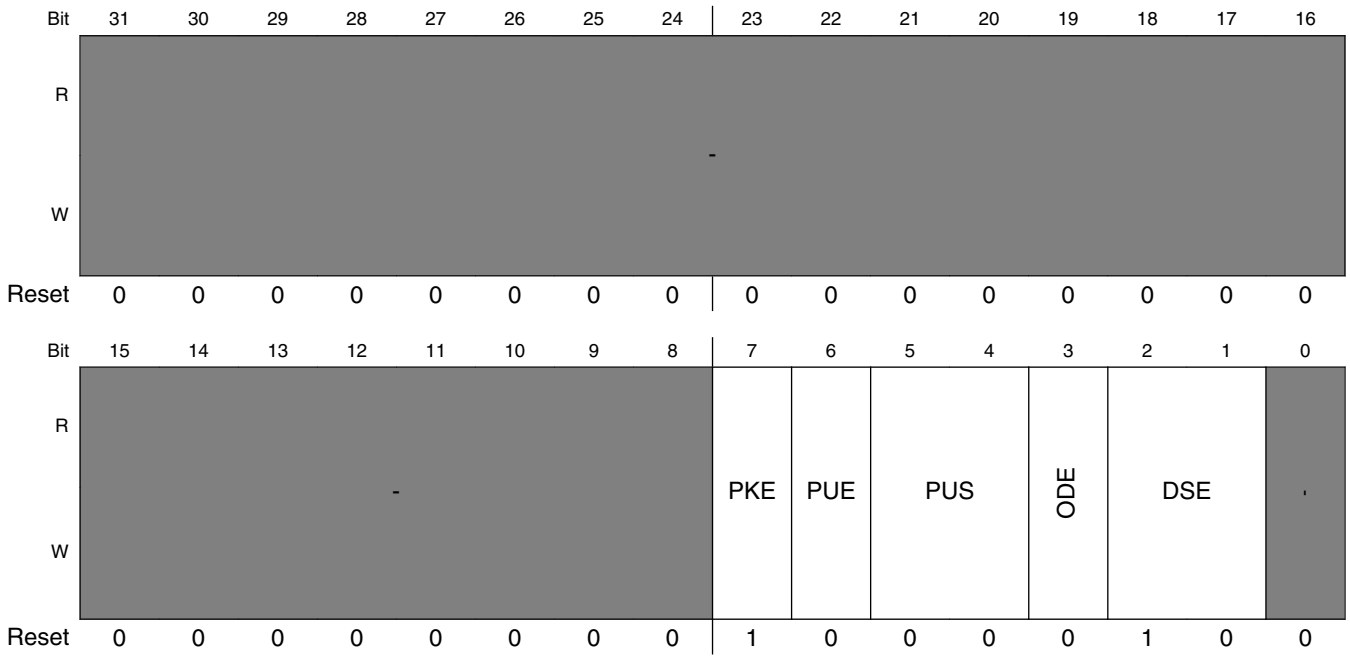
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PUART2_TXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_TXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_TXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_TXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART2_TXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART2_TXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.210 SW_PAD_CTL_PAD_UART2_RXD (IOMUXC_SPADC_PUART2_RXD)

Address: IOMUXC_SPADC_PUART2_RXD is 53FA_8000h base + 344h offset = 53FA_8344h



IOMUXC_SPADC_PUART2_RXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_RXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_RXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_RXD. 00 10 0 kΩ Pull Down 01 47kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PUART2_RXD field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART2_RXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART2_RXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.211 SW_PAD_CTL_PAD_UART2_CTS (IOMUXC_SPADC_PUART2_CTS)

Address: IOMUXC_SPADC_PUART2_CTS is 53FA_8000h base + 348h offset = 53FA_8348h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

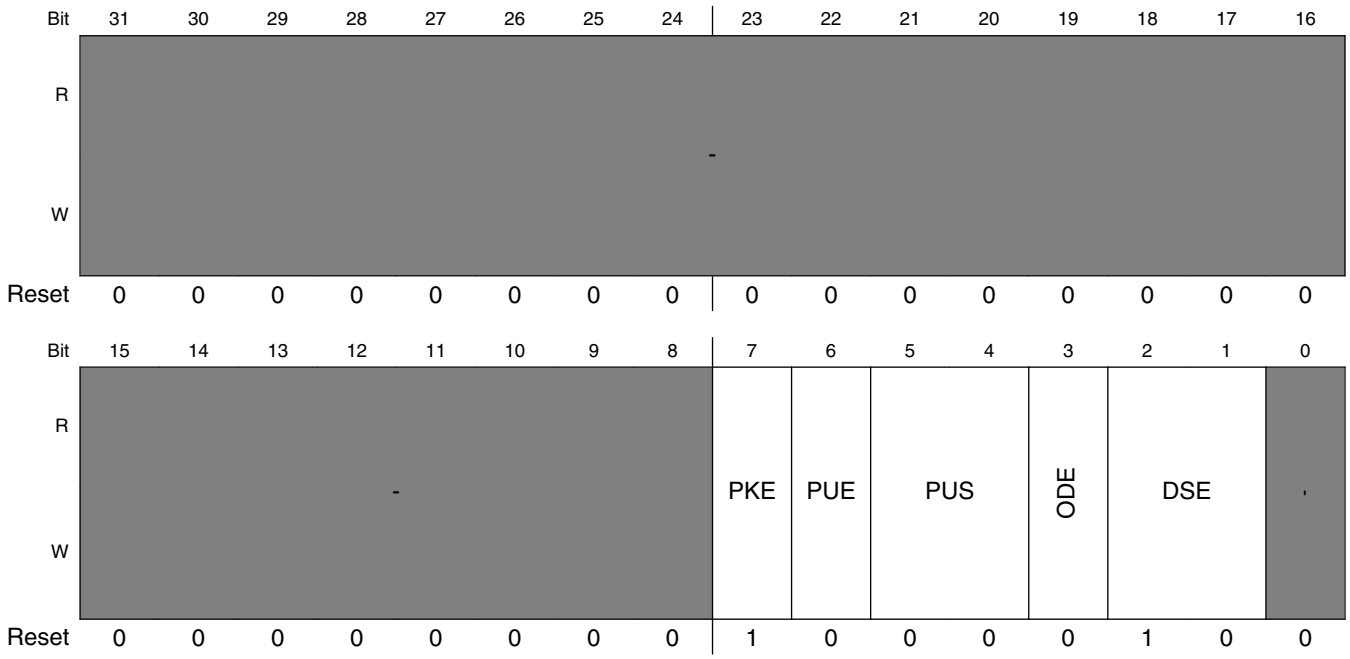
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PUART2_CTS field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_CTS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_CTS. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_CTS. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART2_CTS. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART2_CTS. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.212 SW_PAD_CTL_PAD_UART2_RTS (IOMUXC_SPADC_PUART2_RTS)

Address: IOMUXC_SPADC_PUART2_RTS is 53FA_8000h base + 34Ch offset = 53FA_834Ch



IOMUXC_SPADC_PUART2_RTS field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_RTS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_RTS. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_RTS. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PUART2_RTS field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART2_RTS. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART2_RTS. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.213 SW_PAD_CTL_PAD_UART3_TXD (IOMUXC_SPADC_PUART3_TXD)

Address: IOMUXC_SPADC_PUART3_TXD is 53FA_8000h base + 350h offset = 53FA_8350h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

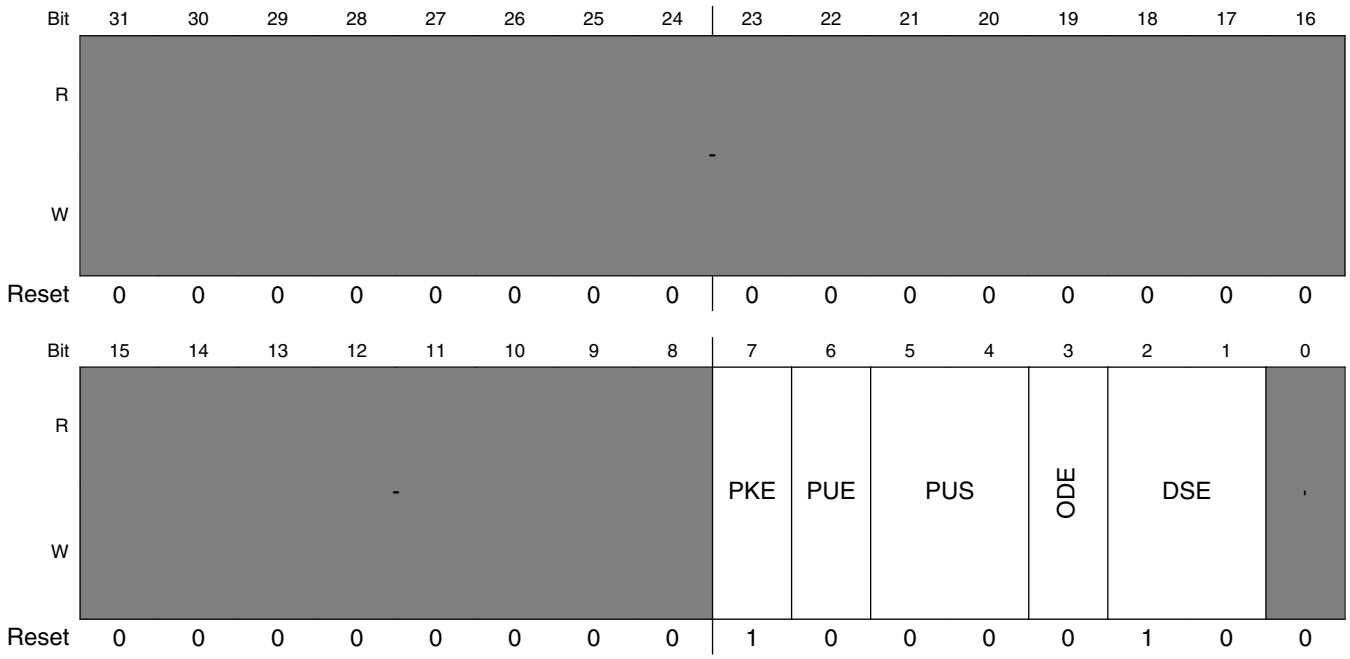
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PUART3_TXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART3_TXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART3_TXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART3_TXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART3_TXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART3_TXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.214 SW_PAD_CTL_PAD_UART3_RXD (IOMUXC_SPADC_PUART3_RXD)

Address: IOMUXC_SPADC_PUART3_RXD is 53FA_8000h base + 354h offset = 53FA_8354h



IOMUXC_SPADC_PUART3_RXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART3_RXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART3_RXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART3_RXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PUART3_RXD field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART3_RXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART3_RXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.215 SW_PAD_CTL_PAD_UART4_TXD (IOMUXC_SPADC_PUART4_TXD)

Address: IOMUXC_SPADC_PUART4_TXD is 53FA_8000h base + 358h offset = 53FA_8358h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

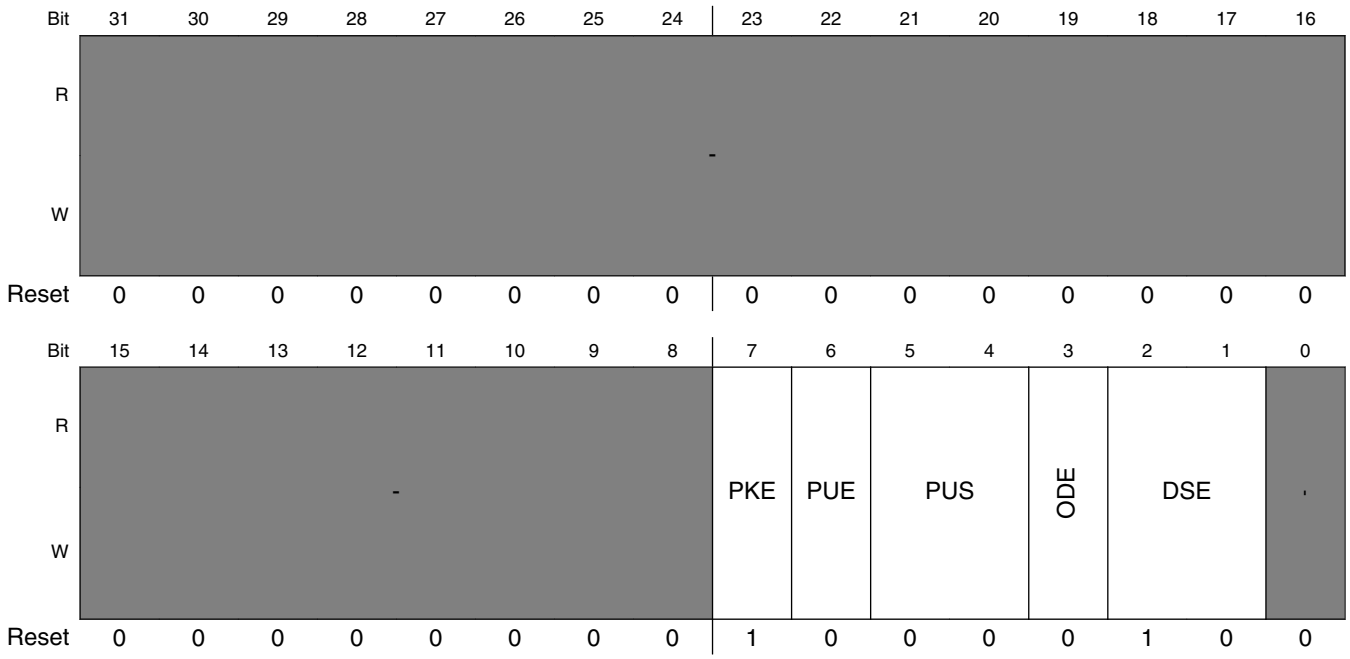
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PUART4_TXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART4_TXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART4_TXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART4_TXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART4_TXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART4_TXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.216 SW_PAD_CTL_PAD_UART4_RXD (IOMUXC_SPADC_PUART4_RXD)

Address: IOMUXC_SPADC_PUART4_RXD is 53FA_8000h base + 35Ch offset = 53FA_835Ch



IOMUXC_SPADC_PUART4_RXD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART4_RXD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART4_RXD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART4_RXD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PUART4_RXD field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: UART4_RXD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: UART4_RXD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.217 SW_PAD_CTL_PAD_CSPI_SCLK (IOMUXC_SPADC_PCSPI_SCLK)

Address: IOMUXC_SPADC_PCSPI_SCLK is 53FA_8000h base + 360h offset = 53FA_8360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PCSPI_SCLK field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI_SCLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSPI_SCLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI_SCLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSPI_SCLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: CSPI_SCLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.218 SW_PAD_CTL_PAD_CSPI_MOSI (IOMUXC_SPADC_PCSPI_MOSI)

Address: IOMUXC_SPADC_PCSPI_MOSI is 53FA_8000h base + 364h offset = 53FA_8364h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PCSPI_MOSI field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI_MOSI. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSPI_MOSI. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI_MOSI. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PCSPI_MOSI field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSPI_MOSI. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: CSPI_MOSI. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.219 SW_PAD_CTL_PAD_CSPI_MISO (IOMUXC_SPADC_PCSPI_MISO)

Address: IOMUXC_SPADC_PCSPI_MISO is 53FA_8000h base + 368h offset = 53FA_8368h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

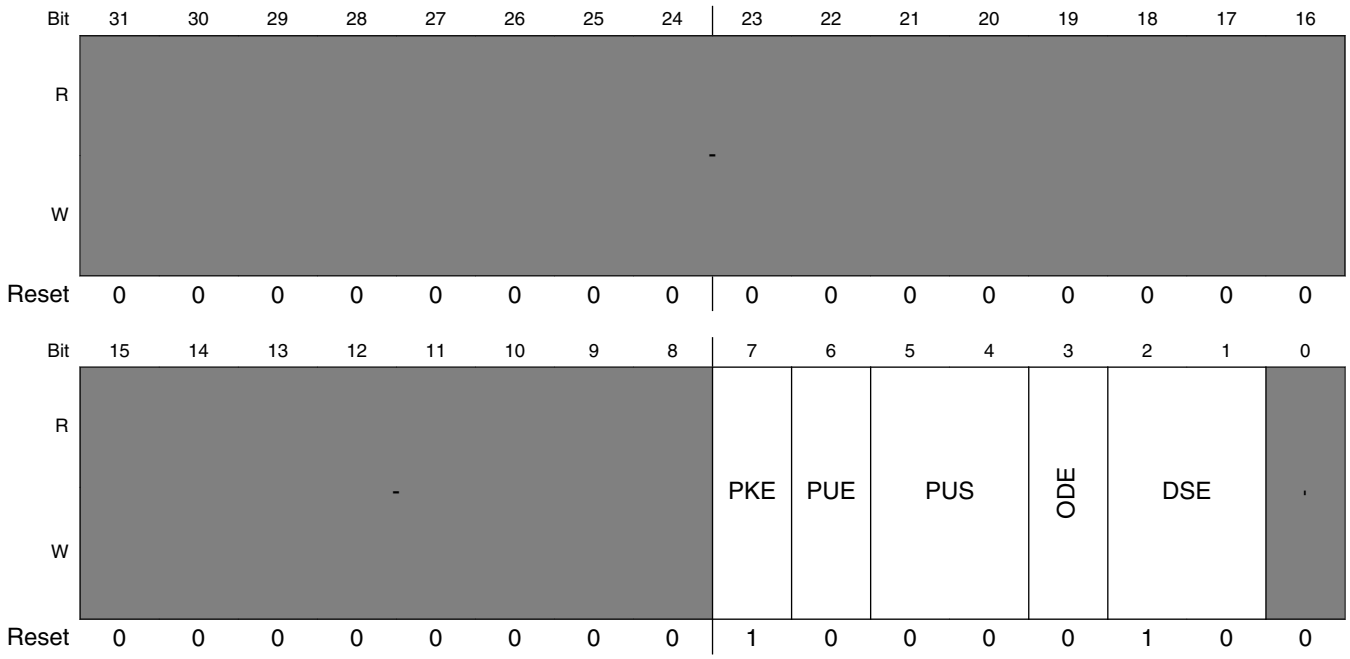
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PCSPI_MISO field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI_MISO. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSPI_MISO. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI_MISO. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSPI_MISO. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: CSPI_MISO. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.220 SW_PAD_CTL_PAD_CSPI_SS0 (IOMUXC_SPADC_PCSPI_SS0)

Address: IOMUXC_SPADC_PCSPI_SS0 is 53FA_8000h base + 36Ch offset = 53FA_836Ch



IOMUXC_SPADC_PCSPI_SS0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI_SS0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSPI_SS0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI_SS0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PCSPI_SS0 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSPI_SS0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: CSPI_SS0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.221 SW_PAD_CTL_PAD_ECSPI1_SCLK (IOMUXC_SPADC_PECSPI1_SCLK)

Address: IOMUXC_SPADC_PECSPI1_SCLK is 53FA_8000h base + 370h offset = 53FA_8370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

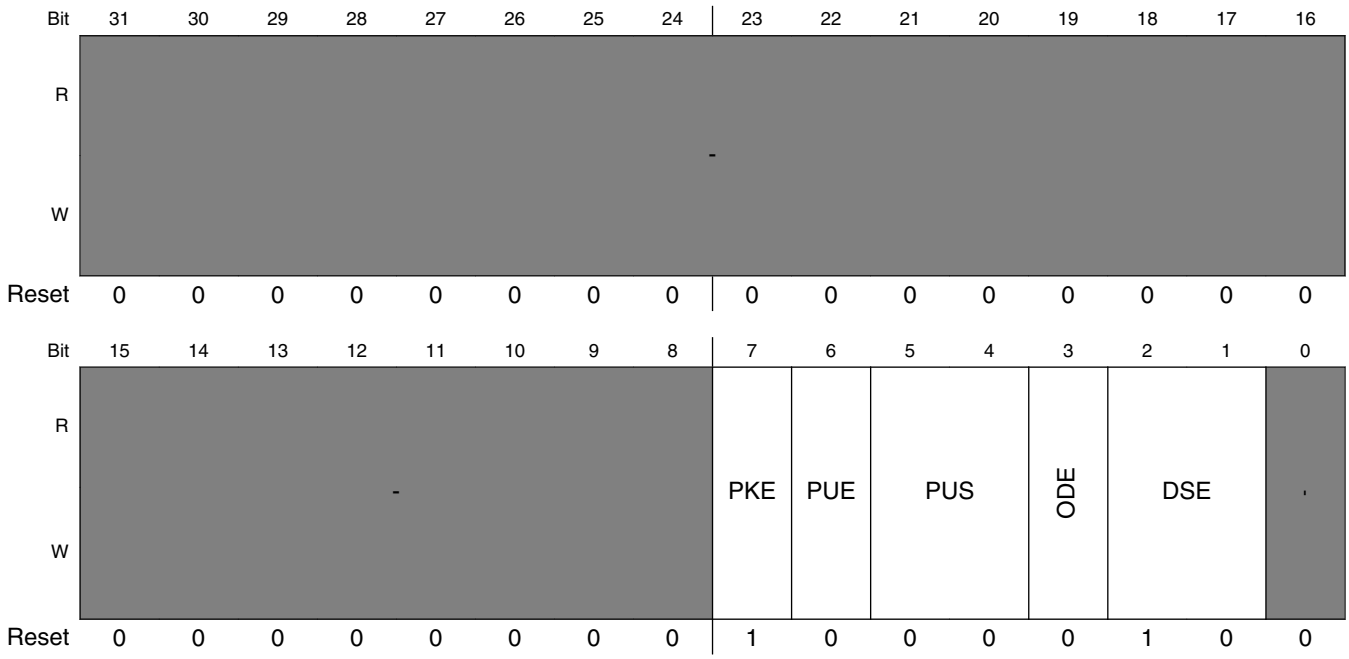
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PECSPI1_SCLK field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECSPI1_SCLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: ECSPI1_SCLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ECSPI1_SCLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: ECSPI1_SCLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: ECSPI1_SCLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.222 SW_PAD_CTL_PAD_ECSPI1_MOSI (IOMUXC_SPADC_PECSPI1_MOSI)

Address: IOMUXC_SPADC_PECSPI1_MOSI is 53FA_8000h base + 374h offset = 53FA_8374h



IOMUXC_SPADC_PECSPI1_MOSI field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECSPI1_MOSI. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: ECSPI1_MOSI. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ECSPI1_MOSI. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PECSPI1_MOSI field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: ECSPI1_MOSI. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: ECSPI1_MOSI. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.223 SW_PAD_CTL_PAD_ECSPI1_MISO (IOMUXC_SPADC_PECSPI1_MISO)

Address: IOMUXC_SPADC_PECSPI1_MISO is 53FA_8000h base + 378h offset = 53FA_8378h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PECSPI1_MISO field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECSPI1_MISO. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: ECSPI1_MISO. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ECSPI1_MISO. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: ECSPI1_MISO. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: ECSPI1_MISO. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.224 SW_PAD_CTL_PAD_ECSPI1_SS0 (IOMUXC_SPADC_PECSPI1_SS0)

Address: IOMUXC_SPADC_PECSPI1_SS0 is 53FA_8000h base + 37Ch offset = 53FA_837Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PECSPI1_SS0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECSPI1_SS0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: ECSPI1_SS0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ECSPI1_SS0. 00 100kΩ Pull Down 01 47kΩ Pull Up 10 100kΩ Pull Up 11 22kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PECSPI1_SS0 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: ECSPI1_SS0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: ECSPI1_SS0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.225 SW_PAD_CTL_PAD_ECSPI2_SCLK (IOMUXC_SPADC_PECSPI2_SCLK)

Address: IOMUXC_SPADC_PECSPI2_SCLK is 53FA_8000h base + 380h offset = 53FA_8380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

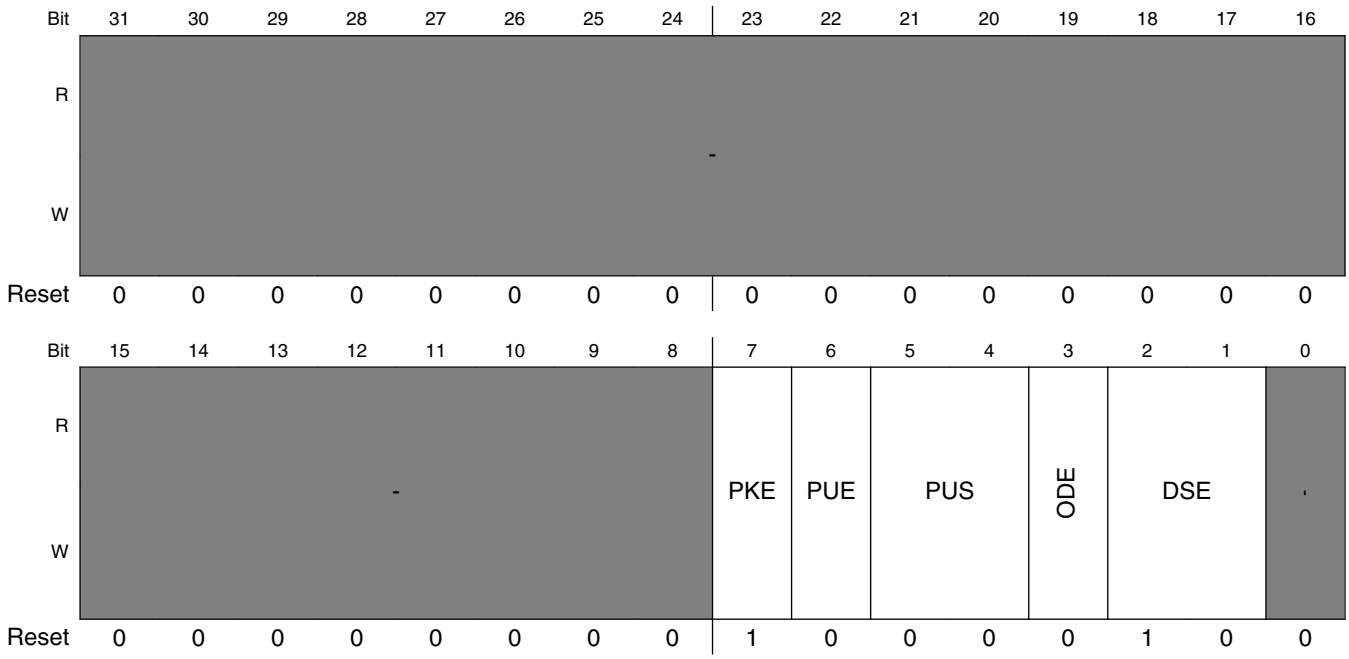
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PECSPI2_SCLK field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECSPI2_SCLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: ECSPI2_SCLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ECSPI2_SCLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: ECSPI2_SCLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: ECSPI2_SCLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.226 SW_PAD_CTL_PAD_ECSPI2_MOSI (IOMUXC_SPADC_PECSPI2_MOSI)

Address: IOMUXC_SPADC_PECSPI2_MOSI is 53FA_8000h base + 384h offset = 53FA_8384h



IOMUXC_SPADC_PECSPI2_MOSI field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECSPI2_MOSI. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: ECSPI2_MOSI. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ECSPI2_MOSI. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PECSPI2_MOSI field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: ECSPI2_MOSI. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: ECSPI2_MOSI. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.227 SW_PAD_CTL_PAD_ECSPI2_MISO (IOMUXC_SPADC_PECSPI2_MISO)

Address: IOMUXC_SPADC_PECSPI2_MISO is 53FA_8000h base + 388h offset = 53FA_8388h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PECSPI2_MISO field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECSPI2_MISO. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: ECSPI2_MISO. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ECSPI2_MISO. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: ECSPI2_MISO. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: ECSPI2_MISO. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.228 SW_PAD_CTL_PAD_ECSPI2_SS0 (IOMUXC_SPADC_PECSPI2_SS0)

Address: IOMUXC_SPADC_PECSPI2_SS0 is 53FA_8000h base + 38Ch offset = 53FA_838Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PECSPI2_SS0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECSPI2_SS0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: ECSPI2_SS0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: ECSPI2_SS0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PECSPI2_SS0 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: ECSPI2_SS0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: ECSPI2_SS0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.229 SW_PAD_CTL_PAD_SD1_CLK (IOMUXC_SPADC_PSD1_CLK)

Address: IOMUXC_SPADC_PSD1_CLK is 53FA_8000h base + 390h offset = 53FA_8390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD1_CLK field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_CLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_CLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_CLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_CLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD1_CLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.230 SW_PAD_CTL_PAD_SD1_CMD (IOMUXC_SPADC_PSD1_CMD)

Address: IOMUXC_SPADC_PSD1_CMD is 53FA_8000h base + 394h offset = 53FA_8394h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD1_CMD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_CMD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_CMD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_CMD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

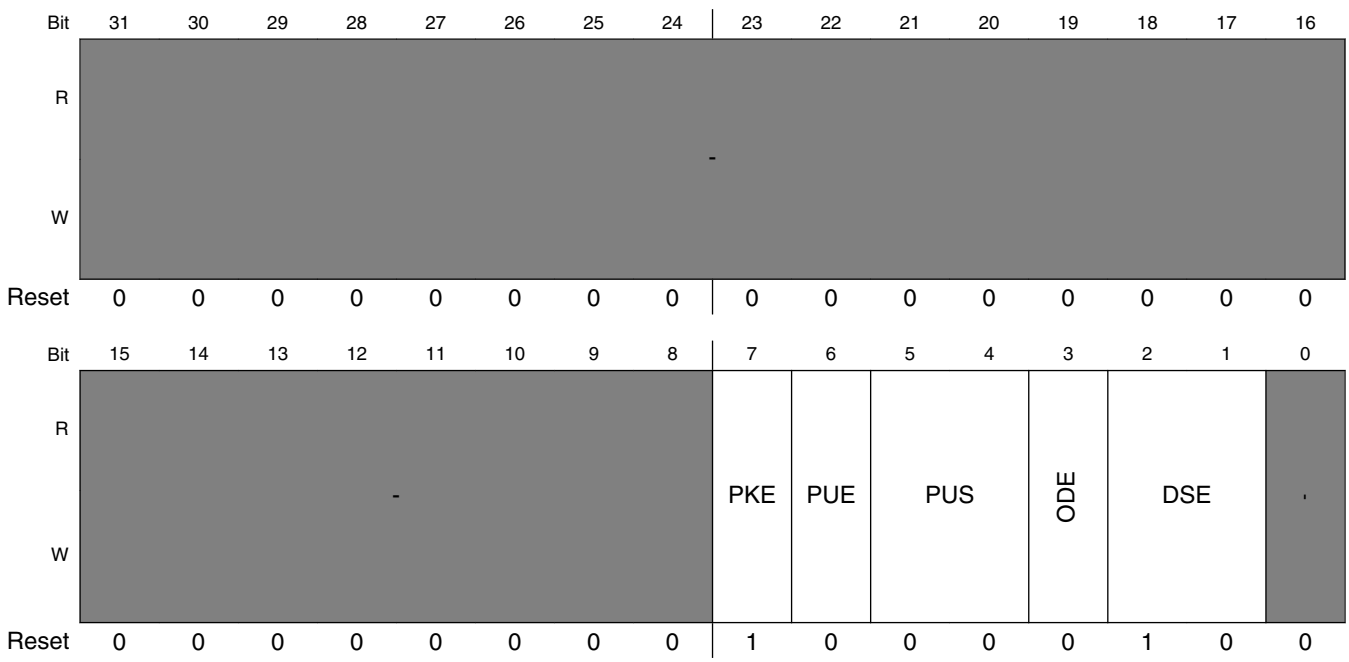
Table continues on the next page...

IOMUXC_SPADC_PSD1_CMD field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_CMD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD1_CMD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.231 SW_PAD_CTL_PAD_SD1_D0 (IOMUXC_SPADC_PSD1_D0)

Address: IOMUXC_SPADC_PSD1_D0 is 53FA_8000h base + 398h offset = 53FA_8398h



IOMUXC_SPADC_PSD1_D0 field descriptions

Field	Description
31–8 -	Reserved

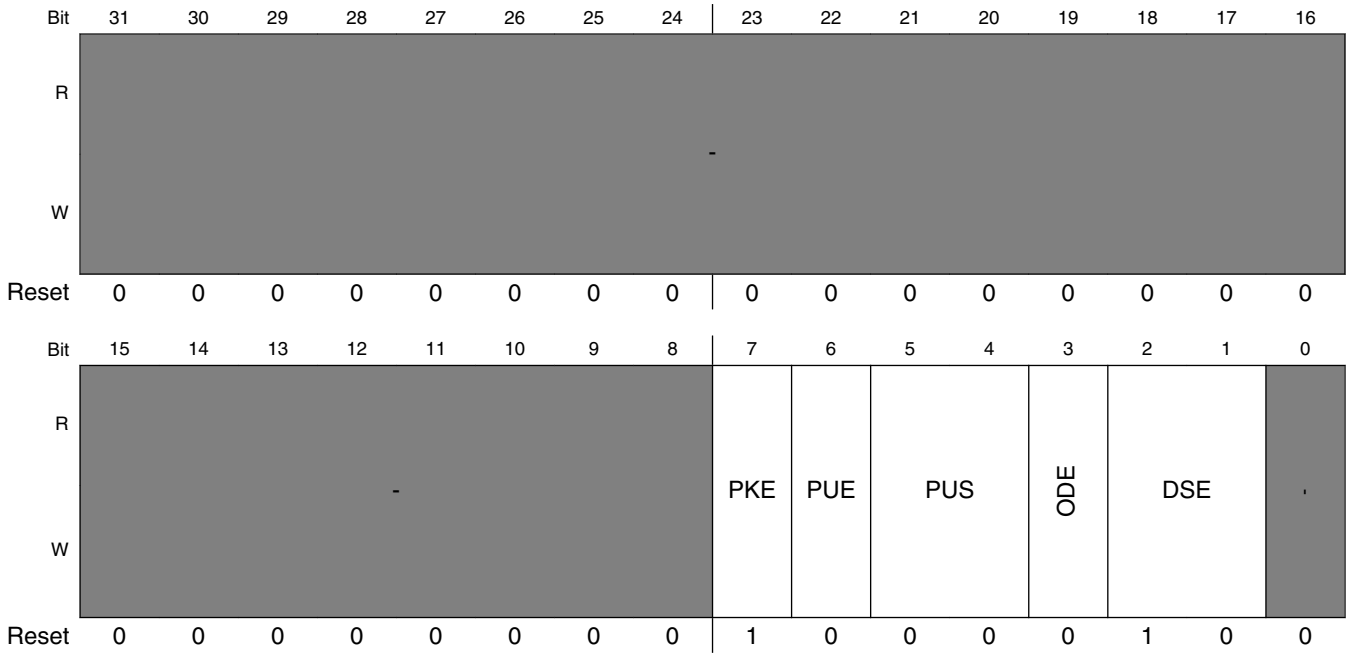
Table continues on the next page...

IOMUXC_SPADC_PSD1_D0 field descriptions (continued)

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_D0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_D0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_D0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_D0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD1_D0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.232 SW_PAD_CTL_PAD_SD1_D1 (IOMUXC_SPADC_PSD1_D1)

Address: IOMUXC_SPADC_PSD1_D1 is 53FA_8000h base + 39Ch offset = 53FA_839Ch



IOMUXC_SPADC_PSD1_D1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_D1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_D1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_D1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

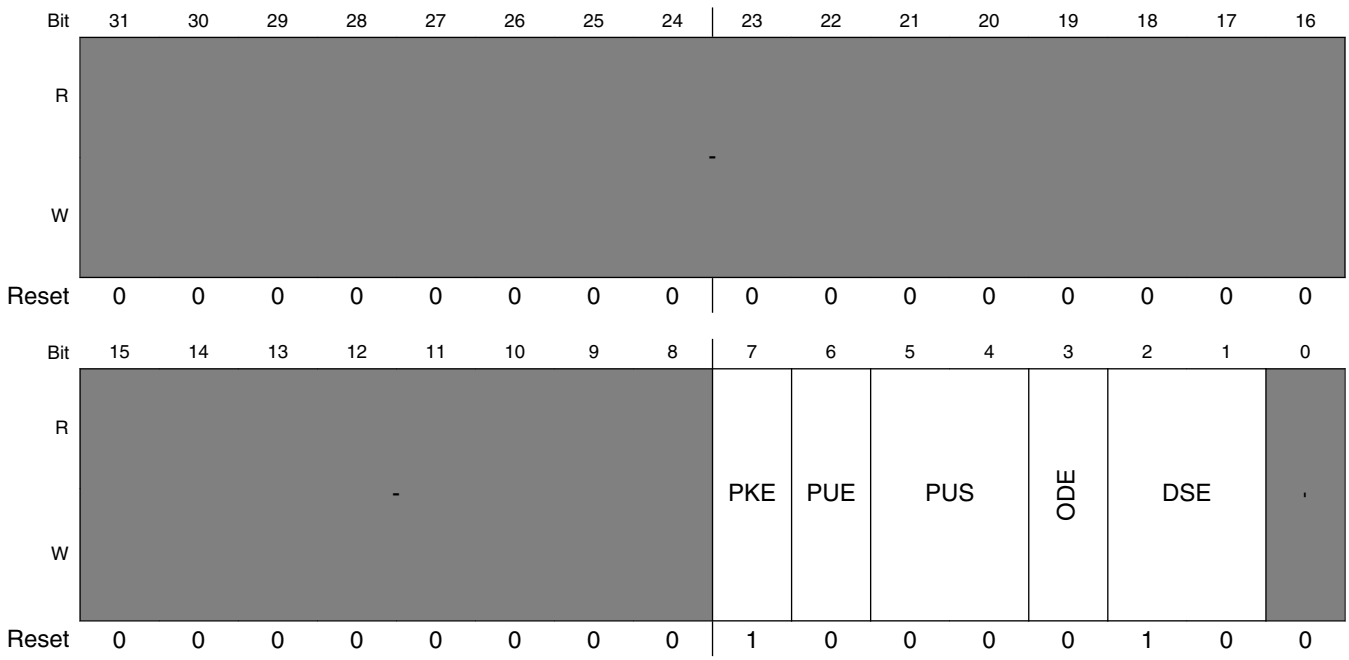
Table continues on the next page...

IOMUXC_SPADC_PSD1_D1 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD1_D1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD1_D1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.233 SW_PAD_CTL_PAD_SD1_D2 (IOMUXC_SPADC_PSD1_D2)

Address: IOMUXC_SPADC_PSD1_D2 is 53FA_8000h base + 3A0h offset = 53FA_83A0h



IOMUXC_SPADC_PSD1_D2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_D2.

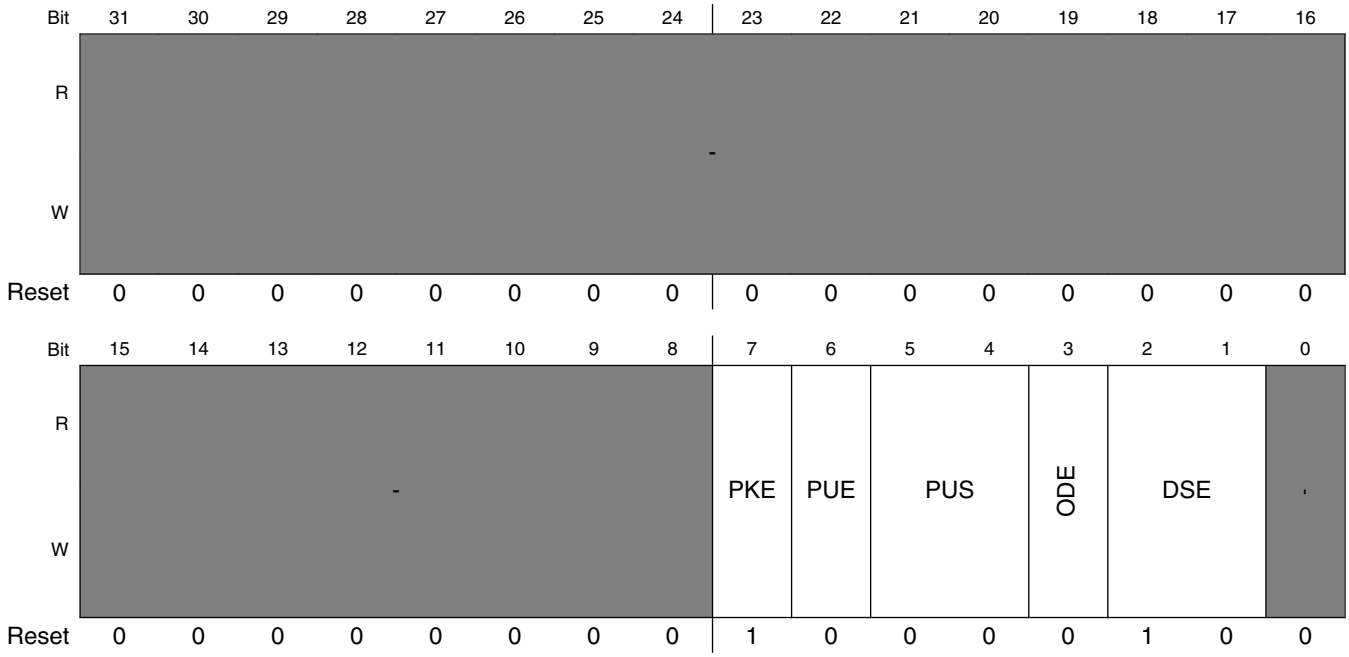
Table continues on the next page...

IOMUXC_SPADC_PSD1_D2 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_D2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_D2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD1_D2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD1_D2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.234 SW_PAD_CTL_PAD_SD1_D3 (IOMUXC_SPADC_PSD1_D3)

Address: IOMUXC_SPADC_PSD1_D3 is 53FA_8000h base + 3A4h offset = 53FA_83A4h



IOMUXC_SPADC_PSD1_D3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_D3. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_D3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_D3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD1_D3 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD1_D3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD1_D3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.235 SW_PAD_CTL_PAD_SD2_CLK (IOMUXC_SPADC_PSD2_CLK)

Address: IOMUXC_SPADC_PSD2_CLK is 53FA_8000h base + 3A8h offset = 53FA_83A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD2_CLK field descriptions

Field	Description
31–8 -	Reserved

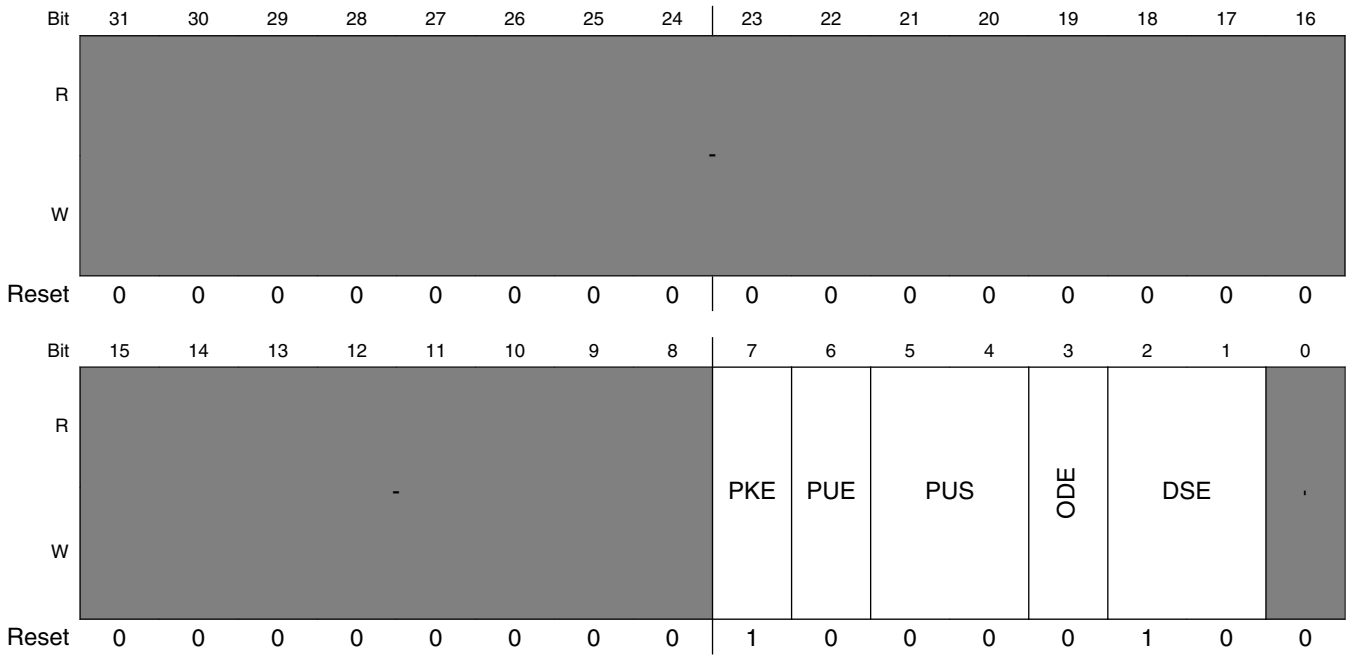
Table continues on the next page...

IOMUXC_SPADC_PSD2_CLK field descriptions (continued)

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_CLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_CLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_CLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD2_CLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_CLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.236 SW_PAD_CTL_PAD_SD2_CMD (IOMUXC_SPADC_PSD2_CMD)

Address: IOMUXC_SPADC_PSD2_CMD is 53FA_8000h base + 3ACh offset = 53FA_83ACh



IOMUXC_SPADC_PSD2_CMD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_CMD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_CMD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_CMD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PSD2_CMD field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD2_CMD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_CMD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.237 SW_PAD_CTL_PAD_SD2_D0 (IOMUXC_SPADC_PSD2_D0)

Address: IOMUXC_SPADC_PSD2_D0 is 53FA_8000h base + 3B0h offset = 53FA_83B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD2_D0 field descriptions

Field	Description
31–8 -	Reserved

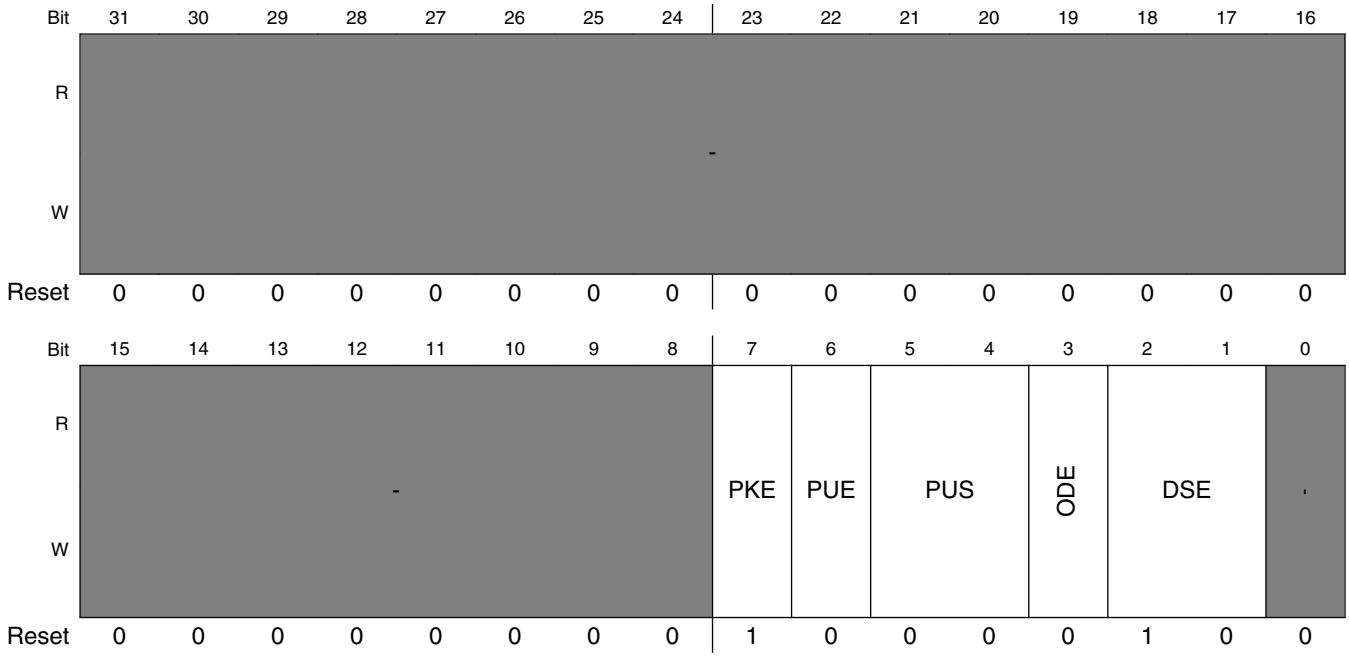
Table continues on the next page...

IOMUXC_SPADC_PSD2_D0 field descriptions (continued)

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_D0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_D0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_D0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD2_D0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_D0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.238 SW_PAD_CTL_PAD_SD2_D1 (IOMUXC_SPADC_PSD2_D1)

Address: IOMUXC_SPADC_PSD2_D1 is 53FA_8000h base + 3B4h offset = 53FA_83B4h



IOMUXC_SPADC_PSD2_D1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_D1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_D1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_D1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD2_D1 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD2_D1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_D1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.239 SW_PAD_CTL_PAD_SD2_D2 (IOMUXC_SPADC_PSD2_D2)

Address: IOMUXC_SPADC_PSD2_D2 is 53FA_8000h base + 3B8h offset = 53FA_83B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD2_D2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_D2.

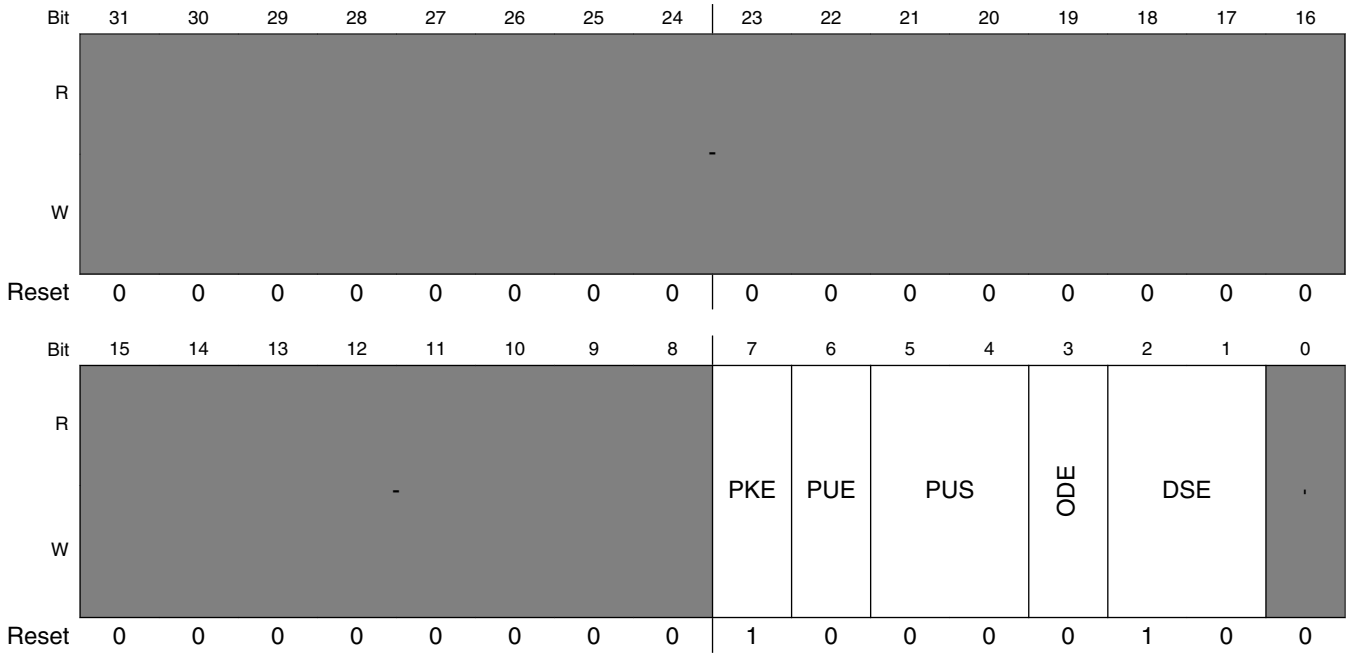
Table continues on the next page...

IOMUXC_SPADC_PSD2_D2 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_D2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_D2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD2_D2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_D2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.240 SW_PAD_CTL_PAD_SD2_D3 (IOMUXC_SPADC_PSD2_D3)

Address: IOMUXC_SPADC_PSD2_D3 is 53FA_8000h base + 3BCh offset = 53FA_83BCh



IOMUXC_SPADC_PSD2_D3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_D3. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_D3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_D3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

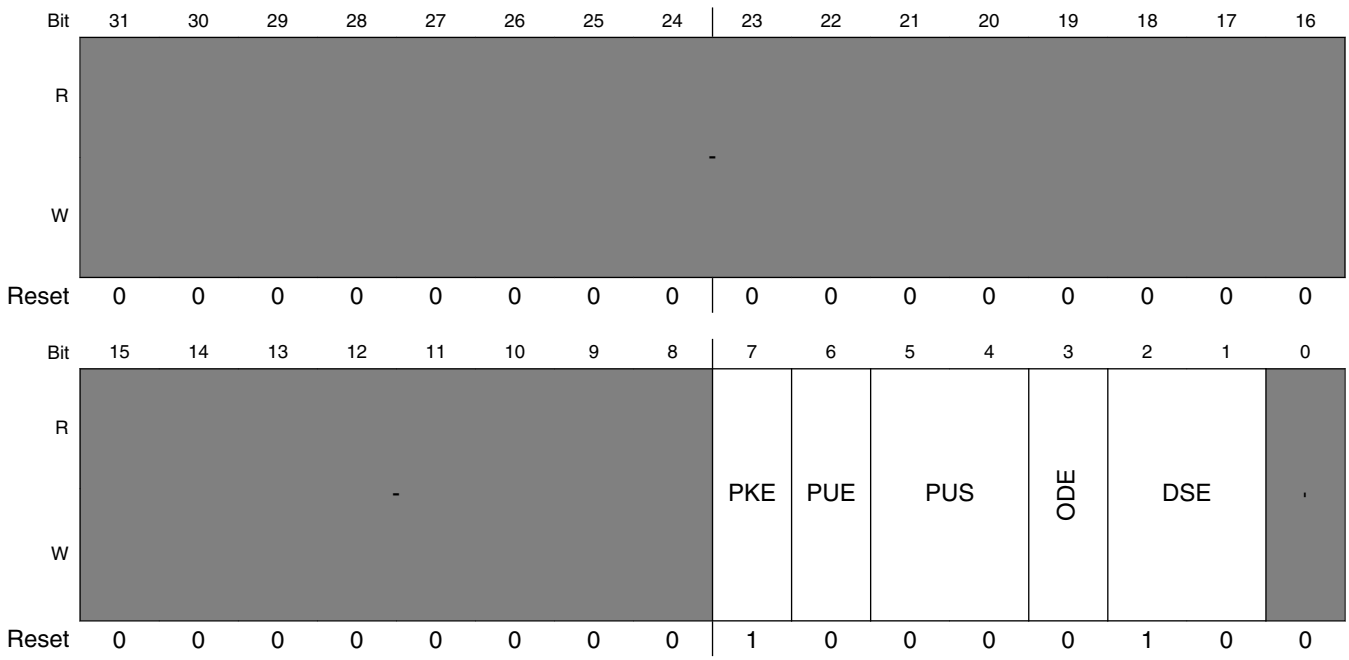
Table continues on the next page...

IOMUXC_SPADC_PSD2_D3 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD2_D3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_D3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.241 SW_PAD_CTL_PAD_SD2_D4 (IOMUXC_SPADC_PSD2_D4)

Address: IOMUXC_SPADC_PSD2_D4 is 53FA_8000h base + 3C0h offset = 53FA_83C0h



IOMUXC_SPADC_PSD2_D4 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_D4.

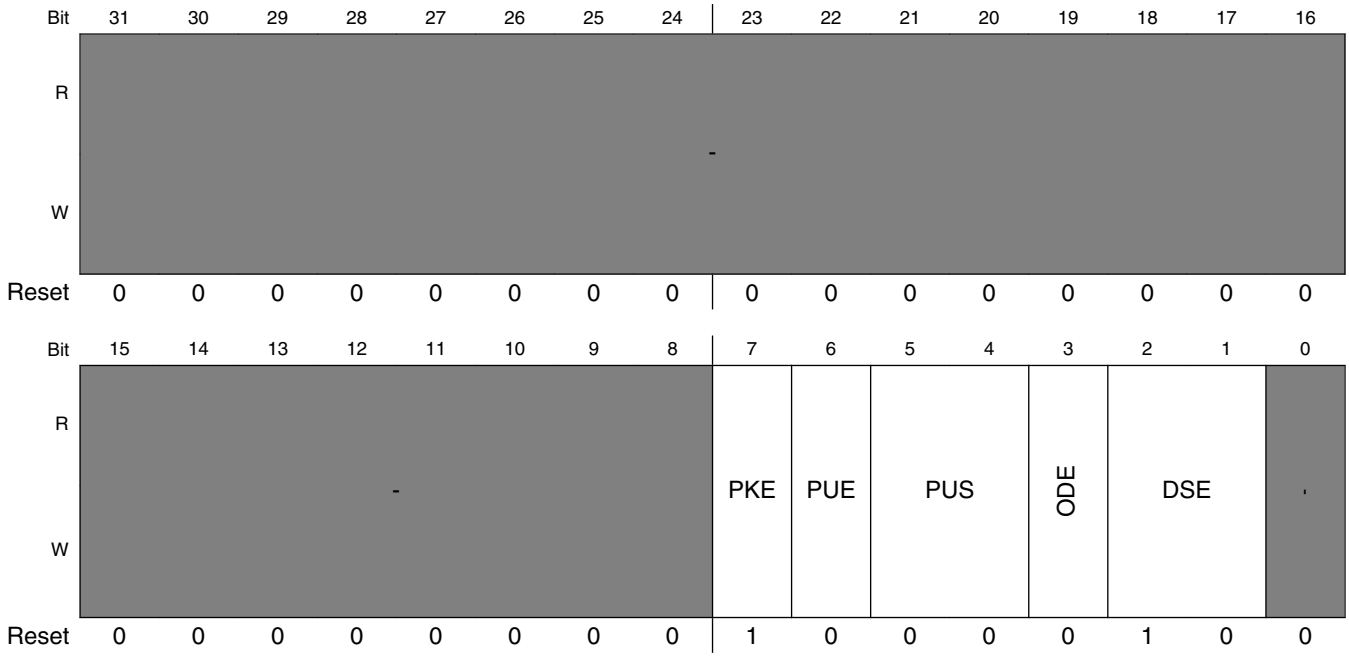
Table continues on the next page...

IOMUXC_SPADC_PSD2_D4 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_D4. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_D4. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD2_D4. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_D4. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.242 SW_PAD_CTL_PAD_SD2_D5 (IOMUXC_SPADC_PSD2_D5)

Address: IOMUXC_SPADC_PSD2_D5 is 53FA_8000h base + 3C4h offset = 53FA_83C4h



IOMUXC_SPADC_PSD2_D5 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_D5. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_D5. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_D5. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD2_D5 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD2_D5. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_D5. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.243 SW_PAD_CTL_PAD_SD2_D6 (IOMUXC_SPADC_PSD2_D6)

Address: IOMUXC_SPADC_PSD2_D6 is 53FA_8000h base + 3C8h offset = 53FA_83C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD2_D6 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_D6.

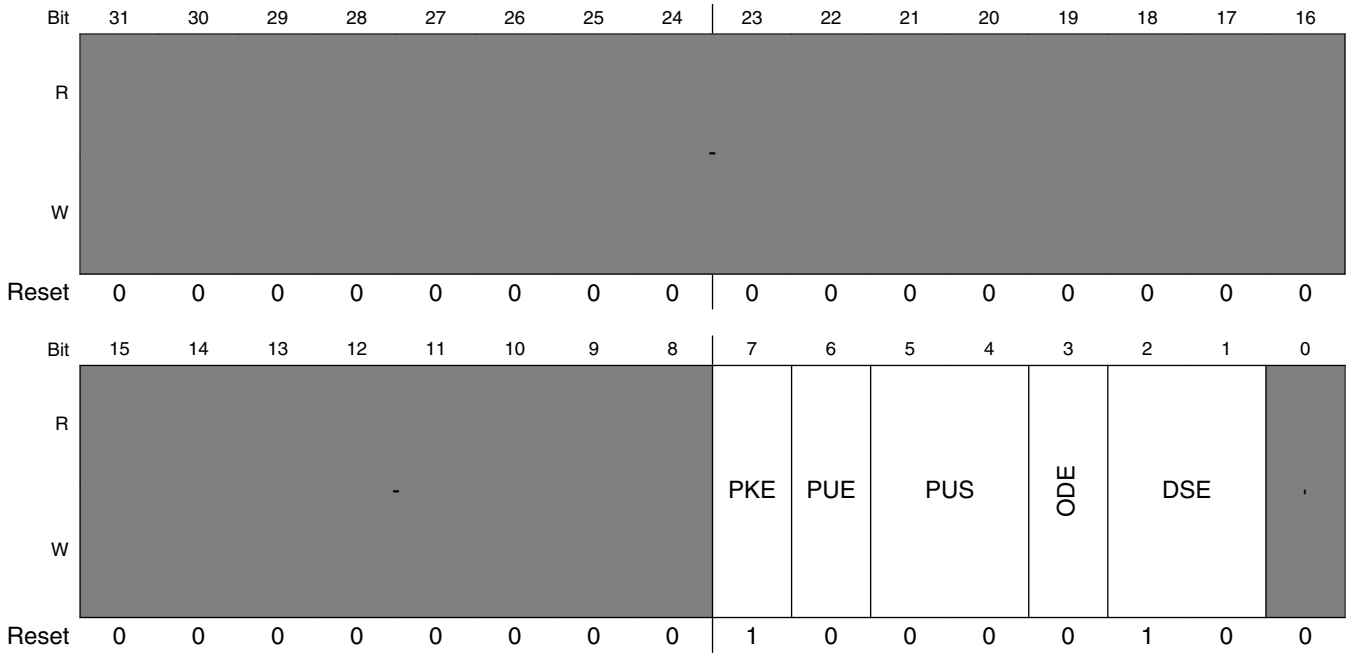
Table continues on the next page...

IOMUXC_SPADC_PSD2_D6 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_D6. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_D6. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD2_D6. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_D6. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.244 SW_PAD_CTL_PAD_SD2_D7 (IOMUXC_SPADC_PSD2_D7)

Address: IOMUXC_SPADC_PSD2_D7 is 53FA_8000h base + 3CCh offset = 53FA_83CCh



IOMUXC_SPADC_PSD2_D7 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_D7. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_D7. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_D7. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD2_D7 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD2_D7. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_D7. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.245 SW_PAD_CTL_PAD_SD2_WP (IOMUXC_SPADC_PSD2_WP)

Address: IOMUXC_SPADC_PSD2_WP is 53FA_8000h base + 3D0h offset = 53FA_83D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD2_WP field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_WP.

Table continues on the next page...

IOMUXC_SPADC_PSD2_WP field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_WP. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_WP. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD2_WP. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_WP. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.246 SW_PAD_CTL_PAD_SD2_CD (IOMUXC_SPADC_PSD2_CD)

Address: IOMUXC_SPADC_PSD2_CD is 53FA_8000h base + 3D4h offset = 53FA_83D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD2_CD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD2_CD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD2_CD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD2_CD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD2_CD field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD2_CD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD2_CD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.247 SW_PAD_CTL_PAD_PMIC_ON_REQ (IOMUXC_SPADC_PPMIC_ON_REQ)

Address: IOMUXC_SPADC_PPMIC_ON_REQ is 53FA_8000h base + 3D8h offset = 53FA_83D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-												ODE	-		SRE
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1

IOMUXC_SPADC_PPMIC_ON_REQ field descriptions

Field	Description
31–4 -	Reserved
3 ODE	Open Drain Enable Field Select one out of next values for pad: PMIC_ON_REQ. 0 Open Drain Disabled 1 Open Drain Enabled

Table continues on the next page...

IOMUXC_SPADC_PPMIC_ON_REQ field descriptions (continued)

Field	Description
2–1 -	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: PMIC_ON_REQ. 0 Slow Slew Rate 1 Fast Slew Rate

35.4.248 SW_PAD_CTL_PAD_PMIC_STBY_REQ (IOMUXC_SPADC_PPMIC_STBY_REQ)

Address: IOMUXC_SPADC_PPMIC_STBY_REQ is 53FA_8000h base + 3DCh offset = 53FA_83DCh

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

IOMUXC_SPADC_PPMIC_STBY_REQ field descriptions

Field	Description
31–3 -	Reserved
2–1 DSE	Drive Strength Field Select one out of next values for pad: PMIC_STBY_REQ. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 SRE	Slew Rate Field Select one out of next values for pad: PMIC_STBY_REQ. 0 Slow Slew Rate 1 Fast Slew Rate

35.4.249 SW_PAD_CTL_PAD_POR_B (IOMUXC_SPADC_PPOR_B)

Address: IOMUXC_SPADC_PPOR_B is 53FA_8000h base + 3E0h offset = 53FA_83E0h

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

IOMUXC_SPADC_PPOR_B field descriptions

Field	Description
31–9 -	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: POR_B. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: POR_B. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.250 SW_PAD_CTL_PAD_BOOT_MODE1 (IOMUXC_SPADC_PBOOT_MODE1)

Address: IOMUXC_SPADC_PBOOT_MODE1 is 53FA_8000h base + 3E4h offset = 53FA_83E4h

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

IOMUXC_SPADC_PBOOT_MODE1 field descriptions

Field	Description
31–9 -	Reserved

Table continues on the next page...

IOMUXC_SPADC_PBOOT_MODE1 field descriptions (continued)

Field	Description
8 HYS	Hyst. Enable Field Select one out of next values for pad: BOOT_MODE1. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: BOOT_MODE1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.251 SW_PAD_CTL_PAD_RESET_IN_B (IOMUXC_SPADC_PRESET_IN_B)

Address: IOMUXC_SPADC_PRESET_IN_B is 53FA_8000h base + 3E8h offset = 53FA_83E8h

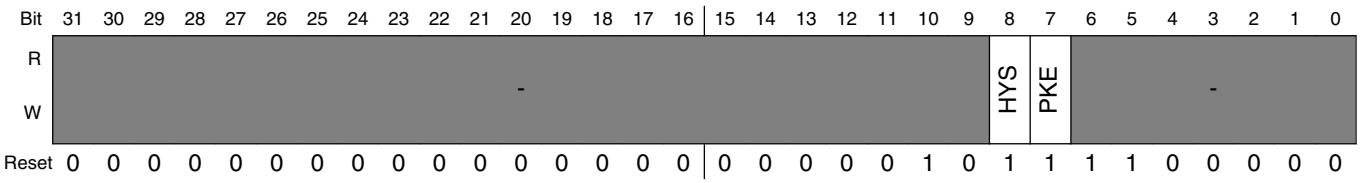
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																							HYS	-							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0

IOMUXC_SPADC_PRESET_IN_B field descriptions

Field	Description
31–9 -	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: RESET_IN_B. 0 Hysteresis Disabled 1 Hysteresis Enabled
7–0 -	Reserved

35.4.252 SW_PAD_CTL_PAD_BOOT_MODE0 (IOMUXC_SPADC_PBOOT_MODE0)

Address: IOMUXC_SPADC_PBOOT_MODE0 is 53FA_8000h base + 3ECh offset = 53FA_83ECh

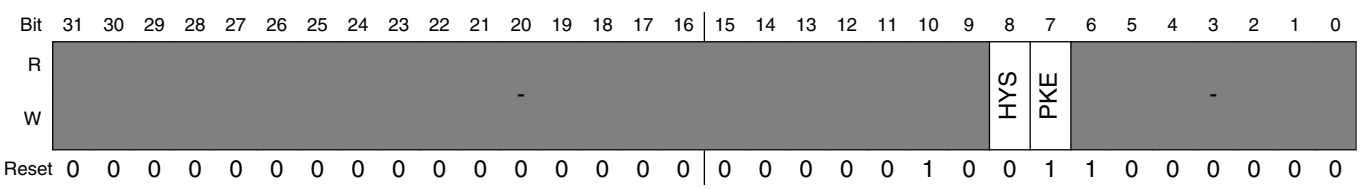


IOMUXC_SPADC_PBOOT_MODE0 field descriptions

Field	Description
31–9 -	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: BOOT_MODE0. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: BOOT_MODE0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.253 SW_PAD_CTL_PAD_TEST_MODE (IOMUXC_SPADC_PTEST_MODE)

Address: IOMUXC_SPADC_PTEST_MODE is 53FA_8000h base + 3F0h offset = 53FA_83F0h



IOMUXC_SPADC_PTEST_MODE field descriptions

Field	Description
31–9 -	Reserved

Table continues on the next page...

IOMUXC_SPADC_PTEST_MODE field descriptions (continued)

Field	Description
8 HYS	Hyst. Enable Field Select one out of next values for pad: TEST_MODE. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: TEST_MODE. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.254 SW_PAD_CTL_PAD_JTAG_TMS (IOMUXC_SPADC_PJTAG_TMS)

Address: IOMUXC_SPADC_PJTAG_TMS is 53FA_8000h base + 3F4h offset = 53FA_83F4h

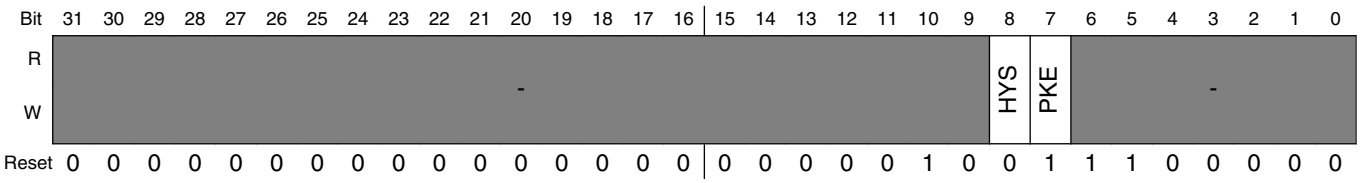
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																							HYS	PKE	-						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0

IOMUXC_SPADC_PJTAG_TMS field descriptions

Field	Description
31–9 -	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TMS. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TMS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.255 SW_PAD_CTL_PAD_JTAG_MOD (IOMUXC_SPADC_PJTAG_MOD)

Address: IOMUXC_SPADC_PJTAG_MOD is 53FA_8000h base + 3F8h offset = 53FA_83F8h

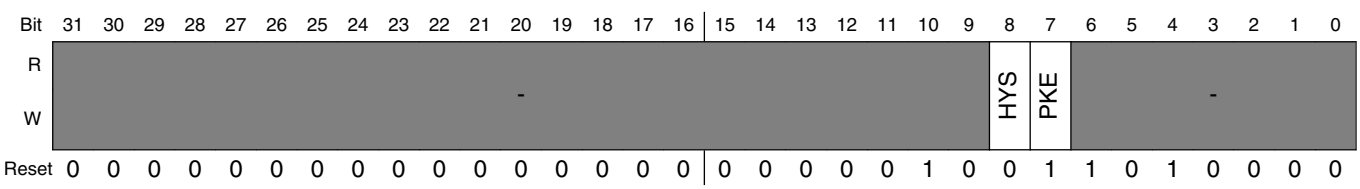


IOMUXC_SPADC_PJTAG_MOD field descriptions

Field	Description
31–9 -	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_MOD. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_MOD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.256 SW_PAD_CTL_PAD_JTAG_TRSTB (IOMUXC_SPADC_PJTAG_TRSTB)

Address: IOMUXC_SPADC_PJTAG_TRSTB is 53FA_8000h base + 3FCh offset = 53FA_83FCh



IOMUXC_SPADC_PJTAG_TRSTB field descriptions

Field	Description
31–9 -	Reserved

Table continues on the next page...

IOMUXC_SPADC_PJTAG_TRSTB field descriptions (continued)

Field	Description
8 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TRSTB. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TRSTB. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.257 SW_PAD_CTL_PAD_JTAG_TDI (IOMUXC_SPADC_PJTAG_TDI)

Address: IOMUXC_SPADC_PJTAG_TDI is 53FA_8000h base + 400h offset = 53FA_8400h

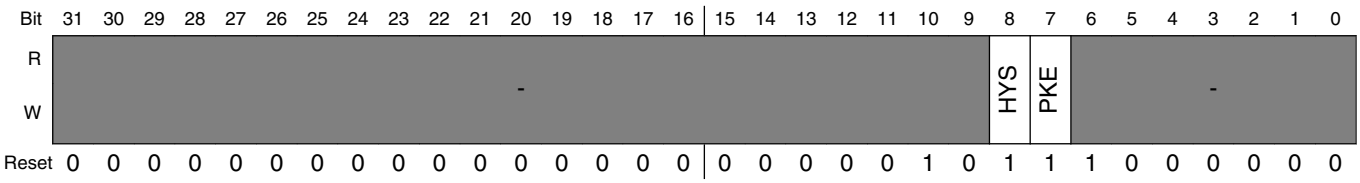
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																-							HYS	PKE	-						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0

IOMUXC_SPADC_PJTAG_TDI field descriptions

Field	Description
31–9 -	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TDI. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TDI. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.258 SW_PAD_CTL_PAD_JTAG_TCK (IOMUXC_SPADC_PJTAG_TCK)

Address: IOMUXC_SPADC_PJTAG_TCK is 53FA_8000h base + 404h offset = 53FA_8404h

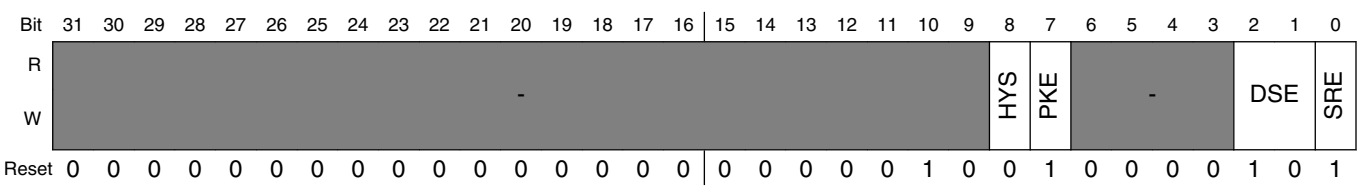


IOMUXC_SPADC_PJTAG_TCK field descriptions

Field	Description
31–9 -	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TCK. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TCK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.259 SW_PAD_CTL_PAD_JTAG_TDO (IOMUXC_SPADC_PJTAG_TDO)

Address: IOMUXC_SPADC_PJTAG_TDO is 53FA_8000h base + 408h offset = 53FA_8408h



IOMUXC_SPADC_PJTAG_TDO field descriptions

Field	Description
31–9 -	Reserved

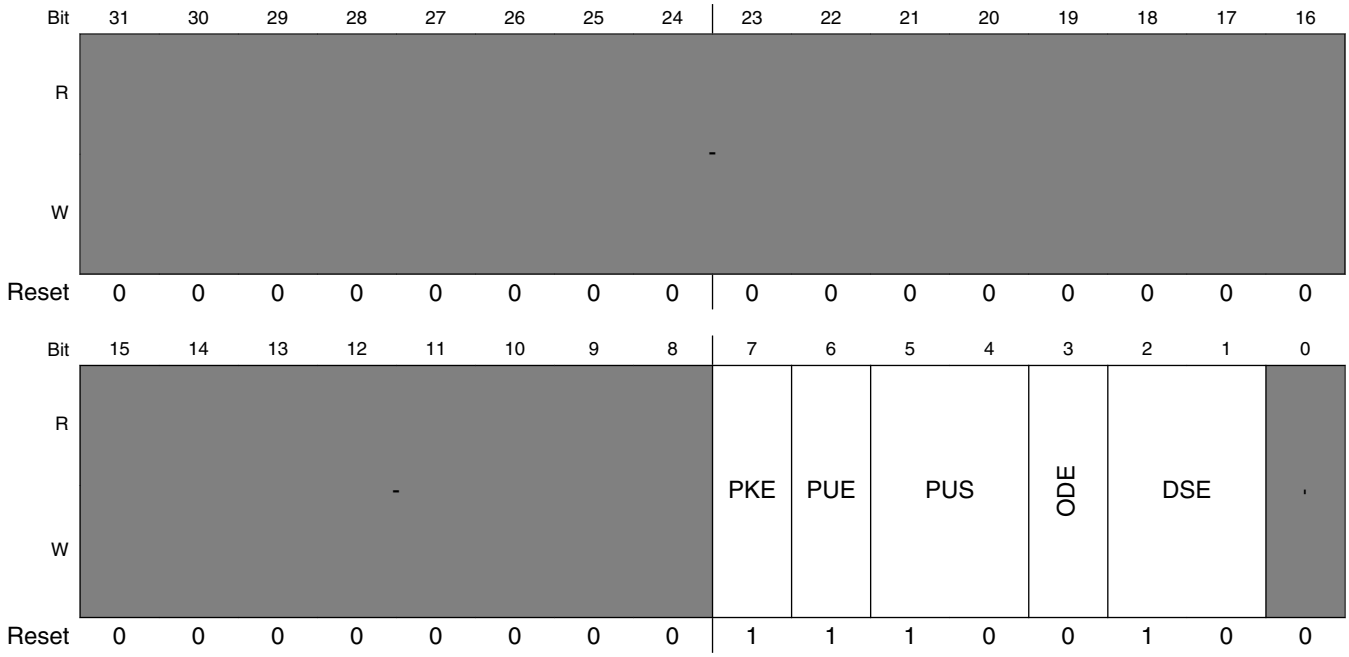
Table continues on the next page...

IOMUXC_SPADC_PJTAG_TDO field descriptions (continued)

Field	Description
8 HYS	Hyst. Enable Field Select one out of next values for pad: JTAG_TDO. 0 Hysteresis Disabled 1 Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: JTAG_TDO. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–3 -	Reserved
2–1 DSE	Drive Strength Field Select one out of next values for pad: JTAG_TDO. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 SRE	Slew Rate Field Select one out of next values for pad: JTAG_TDO. 0 Slow Slew Rate 1 Fast Slew Rate

35.4.260 SW_PAD_CTL_PAD_DISP_D0 (IOMUXC_SPADC_PDISP_D0)

Address: IOMUXC_SPADC_PDISP_D0 is 53FA_8000h base + 40Ch offset = 53FA_840Ch



IOMUXC_SPADC_PDISP_D0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

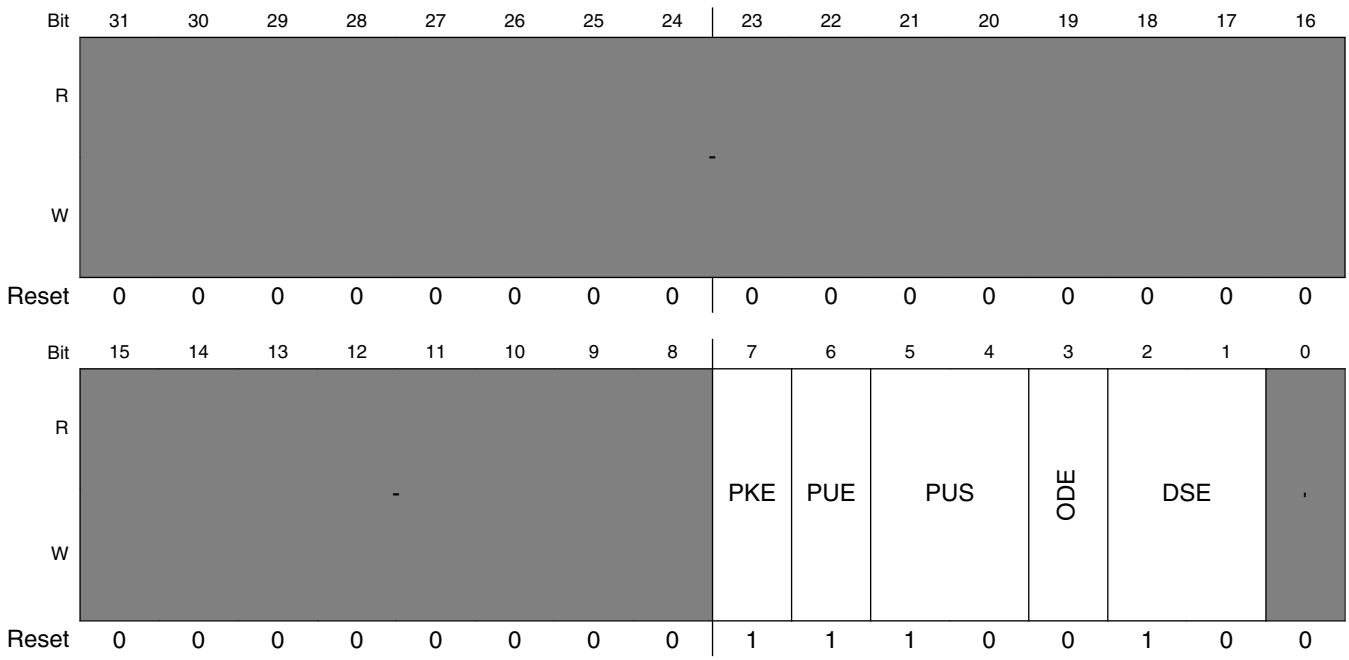
Table continues on the next page...

IOMUXC_SPADC_PDISP_D0 field descriptions (continued)

Field	Description
	Select one out of next values for pad: DISP_D0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.261 SW_PAD_CTL_PAD_DISP_D1 (IOMUXC_SPADC_PDISP_D1)

Address: IOMUXC_SPADC_PDISP_D1 is 53FA_8000h base + 410h offset = 53FA_8410h



IOMUXC_SPADC_PDISP_D1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D1.

Table continues on the next page...

IOMUXC_SPADC_PDISP_D1 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.262 SW_PAD_CTL_PAD_DISP_D2 (IOMUXC_SPADC_PDISP_D2)

Address: IOMUXC_SPADC_PDISP_D2 is 53FA_8000h base + 414h offset = 53FA_8414h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PDISP_D2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PDISP_D2 field descriptions (continued)

Field	Description
	Select one out of next values for pad: DISP_D2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.263 SW_PAD_CTL_PAD_DISP_D3 (IOMUXC_SPADC_PDISP_D3)

Address: IOMUXC_SPADC_PDISP_D3 is 53FA_8000h base + 418h offset = 53FA_8418h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PDISP_D3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D3.

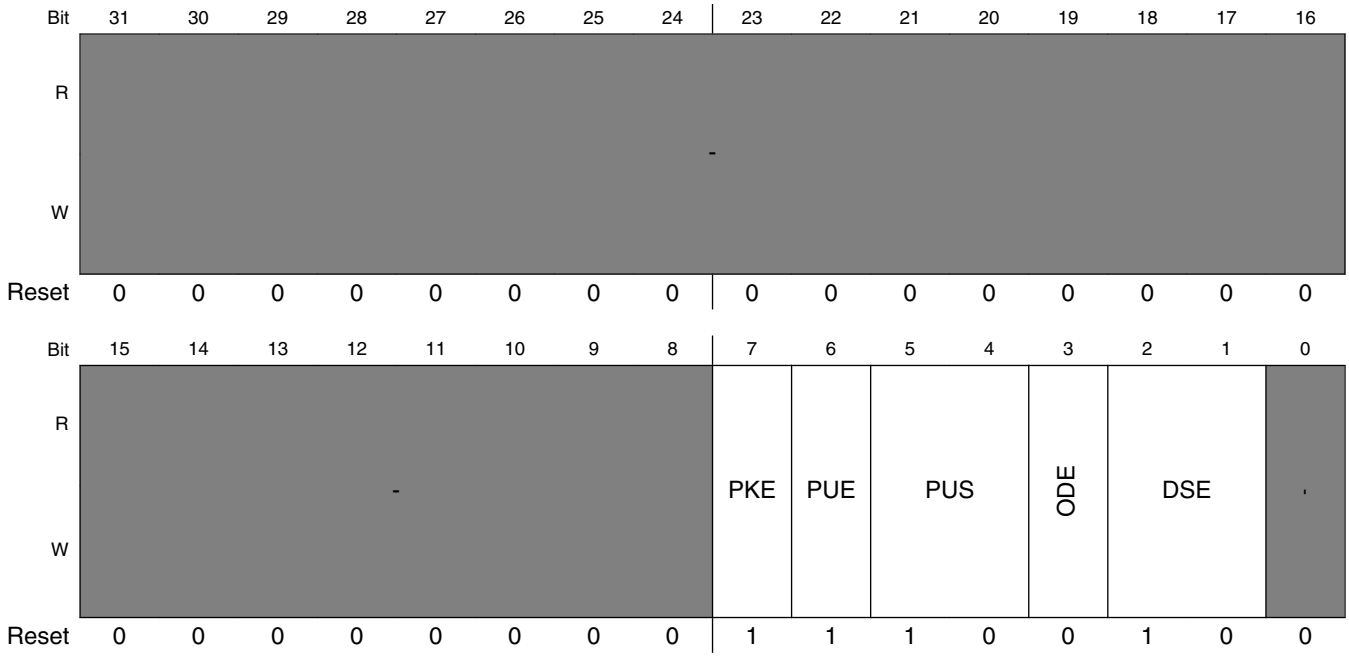
Table continues on the next page...

IOMUXC_SPADC_PDISP_D3 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.264 SW_PAD_CTL_PAD_DISP_D4 (IOMUXC_SPADC_PDISP_D4)

Address: IOMUXC_SPADC_PDISP_D4 is 53FA_8000h base + 41Ch offset = 53FA_841Ch



IOMUXC_SPADC_PDISP_D4 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D4. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D4. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D4. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PDISP_D4 field descriptions (continued)

Field	Description
	Select one out of next values for pad: DISP_D4. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D4. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.265 SW_PAD_CTL_PAD_DISP_D5 (IOMUXC_SPADC_PDISP_D5)

Address: IOMUXC_SPADC_PDISP_D5 is 53FA_8000h base + 420h offset = 53FA_8420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PDISP_D5 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D5.

Table continues on the next page...

IOMUXC_SPADC_PDISP_D5 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D5. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D5. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D5. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D5. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.266 SW_PAD_CTL_PAD_DISP_D6 (IOMUXC_SPADC_PDISP_D6)

Address: IOMUXC_SPADC_PDISP_D6 is 53FA_8000h base + 424h offset = 53FA_8424h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PDISP_D6 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D6. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D6. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D6. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PDISP_D6 field descriptions (continued)

Field	Description
	Select one out of next values for pad: DISP_D6. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D6. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.267 SW_PAD_CTL_PAD_DISP_D7 (IOMUXC_SPADC_PDISP_D7)

Address: IOMUXC_SPADC_PDISP_D7 is 53FA_8000h base + 428h offset = 53FA_8428h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PDISP_D7 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D7.

Table continues on the next page...

IOMUXC_SPADC_PDISP_D7 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D7. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D7. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D7. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D7. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.268 SW_PAD_CTL_PAD_DISP_WR (IOMUXC_SPADC_PDISP_WR)

Address: IOMUXC_SPADC_PDISP_WR is 53FA_8000h base + 42Ch offset = 53FA_842Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PDISP_WR field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_WR. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_WR. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_WR. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PDISP_WR field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_WR. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_WR. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.269 SW_PAD_CTL_PAD_DISP_RD (IOMUXC_SPADC_PDISP_RD)

Address: IOMUXC_SPADC_PDISP_RD is 53FA_8000h base + 430h offset = 53FA_8430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

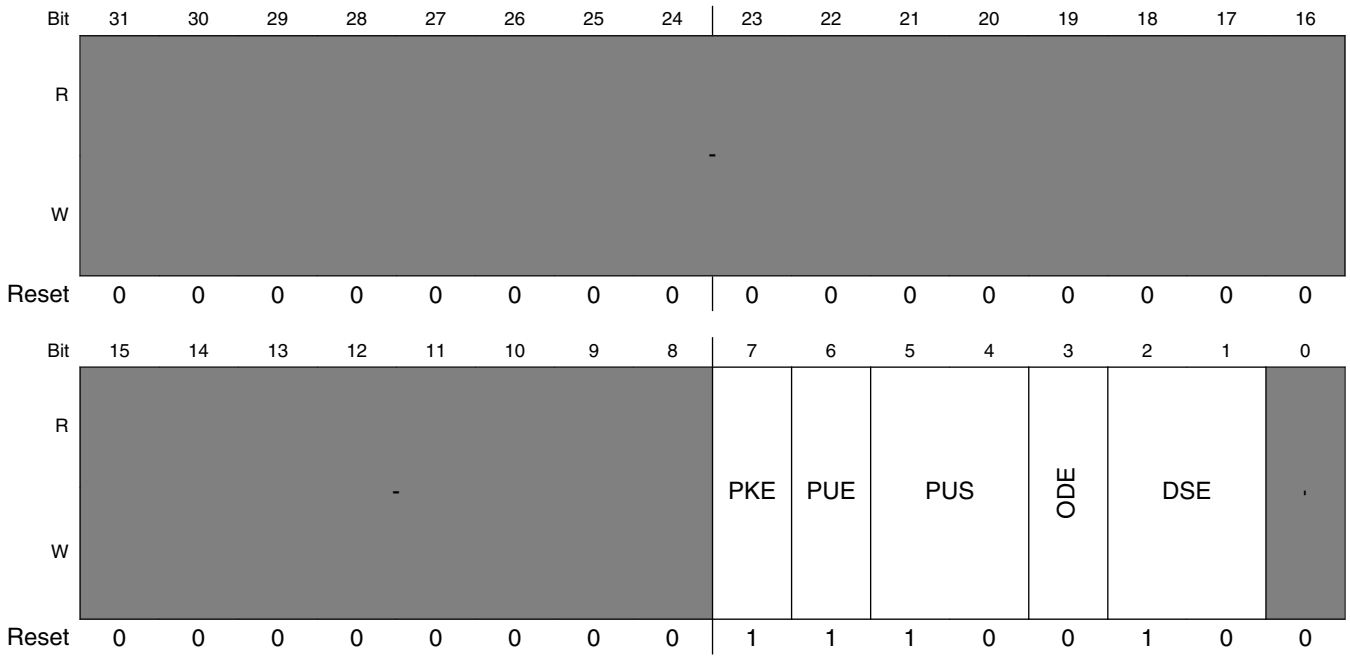
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PDISP_RD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_RD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_RD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_RD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_RD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_RD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.270 SW_PAD_CTL_PAD_DISP_RS (IOMUXC_SPADC_PDISP_RS)

Address: IOMUXC_SPADC_PDISP_RS is 53FA_8000h base + 434h offset = 53FA_8434h



IOMUXC_SPADC_PDISP_RS field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_RS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_RS. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_RS. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PDISP_RS field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_RS. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_RS. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.271 SW_PAD_CTL_PAD_DISP_CS (IOMUXC_SPADC_PDISP_CS)

Address: IOMUXC_SPADC_PDISP_CS is 53FA_8000h base + 438h offset = 53FA_8438h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

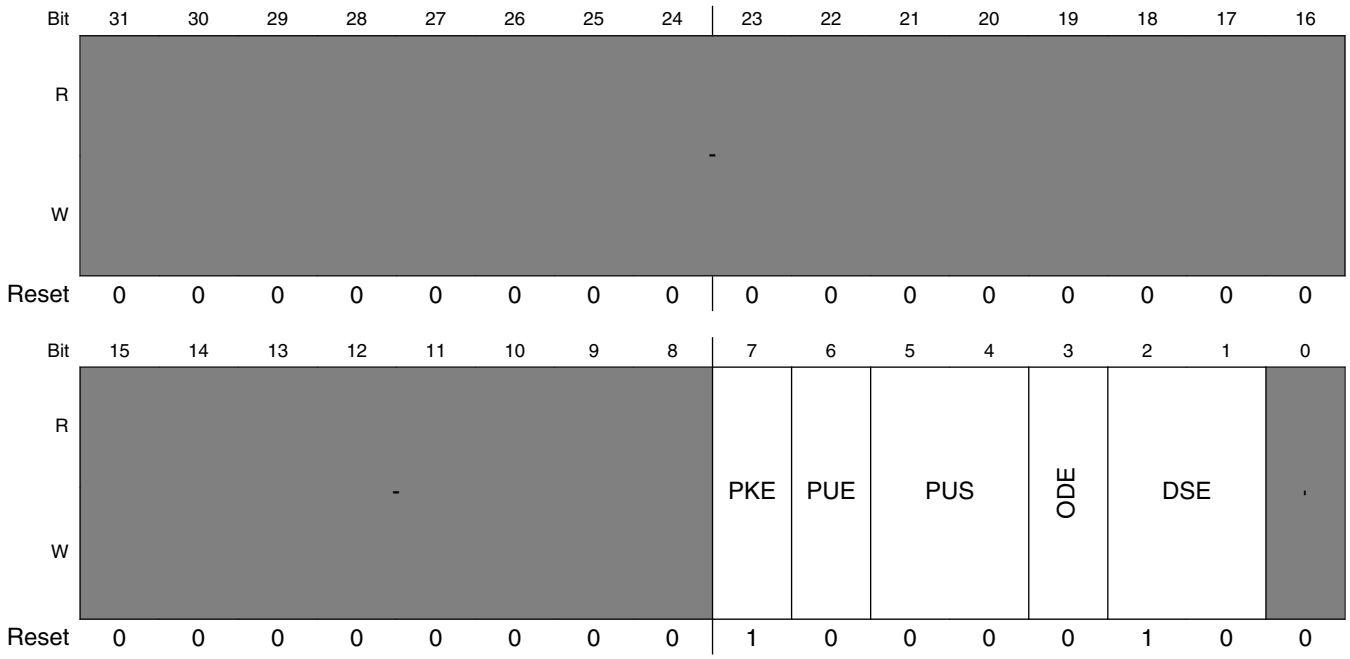
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PDISP_CS field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_CS. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_CS. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_CS. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_CS. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_CS. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.272 SW_PAD_CTL_PAD_DISP_BUSY (IOMUXC_SPADC_PDISP_BUSY)

Address: IOMUXC_SPADC_PDISP_BUSY is 53FA_8000h base + 43Ch offset = 53FA_843Ch



IOMUXC_SPADC_PDISP_BUSY field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_BUSY. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_BUSY. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_BUSY. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PDISP_BUSY field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_BUSY. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_BUSY. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.273 SW_PAD_CTL_PAD_DISP_RESET (IOMUXC_SPADC_PDISP_RESET)

Address: IOMUXC_SPADC_PDISP_RESET is 53FA_8000h base + 440h offset = 53FA_8440h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PDISP_RESET field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_RESET. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_RESET. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_RESET. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_RESET. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_RESET. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.274 SW_PAD_CTL_PAD_SD3_CMD (IOMUXC_SPADC_PSD3_CMD)

Address: IOMUXC_SPADC_PSD3_CMD is 53FA_8000h base + 444h offset = 53FA_8444h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD3_CMD field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_CMD. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_CMD. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_CMD. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PSD3_CMD field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD3_CMD. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_CMD. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.275 SW_PAD_CTL_PAD_SD3_CLK (IOMUXC_SPADC_PSD3_CLK)

Address: IOMUXC_SPADC_PSD3_CLK is 53FA_8000h base + 448h offset = 53FA_8448h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

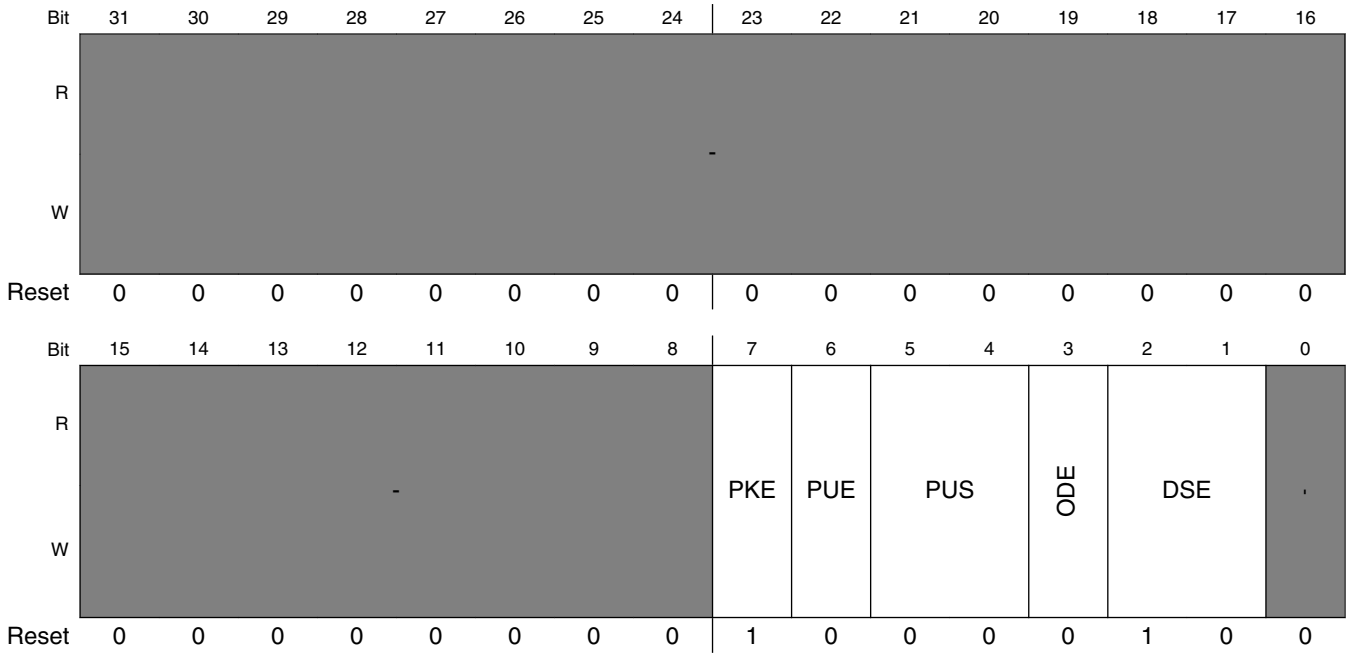
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD3_CLK field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_CLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_CLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_CLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD3_CLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_CLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.276 SW_PAD_CTL_PAD_SD3_D0 (IOMUXC_SPADC_PSD3_D0)

Address: IOMUXC_SPADC_PSD3_D0 is 53FA_8000h base + 44Ch offset = 53FA_844Ch



IOMUXC_SPADC_PSD3_D0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_D0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_D0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_D0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD3_D0 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD3_D0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_D0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.277 SW_PAD_CTL_PAD_SD3_D1 (IOMUXC_SPADC_PSD3_D1)

Address: IOMUXC_SPADC_PSD3_D1 is 53FA_8000h base + 450h offset = 53FA_8450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD3_D1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_D1.

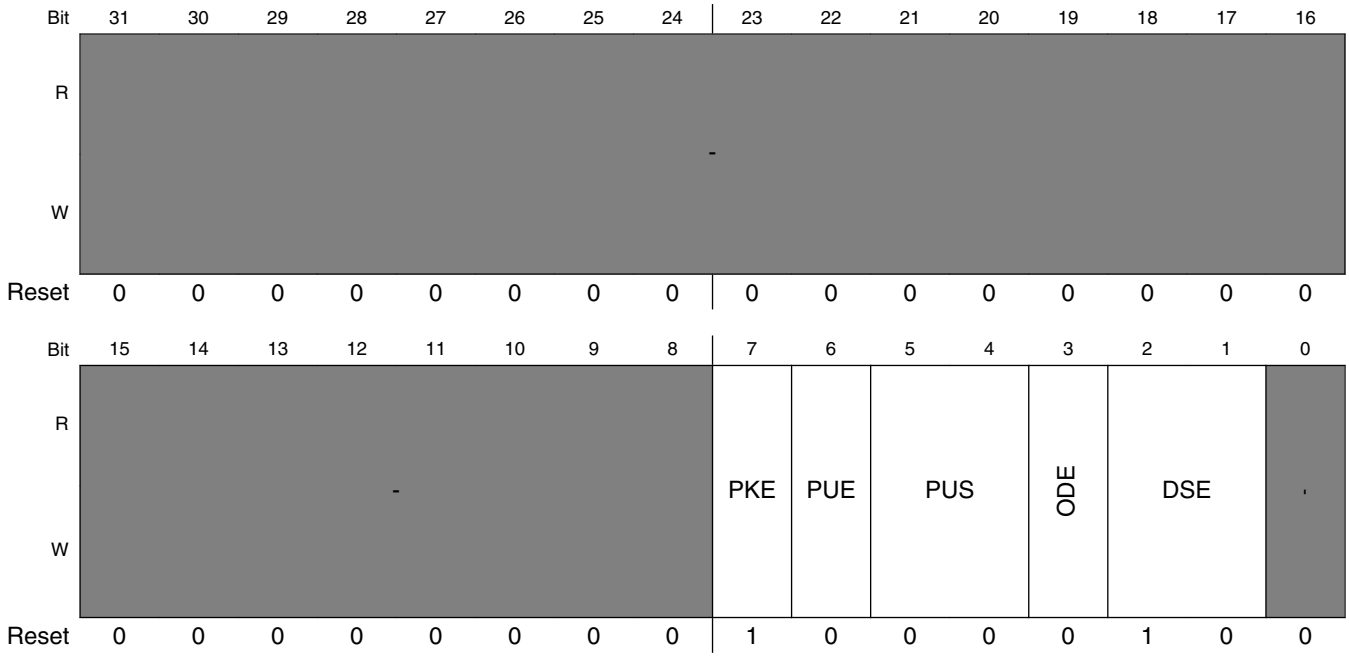
Table continues on the next page...

IOMUXC_SPADC_PSD3_D1 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_D1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_D1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD3_D1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_D1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.278 SW_PAD_CTL_PAD_SD3_D2 (IOMUXC_SPADC_PSD3_D2)

Address: IOMUXC_SPADC_PSD3_D2 is 53FA_8000h base + 454h offset = 53FA_8454h



IOMUXC_SPADC_PSD3_D2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_D2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_D2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_D2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD3_D2 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD3_D2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_D2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.279 SW_PAD_CTL_PAD_SD3_D3 (IOMUXC_SPADC_PSD3_D3)

Address: IOMUXC_SPADC_PSD3_D3 is 53FA_8000h base + 458h offset = 53FA_8458h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD3_D3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_D3.

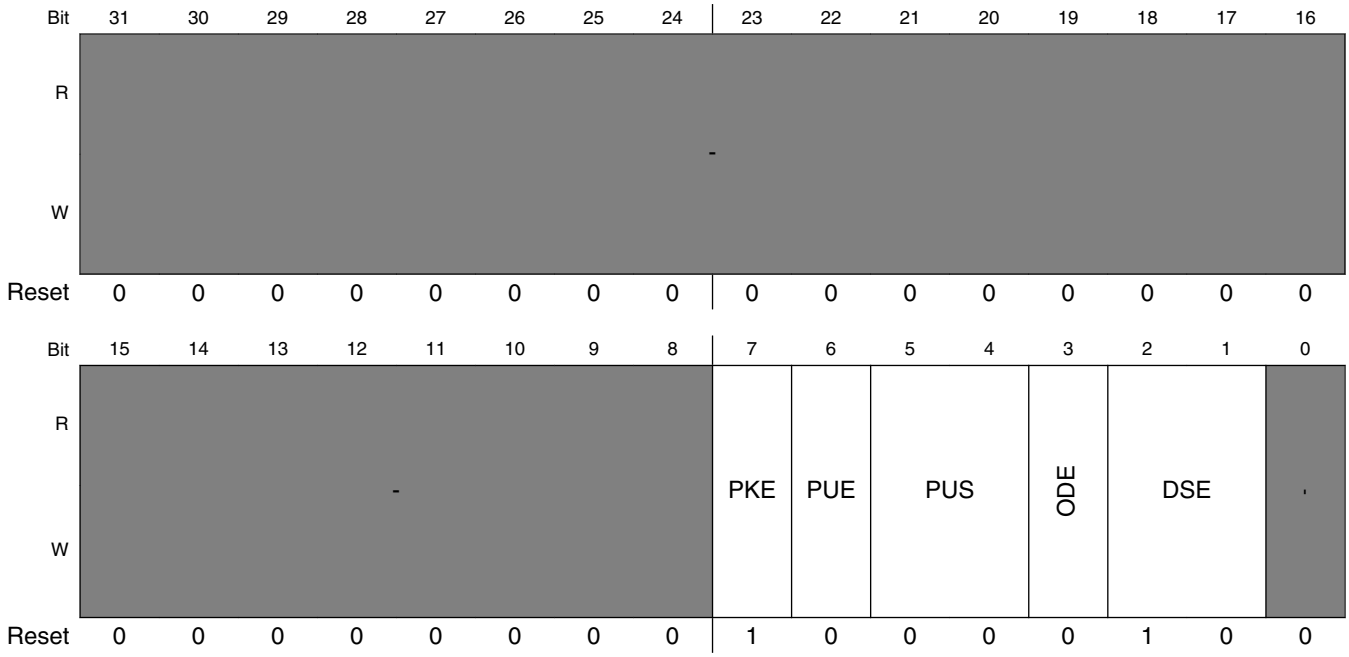
Table continues on the next page...

IOMUXC_SPADC_PSD3_D3 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_D3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_D3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD3_D3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_D3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.280 SW_PAD_CTL_PAD_SD3_D4 (IOMUXC_SPADC_PSD3_D4)

Address: IOMUXC_SPADC_PSD3_D4 is 53FA_8000h base + 45Ch offset = 53FA_845Ch



IOMUXC_SPADC_PSD3_D4 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_D4. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_D4. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_D4. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD3_D4 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD3_D4. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_D4. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.281 SW_PAD_CTL_PAD_SD3_D5 (IOMUXC_SPADC_PSD3_D5)

Address: IOMUXC_SPADC_PSD3_D5 is 53FA_8000h base + 460h offset = 53FA_8460h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD3_D5 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_D5.

Table continues on the next page...

IOMUXC_SPADC_PSD3_D5 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_D5. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_D5. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD3_D5. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_D5. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.282 SW_PAD_CTL_PAD_SD3_D6 (IOMUXC_SPADC_PSD3_D6)

Address: IOMUXC_SPADC_PSD3_D6 is 53FA_8000h base + 464h offset = 53FA_8464h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD3_D6 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_D6. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_D6. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_D6. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD3_D6 field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD3_D6. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_D6. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.283 SW_PAD_CTL_PAD_SD3_D7 (IOMUXC_SPADC_PSD3_D7)

Address: IOMUXC_SPADC_PSD3_D7 is 53FA_8000h base + 468h offset = 53FA_8468h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD3_D7 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_D7.

Table continues on the next page...

IOMUXC_SPADC_PSD3_D7 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_D7. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_D7. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: SD3_D7. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_D7. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.284 SW_PAD_CTL_PAD_SD3_WP (IOMUXC_SPADC_PSD3_WP)

Address: IOMUXC_SPADC_PSD3_WP is 53FA_8000h base + 46Ch offset = 53FA_846Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PSD3_WP field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD3_WP. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD3_WP. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD3_WP. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PSD3_WP field descriptions (continued)

Field	Description
	Select one out of next values for pad: SD3_WP. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: SD3_WP. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.285 SW_PAD_CTL_PAD_DISP_D8 (IOMUXC_SPADC_PDISP_D8)

Address: IOMUXC_SPADC_PDISP_D8 is 53FA_8000h base + 470h offset = 53FA_8470h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PDISP_D8 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D8.

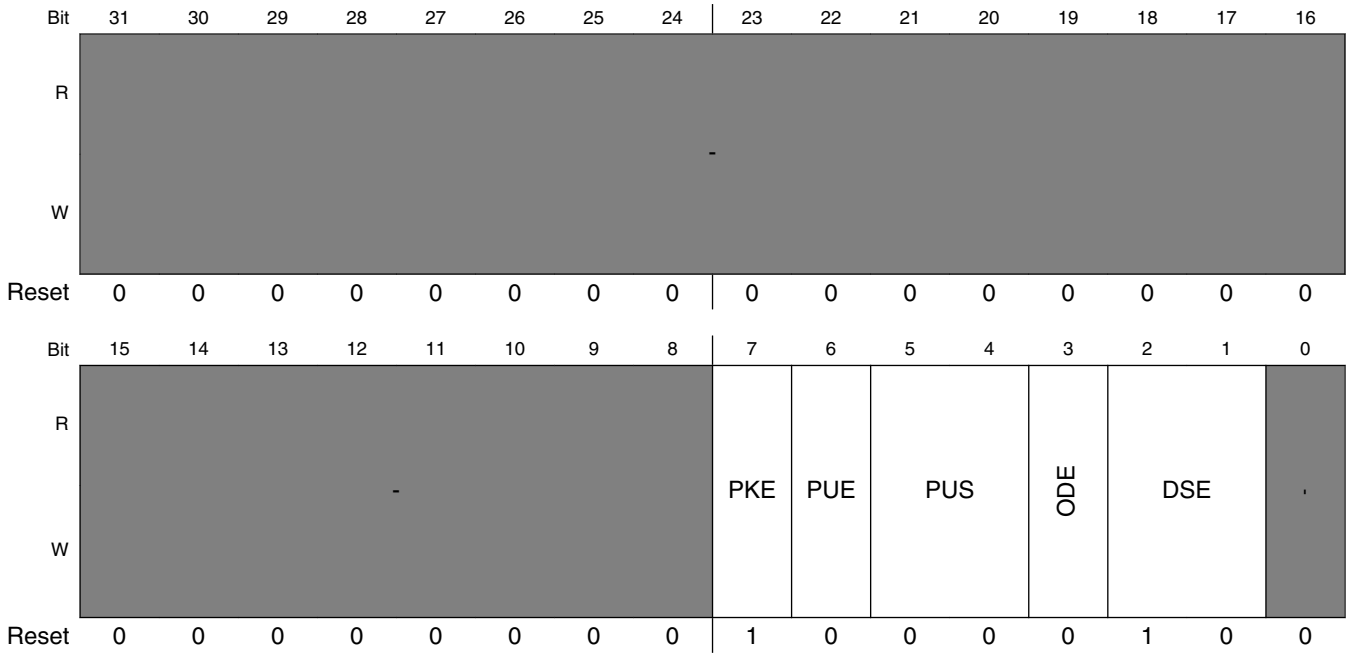
Table continues on the next page...

IOMUXC_SPADC_PDISP_D8 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D8. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D8. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D8. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D8. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.286 SW_PAD_CTL_PAD_DISP_D9 (IOMUXC_SPADC_PDISP_D9)

Address: IOMUXC_SPADC_PDISP_D9 is 53FA_8000h base + 474h offset = 53FA_8474h



IOMUXC_SPADC_PDISP_D9 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D9. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D9. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D9. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PDISP_D9 field descriptions (continued)

Field	Description
	Select one out of next values for pad: DISP_D9. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D9. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.287 SW_PAD_CTL_PAD_DISP_D10 (IOMUXC_SPADC_PDISP_D10)

Address: IOMUXC_SPADC_PDISP_D10 is 53FA_8000h base + 478h offset = 53FA_8478h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PDISP_D10 field descriptions

Field	Description
31–8 -	Reserved

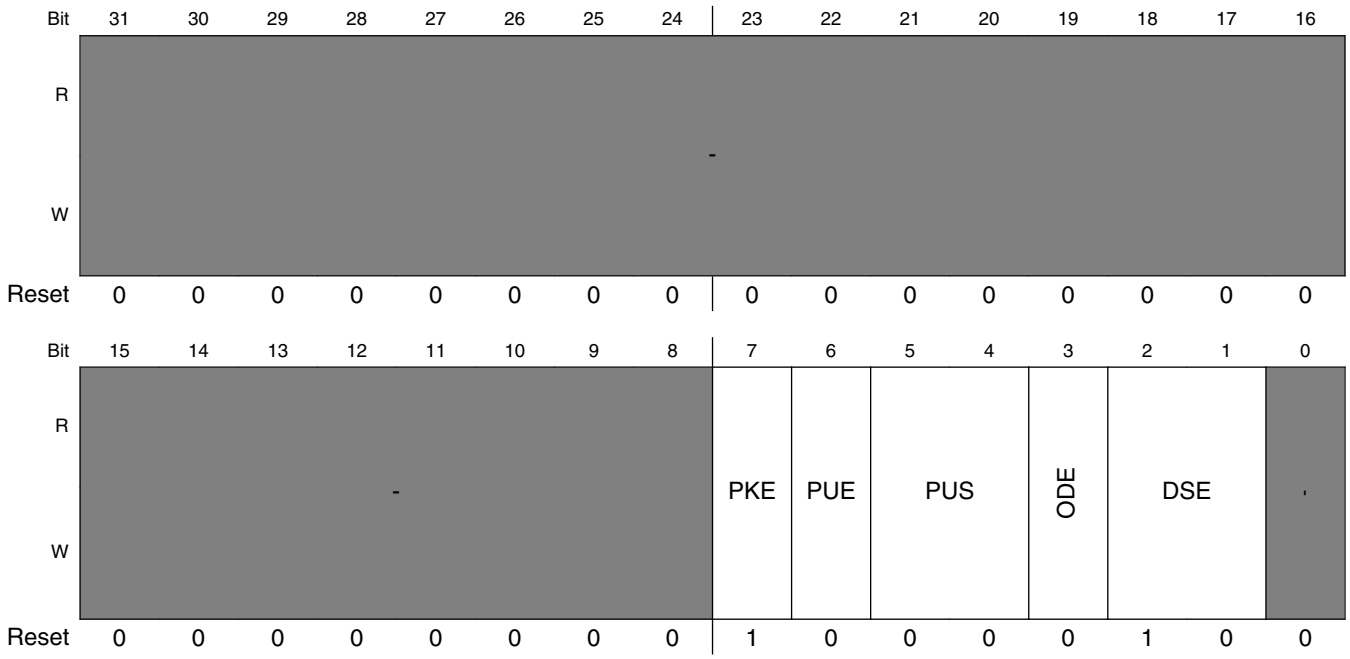
Table continues on the next page...

IOMUXC_SPADC_PDISP_D10 field descriptions (continued)

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D10. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D10. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D10. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D10. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D10. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.288 SW_PAD_CTL_PAD_DISP_D11 (IOMUXC_SPADC_PDISP_D11)

Address: IOMUXC_SPADC_PDISP_D11 is 53FA_8000h base + 47Ch offset = 53FA_847Ch



IOMUXC_SPADC_PDISP_D11 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D11. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D11. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D11. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

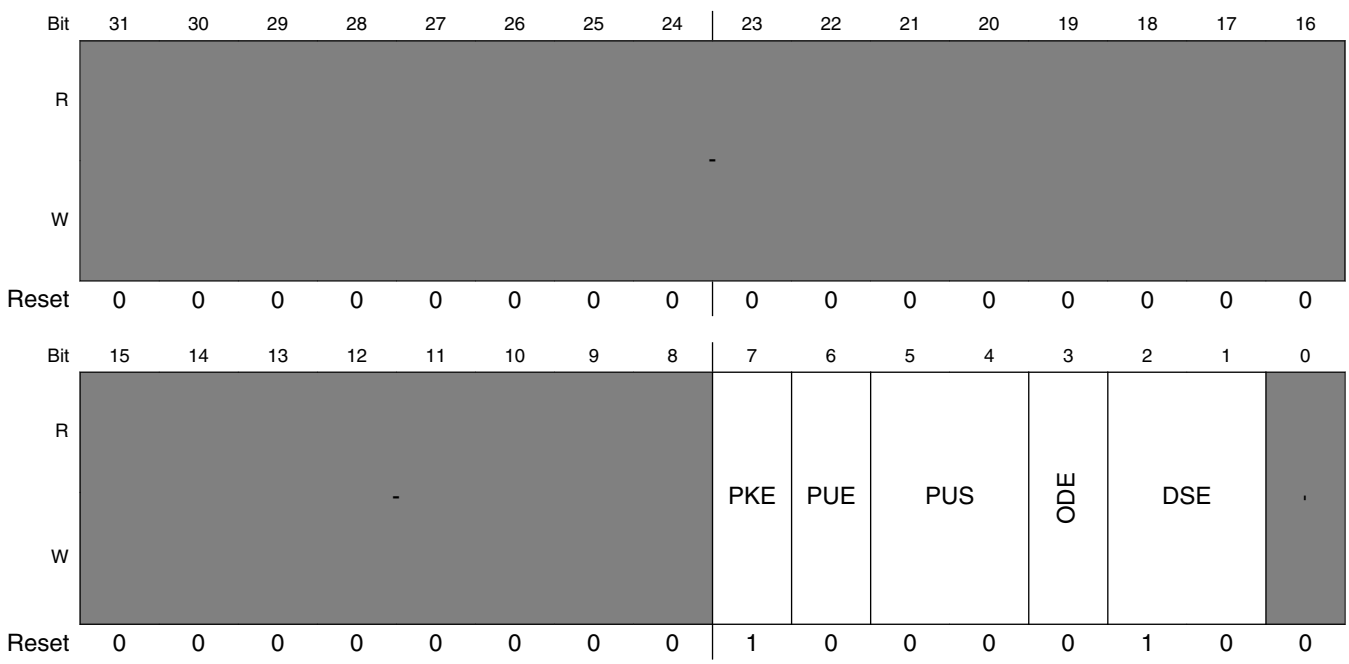
Table continues on the next page...

IOMUXC_SPADC_PDISP_D11 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D11. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D11. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.289 SW_PAD_CTL_PAD_DISP_D12 (IOMUXC_SPADC_PDISP_D12)

Address: IOMUXC_SPADC_PDISP_D12 is 53FA_8000h base + 480h offset = 53FA_8480h

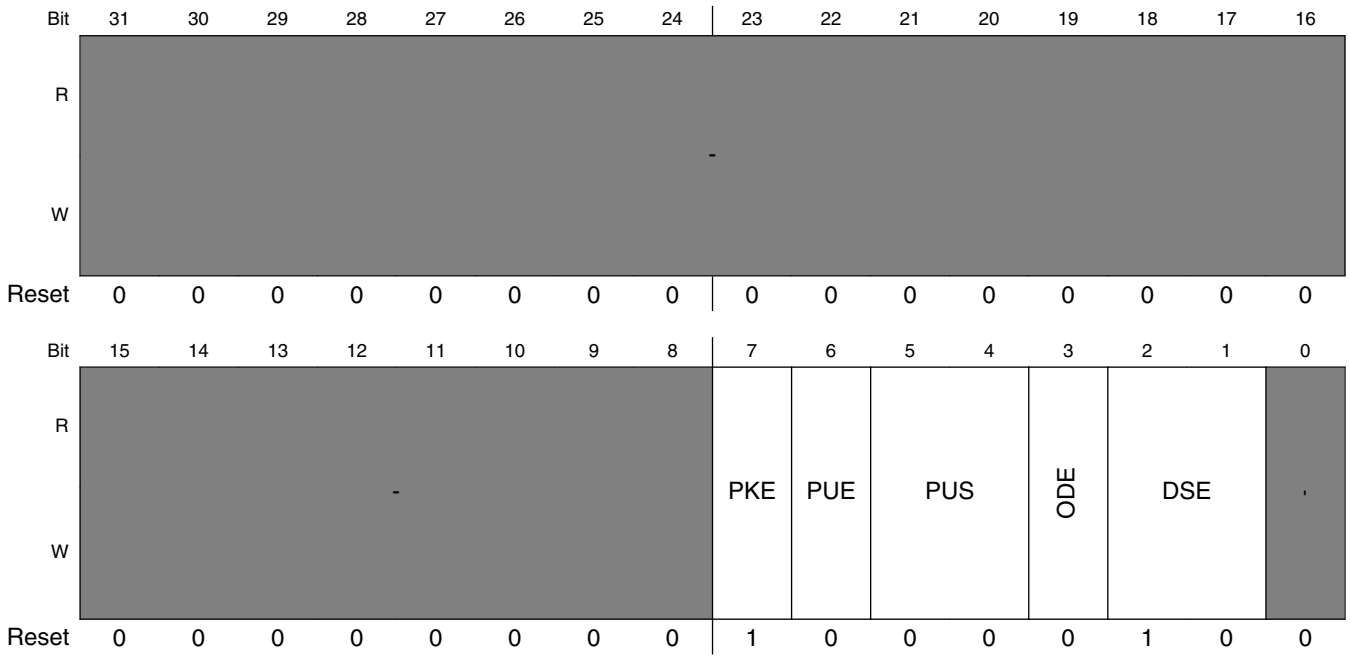


IOMUXC_SPADC_PDISP_D12 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D12. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D12. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D12. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D12. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D12. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.290 SW_PAD_CTL_PAD_DISP_D13 (IOMUXC_SPADC_PDISP_D13)

Address: IOMUXC_SPADC_PDISP_D13 is 53FA_8000h base + 484h offset = 53FA_8484h



IOMUXC_SPADC_PDISP_D13 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D13. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D13. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D13. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PDISP_D13 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D13. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D13. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.291 SW_PAD_CTL_PAD_DISP_D14 (IOMUXC_SPADC_PDISP_D14)

Address: IOMUXC_SPADC_PDISP_D14 is 53FA_8000h base + 488h offset = 53FA_8488h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

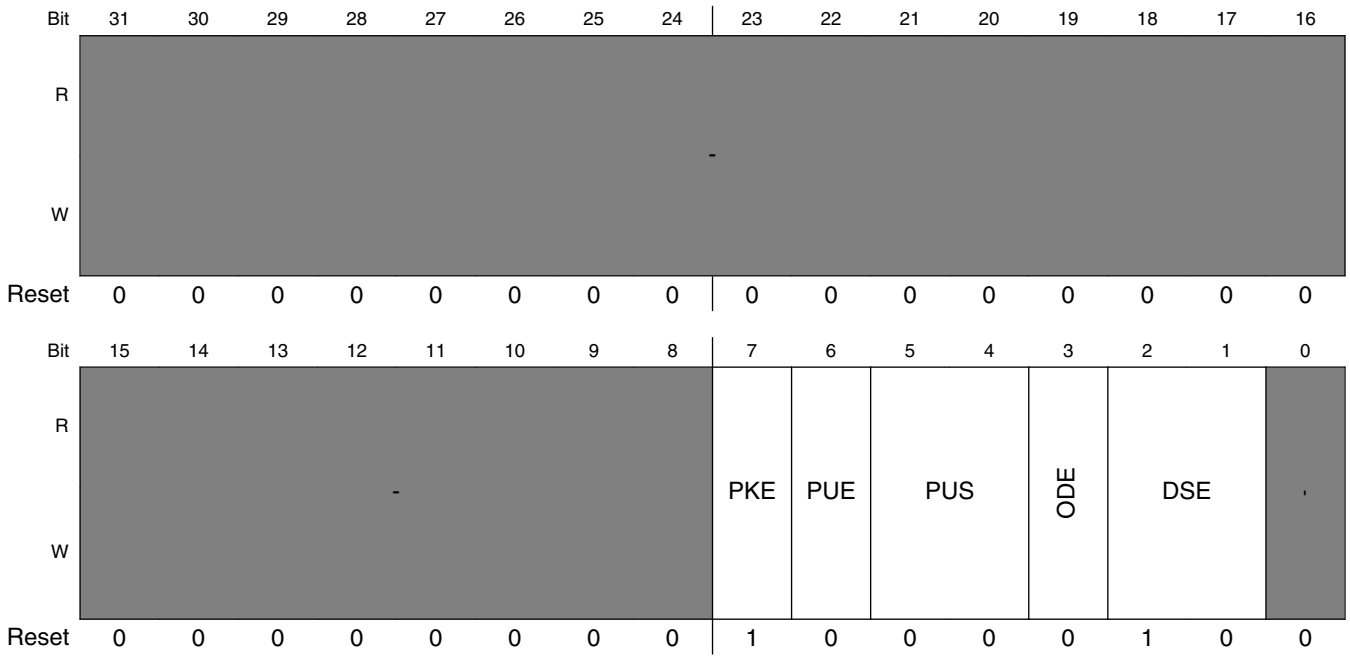
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PDISP_D14 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D14. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D14. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D14. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D14. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D14. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.292 SW_PAD_CTL_PAD_DISP_D15 (IOMUXC_SPADC_PDISP_D15)

Address: IOMUXC_SPADC_PDISP_D15 is 53FA_8000h base + 48Ch offset = 53FA_848Ch



IOMUXC_SPADC_PDISP_D15 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DISP_D15. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DISP_D15. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: DISP_D15. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

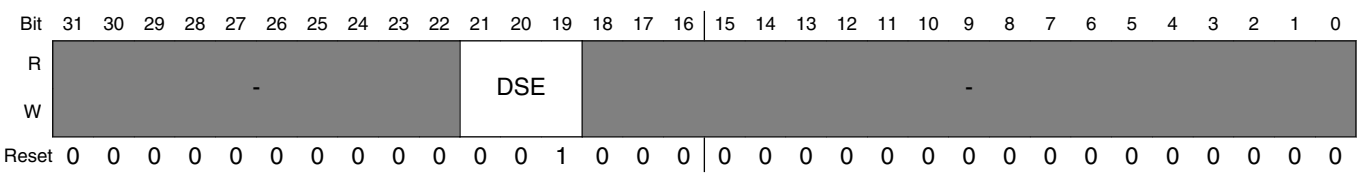
Table continues on the next page...

IOMUXC_SPADC_PDISP_D15 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: DISP_D15. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: DISP_D15. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.293 SW_PAD_CTL_PAD_DRAM_OPEN (IOMUXC_SPADC_PDRAM_OPEN)

Address: IOMUXC_SPADC_PDRAM_OPEN is 53FA_8000h base + 490h offset = 53FA_8490h

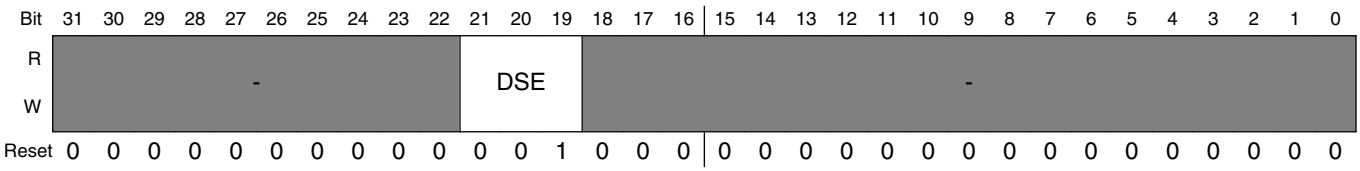


IOMUXC_SPADC_PDRAM_OPEN field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_OPEN. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.294 SW_PAD_CTL_PAD_DRAM_OPENFB (IOMUXC_SPADC_PDRAM_OPENFB)

Address: IOMUXC_SPADC_PDRAM_OPENFB is 53FA_8000h base + 494h offset = 53FA_8494h

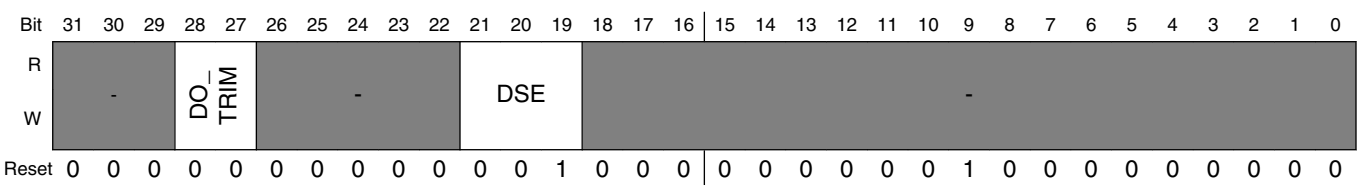


IOMUXC_SPADC_PDRAM_OPENFB field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_OPENFB. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.295 SW_PAD_CTL_PAD_DRAM_SDCLK_1 (IOMUXC_SPADC_PDRAM_SDCLK_1)

Address: IOMUXC_SPADC_PDRAM_SDCLK_1 is 53FA_8000h base + 498h offset = 53FA_8498h



IOMUXC_SPADC_PDRAM_SDCLK_1 field descriptions

Field	Description
31–29 -	Reserved

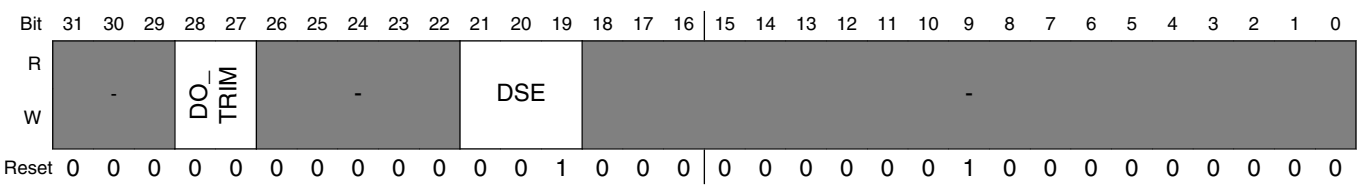
Table continues on the next page...

IOMUXC_SPADC_PDRAM_SDCLK_1 field descriptions (continued)

Field	Description
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDCLK_1. 00 min 01 +50ps 10 +100ps 11 +150ps
26–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDCLK_1. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.296 SW_PAD_CTL_PAD_DRAM_SDCLK_0 (IOMUXC_SPADC_PDRAM_SDCLK_0)

Address: IOMUXC_SPADC_PDRAM_SDCLK_0 is 53FA_8000h base + 49Ch offset = 53FA_849Ch



IOMUXC_SPADC_PDRAM_SDCLK_0 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDCLK_0. 00 min 01 +50ps

Table continues on the next page...

IOMUXC_SPADC_PDRAM_SDCLK_0 field descriptions (continued)

Field	Description
	10 +100ps 11 +150ps
26–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDCLK_0. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.297 SW_PAD_CTL_PAD_DRAM_SDCKE (IOMUXC_SPADC_PDRAM_SDCKE)

Address: IOMUXC_SPADC_PDRAM_SDCKE is 53FA_8000h base + 4A0h offset = 53FA_84A0h

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

IOMUXC_SPADC_PDRAM_SDCKE field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDCKE. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDCKE.

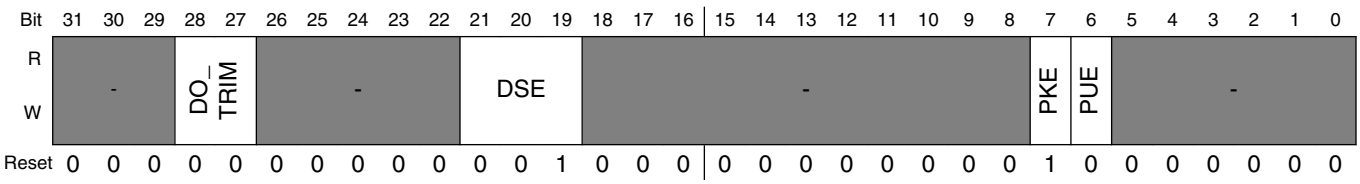
Table continues on the next page...

IOMUXC_SPADC_PDRAM_SDCKE field descriptions (continued)

Field	Description
	0 Keeper 1 Pull
5–0 -	Reserved

35.4.298 SW_PAD_CTL_PAD_DRAM_SDODT0 (IOMUXC_SPADC_PDRAM_SDODT0)

Address: IOMUXC_SPADC_PDRAM_SDODT0 is 53FA_8000h base + 4A4h offset = 53FA_84A4h



IOMUXC_SPADC_PDRAM_SDODT0 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDODT0. 00 min 01 +50ps 10 +100ps 11 +150ps
26–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDODT0. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–8 -	Reserved

Table continues on the next page...

IOMUXC_SPADC_PDRAM_SDODT0 field descriptions (continued)

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDODT0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDODT0. 0 Keeper 1 Pull
5–0 -	Reserved

35.4.299 SW_PAD_CTL_PAD_DRAM_D16 (IOMUXC_SPADC_PDRAM_D16)

Address: IOMUXC_SPADC_PDRAM_D16 is 53FA_8000h base + 4A8h offset = 53FA_84A8h

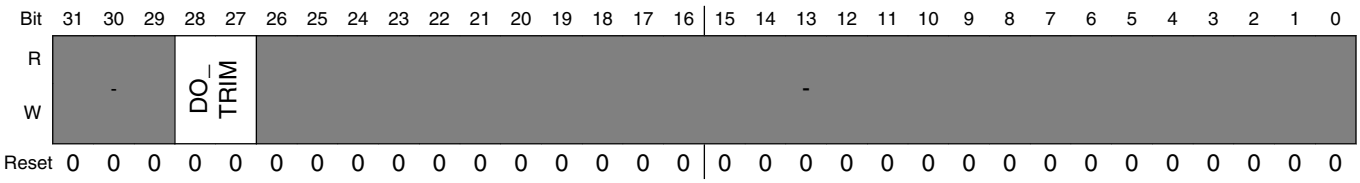
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-			DO_TRIM		-																										
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SPADC_PDRAM_D16 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D16. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.300 SW_PAD_CTL_PAD_DRAM_D17 (IOMUXC_SPADC_PDRAM_D17)

Address: IOMUXC_SPADC_PDRAM_D17 is 53FA_8000h base + 4ACh offset = 53FA_84ACh

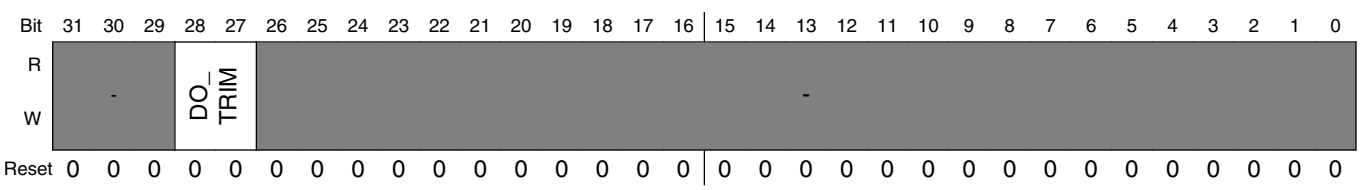


IOMUXC_SPADC_PDRAM_D17 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D17. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.301 SW_PAD_CTL_PAD_DRAM_D18 (IOMUXC_SPADC_PDRAM_D18)

Address: IOMUXC_SPADC_PDRAM_D18 is 53FA_8000h base + 4B0h offset = 53FA_84B0h



IOMUXC_SPADC_PDRAM_D18 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D18.

Table continues on the next page...

IOMUXC_SPADC_PDRAM_D18 field descriptions (continued)

Field	Description
	00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.302 SW_PAD_CTL_PAD_DRAM_D19 (IOMUXC_SPADC_PDRAM_D19)

Address: IOMUXC_SPADC_PDRAM_D19 is 53FA_8000h base + 4B4h offset = 53FA_84B4h

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

IOMUXC_SPADC_PDRAM_D19 field descriptions

Field	Description
31–29 -	RESERVED
28–27 DO_TRIM	DO_TRIM FIELD SELECT ONE OUT OF NEXT VALUES FOR PAD: DRAM_D19. 00 MIN 01 +50PS 10 +100PS 11 +150PS
26–0 -	RESERVED

35.4.303 SW_PAD_CTL_PAD_DRAM_D20 (IOMUXC_SPADC_PDRAM_D20)

Address: IOMUXC_SPADC_PDRAM_D20 is 53FA_8000h base + 4B8h offset = 53FA_84B8h

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

IOMUXC_SPADC_PDRAM_D20 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D20. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.304 SW_PAD_CTL_PAD_DRAM_D21 (IOMUXC_SPADC_PDRAM_D21)

Address: IOMUXC_SPADC_PDRAM_D21 is 53FA_8000h base + 4BCh offset = 53FA_84BCh

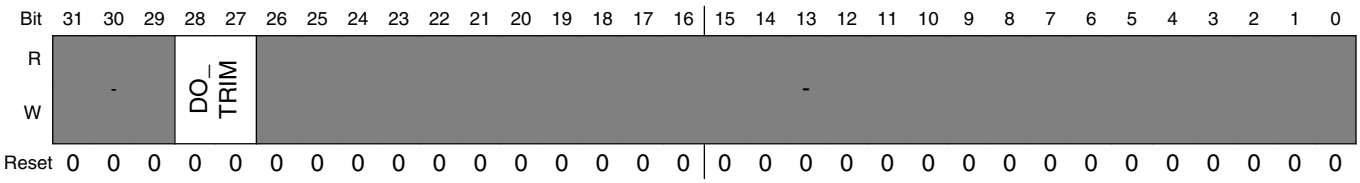
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-			DO_TRIM		-											-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SPADC_PDRAM_D21 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D21. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.305 SW_PAD_CTL_PAD_DRAM_D22 (IOMUXC_SPADC_PDRAM_D22)

Address: IOMUXC_SPADC_PDRAM_D22 is 53FA_8000h base + 4C0h offset = 53FA_84C0h

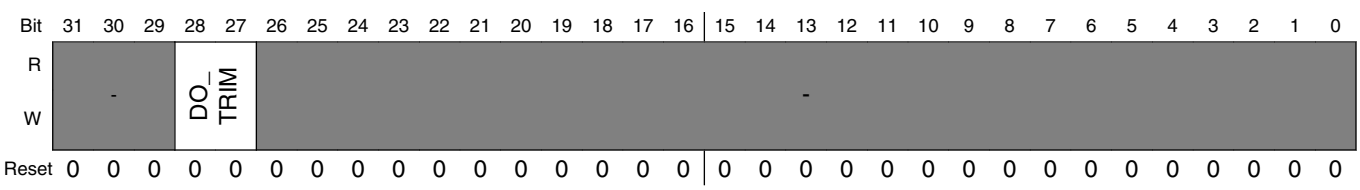


IOMUXC_SPADC_PDRAM_D22 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D22. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.306 SW_PAD_CTL_PAD_DRAM_D23 (IOMUXC_SPADC_PDRAM_D23)

Address: IOMUXC_SPADC_PDRAM_D23 is 53FA_8000h base + 4C4h offset = 53FA_84C4h



IOMUXC_SPADC_PDRAM_D23 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D23.

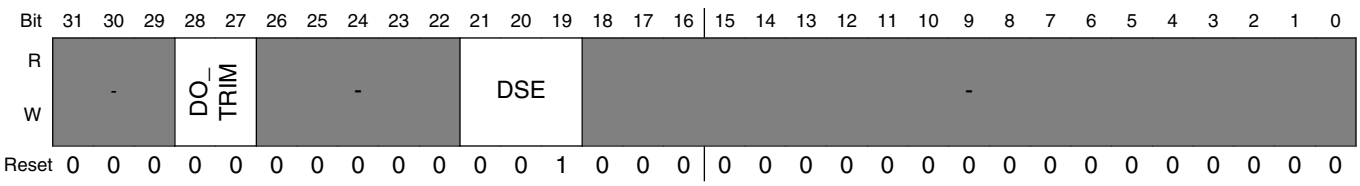
Table continues on the next page...

IOMUXC_SPADC_PDRAM_D23 field descriptions (continued)

Field	Description
	00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.307 SW_PAD_CTL_PAD_DRAM_DQM2 (IOMUXC_SPADC_PDRAM_DQM2)

Address: IOMUXC_SPADC_PDRAM_DQM2 is 53FA_8000h base + 4C8h offset = 53FA_84C8h



IOMUXC_SPADC_PDRAM_DQM2 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_DQM2. 00 min 01 +50ps 10 +100ps 11 +150ps
26–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_DQM2. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:

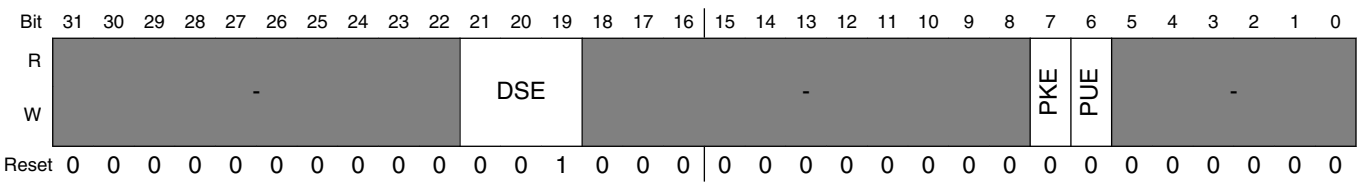
Table continues on the next page...

IOMUXC_SPADC_PDRAM_DQM2 field descriptions (continued)

Field	Description
18–0 -	Reserved

35.4.308 SW_PAD_CTL_PAD_DRAM_SDQS2 (IOMUXC_SPADC_PDRAM_SDQS2)

Address: IOMUXC_SPADC_PDRAM_SDQS2 is 53FA_8000h base + 4CCh offset = 53FA_84CCh

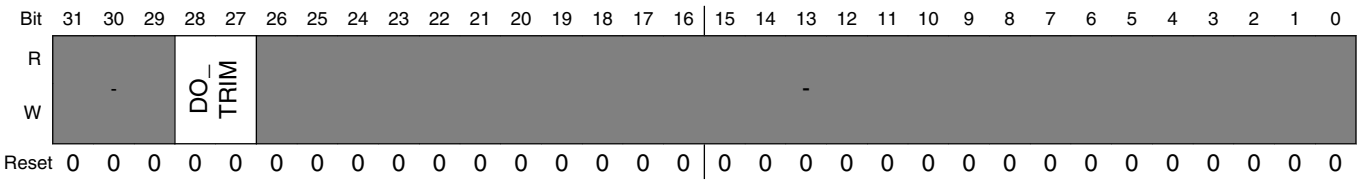


IOMUXC_SPADC_PDRAM_SDQS2 field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDQS2. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDQS2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDQS2. 0 Keeper 1 Pull
5–0 -	Reserved

35.4.309 SW_PAD_CTL_PAD_DRAM_D0 (IOMUXC_SPADC_PDRAM_D0)

Address: IOMUXC_SPADC_PDRAM_D0 is 53FA_8000h base + 4D0h offset = 53FA_84D0h

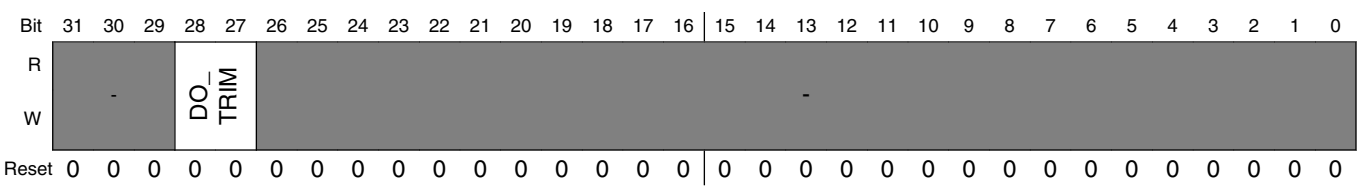


IOMUXC_SPADC_PDRAM_D0 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D0. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.310 SW_PAD_CTL_PAD_DRAM_D1 (IOMUXC_SPADC_PDRAM_D1)

Address: IOMUXC_SPADC_PDRAM_D1 is 53FA_8000h base + 4D4h offset = 53FA_84D4h



IOMUXC_SPADC_PDRAM_D1 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D1.

Table continues on the next page...

IOMUXC_SPADC_PDRAM_D1 field descriptions (continued)

Field	Description
	00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.311 SW_PAD_CTL_PAD_DRAM_D2 (IOMUXC_SPADC_PDRAM_D2)

Address: IOMUXC_SPADC_PDRAM_D2 is 53FA_8000h base + 4D8h offset = 53FA_84D8h

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

IOMUXC_SPADC_PDRAM_D2 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D2. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.312 SW_PAD_CTL_PAD_DRAM_D3 (IOMUXC_SPADC_PDRAM_D3)

Address: IOMUXC_SPADC_PDRAM_D3 is 53FA_8000h base + 4DCh offset = 53FA_84DCh

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

IOMUXC_SPADC_PDRAM_D3 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D3. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.313 SW_PAD_CTL_PAD_DRAM_D4 (IOMUXC_SPADC_PDRAM_D4)

Address: IOMUXC_SPADC_PDRAM_D4 is 53FA_8000h base + 4E0h offset = 53FA_84E0h

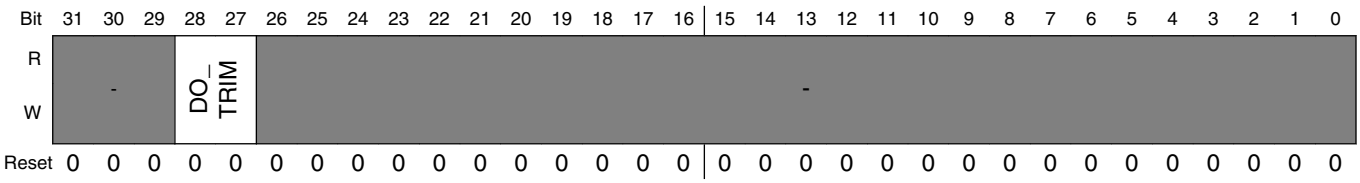
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-			DO_TRIM		-											-															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SPADC_PDRAM_D4 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D4. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.314 SW_PAD_CTL_PAD_DRAM_D5 (IOMUXC_SPADC_PDRAM_D5)

Address: IOMUXC_SPADC_PDRAM_D5 is 53FA_8000h base + 4E4h offset = 53FA_84E4h

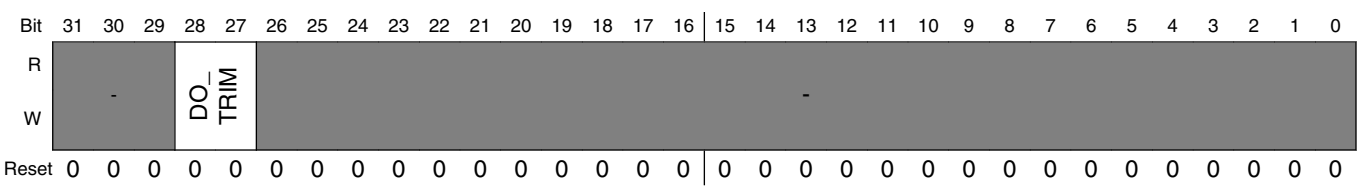


IOMUXC_SPADC_PDRAM_D5 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D5. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.315 SW_PAD_CTL_PAD_DRAM_D6 (IOMUXC_SPADC_PDRAM_D6)

Address: IOMUXC_SPADC_PDRAM_D6 is 53FA_8000h base + 4E8h offset = 53FA_84E8h



IOMUXC_SPADC_PDRAM_D6 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D6.

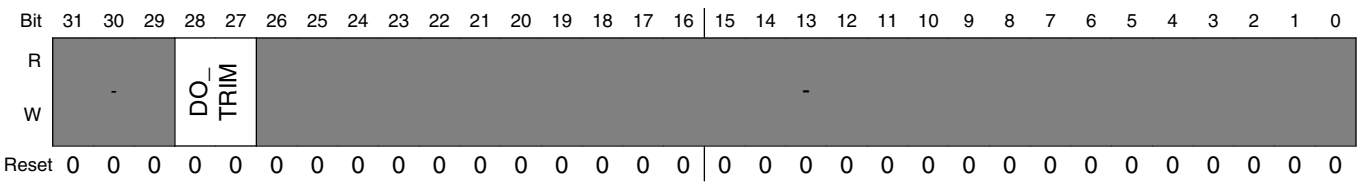
Table continues on the next page...

IOMUXC_SPADC_PDRAM_D6 field descriptions (continued)

Field	Description
	00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.316 SW_PAD_CTL_PAD_DRAM_D7 (IOMUXC_SPADC_PDRAM_D7)

Address: IOMUXC_SPADC_PDRAM_D7 is 53FA_8000h base + 4ECh offset = 53FA_84ECh

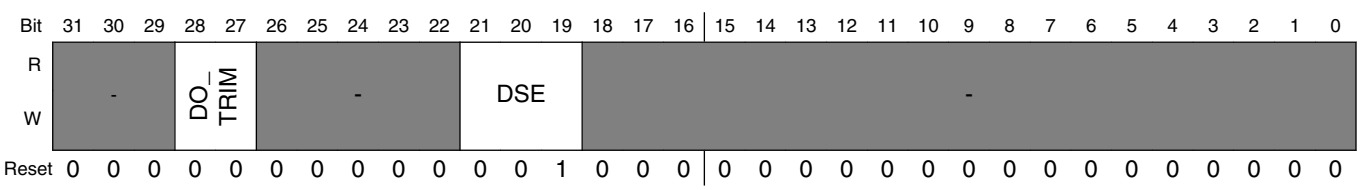


IOMUXC_SPADC_PDRAM_D7 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D7. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.317 SW_PAD_CTL_PAD_DRAM_DQM0 (IOMUXC_SPADC_PDRAM_DQM0)

Address: IOMUXC_SPADC_PDRAM_DQM0 is 53FA_8000h base + 4F0h offset = 53FA_84F0h

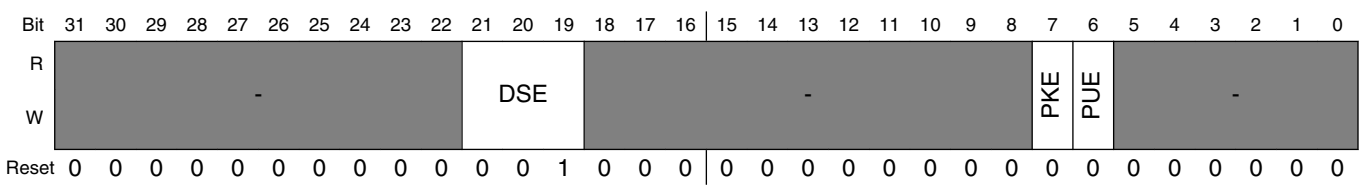


IOMUXC_SPADC_PDRAM_DQM0 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_DQM0. 00 min 01 +50ps 10 +100ps 11 +150ps
26–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_DQM0. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.318 SW_PAD_CTL_PAD_DRAM_SDQS0 (IOMUXC_SPADC_PDRAM_SDQS0)

Address: IOMUXC_SPADC_PDRAM_SDQS0 is 53FA_8000h base + 4F4h offset = 53FA_84F4h



IOMUXC_SPADC_PDRAM_SDQS0 field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDQS0.

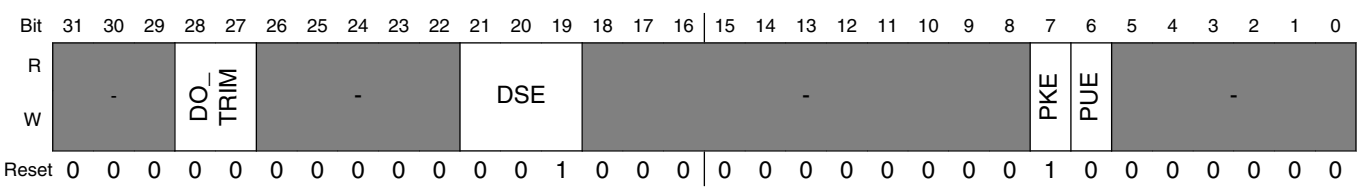
Table continues on the next page...

IOMUXC_SPADC_PDRAM_SDQS0 field descriptions (continued)

Field	Description
	000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDQS0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDQS0. 0 Keeper 1 Pull
5–0 -	Reserved

35.4.319 SW_PAD_CTL_PAD_DRAM_SDODT1 (IOMUXC_SPADC_PDRAM_SDODT1)

Address: IOMUXC_SPADC_PDRAM_SDODT1 is 53FA_8000h base + 4F8h offset = 53FA_84F8h



IOMUXC_SPADC_PDRAM_SDODT1 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_SDODT1. 00 min 01 +50ps

Table continues on the next page...

IOMUXC_SPADC_PDRAM_SDODT1 field descriptions (continued)

Field	Description
	10 +100ps 11 +150ps
26–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDODT1. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDODT1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDODT1. 0 Keeper 1 Pull
5–0 -	Reserved

35.4.320 SW_PAD_CTL_PAD_DRAM_SDQS1 (IOMUXC_SPADC_PDRAM_SDQS1)

Address: IOMUXC_SPADC_PDRAM_SDQS1 is 53FA_8000h base + 4FCh offset = 53FA_84FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-										DSE			-											PKE	PUE	-					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SPADC_PDRAM_SDQS1 field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDQS1. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDQS1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDQS1. 0 Keeper 1 Pull
5–0 -	Reserved

35.4.321 SW_PAD_CTL_PAD_DRAM_DQM1 (IOMUXC_SPADC_PDRAM_DQM1)

Address: IOMUXC_SPADC_PDRAM_DQM1 is 53FA_8000h base + 500h offset = 53FA_8500h

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

IOMUXC_SPADC_PDRAM_DQM1 field descriptions

Field	Description
31–29 -	Reserved

Table continues on the next page...

IOMUXC_SPADC_PDRAM_DQM1 field descriptions (continued)

Field	Description
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_DQM1. 00 min 01 +50ps 10 +100ps 11 +150ps
26–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_DQM1. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.322 SW_PAD_CTL_PAD_DRAM_D8 (IOMUXC_SPADC_PDRAM_D8)

Address: IOMUXC_SPADC_PDRAM_D8 is 53FA_8000h base + 504h offset = 53FA_8504h

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

IOMUXC_SPADC_PDRAM_D8 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D8. 00 min 01 +50ps

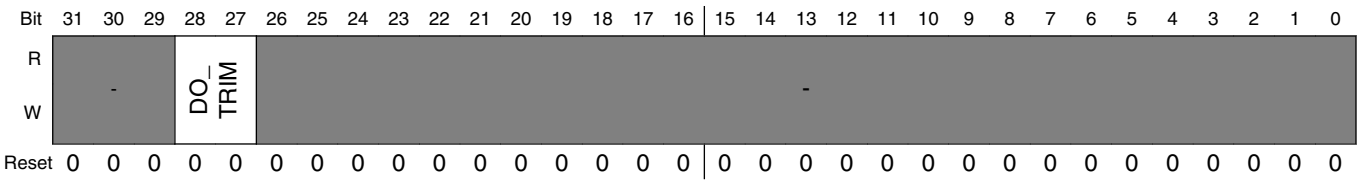
Table continues on the next page...

IOMUXC_SPADC_PDRAM_D8 field descriptions (continued)

Field	Description
	10 +100ps 11 +150ps
26–0 -	Reserved

35.4.323 SW_PAD_CTL_PAD_DRAM_D9 (IOMUXC_SPADC_PDRAM_D9)

Address: IOMUXC_SPADC_PDRAM_D9 is 53FA_8000h base + 508h offset = 53FA_8508h

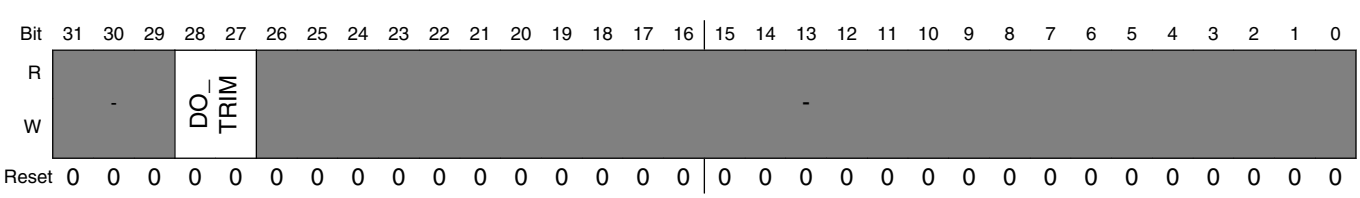


IOMUXC_SPADC_PDRAM_D9 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D9. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.324 SW_PAD_CTL_PAD_DRAM_D10 (IOMUXC_SPADC_PDRAM_D10)

Address: IOMUXC_SPADC_PDRAM_D10 is 53FA_8000h base + 50Ch offset = 53FA_850Ch



IOMUXC_SPADC_PDRAM_D10 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D10. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.325 SW_PAD_CTL_PAD_DRAM_D11 (IOMUXC_SPADC_PDRAM_D11)

Address: IOMUXC_SPADC_PDRAM_D11 is 53FA_8000h base + 510h offset = 53FA_8510h

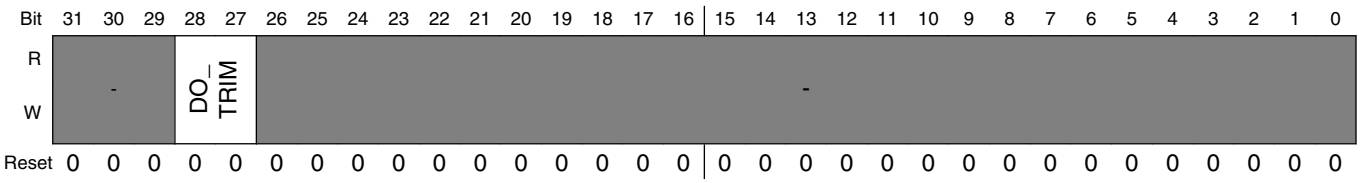
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-			DO_TRIM		-											-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SPADC_PDRAM_D11 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D11. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.326 SW_PAD_CTL_PAD_DRAM_D12 (IOMUXC_SPADC_PDRAM_D12)

Address: IOMUXC_SPADC_PDRAM_D12 is 53FA_8000h base + 514h offset = 53FA_8514h

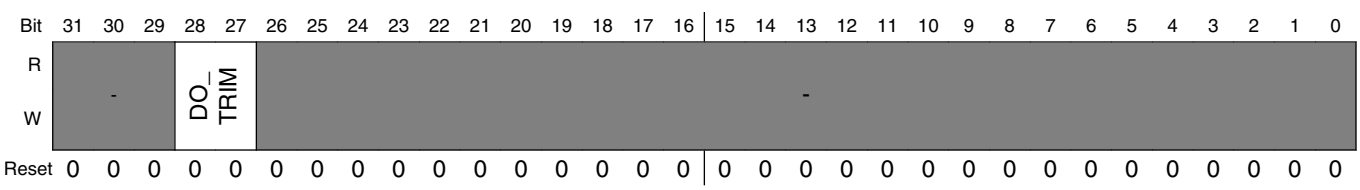


IOMUXC_SPADC_PDRAM_D12 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D12. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.327 SW_PAD_CTL_PAD_DRAM_D13 (IOMUXC_SPADC_PDRAM_D13)

Address: IOMUXC_SPADC_PDRAM_D13 is 53FA_8000h base + 518h offset = 53FA_8518h



IOMUXC_SPADC_PDRAM_D13 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D13.

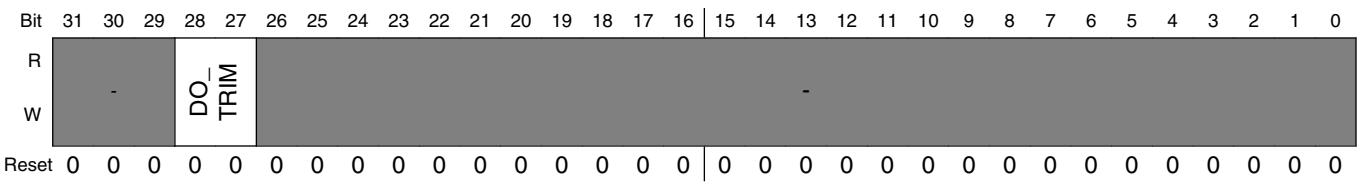
Table continues on the next page...

IOMUXC_SPADC_PDRAM_D13 field descriptions (continued)

Field	Description
	00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.328 SW_PAD_CTL_PAD_DRAM_D14 (IOMUXC_SPADC_PDRAM_D14)

Address: IOMUXC_SPADC_PDRAM_D14 is 53FA_8000h base + 51Ch offset = 53FA_851Ch

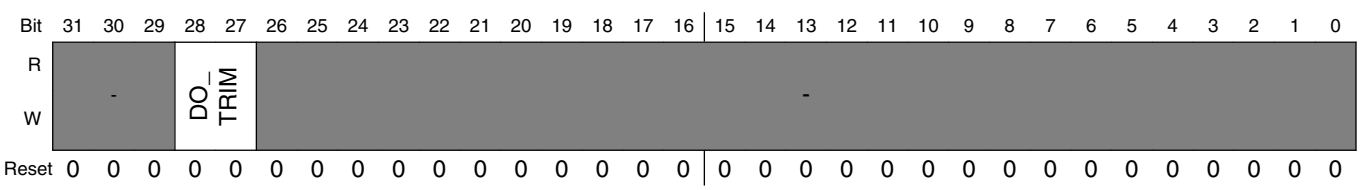


IOMUXC_SPADC_PDRAM_D14 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D14. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.329 SW_PAD_CTL_PAD_DRAM_D15 (IOMUXC_SPADC_PDRAM_D15)

Address: IOMUXC_SPADC_PDRAM_D15 is 53FA_8000h base + 520h offset = 53FA_8520h



IOMUXC_SPADC_PDRAM_D15 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D15. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.330 SW_PAD_CTL_PAD_DRAM_SDQS3 (IOMUXC_SPADC_PDRAM_SDQS3)

Address: IOMUXC_SPADC_PDRAM_SDQS3 is 53FA_8000h base + 524h offset = 53FA_8524h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-										DSE			-											PKE	PUE	-						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SPADC_PDRAM_SDQS3 field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_SDQS3. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: DRAM_SDQS3.

Table continues on the next page...

IOMUXC_SPADC_PDRAM_SDQS3 field descriptions (continued)

Field	Description
	0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: DRAM_SDQS3. 0 Keeper 1 Pull
5–0 -	Reserved

35.4.331 SW_PAD_CTL_PAD_DRAM_DQM3 (IOMUXC_SPADC_PDRAM_DQM3)

Address: IOMUXC_SPADC_PDRAM_DQM3 is 53FA_8000h base + 528h offset = 53FA_8528h

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

IOMUXC_SPADC_PDRAM_DQM3 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_DQM3. 00 min 01 +50ps 10 +100ps 11 +150ps
26–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for pad: DRAM_DQM3. 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2:

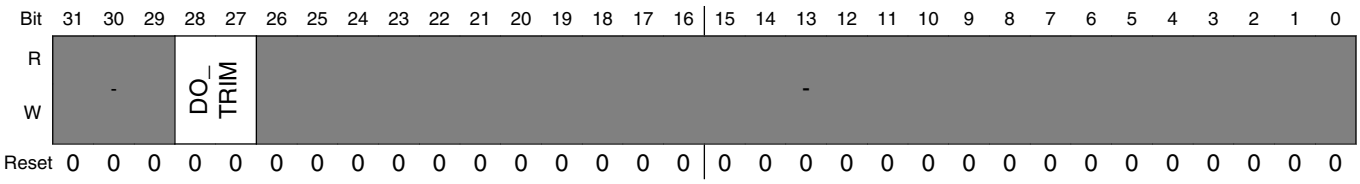
Table continues on the next page...

IOMUXC_SPADC_PDRAM_DQM3 field descriptions (continued)

Field	Description
	110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.332 SW_PAD_CTL_PAD_DRAM_D24 (IOMUXC_SPADC_PDRAM_D24)

Address: IOMUXC_SPADC_PDRAM_D24 is 53FA_8000h base + 52Ch offset = 53FA_852Ch

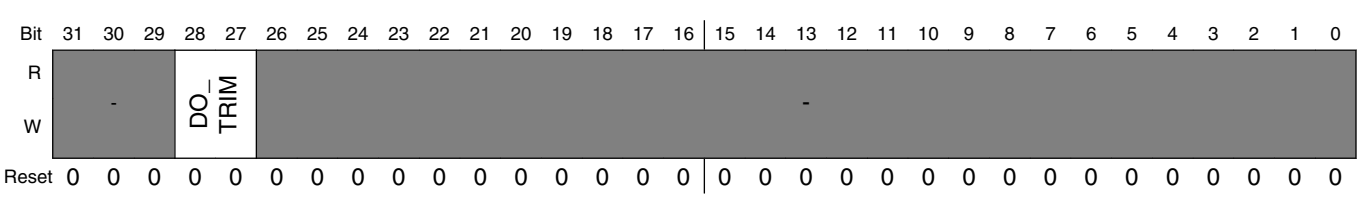


IOMUXC_SPADC_PDRAM_D24 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D24. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.333 SW_PAD_CTL_PAD_DRAM_D25 (IOMUXC_SPADC_PDRAM_D25)

Address: IOMUXC_SPADC_PDRAM_D25 is 53FA_8000h base + 530h offset = 53FA_8530h



IOMUXC_SPADC_PDRAM_D25 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D25. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.334 SW_PAD_CTL_PAD_DRAM_D26 (IOMUXC_SPADC_PDRAM_D26)

Address: IOMUXC_SPADC_PDRAM_D26 is 53FA_8000h base + 534h offset = 53FA_8534h

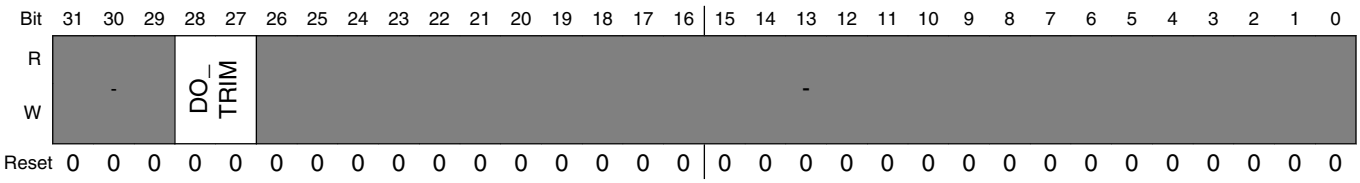
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-			DO_TRIM		-											-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SPADC_PDRAM_D26 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D26. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.335 SW_PAD_CTL_PAD_DRAM_D27 (IOMUXC_SPADC_PDRAM_D27)

Address: IOMUXC_SPADC_PDRAM_D27 is 53FA_8000h base + 538h offset = 53FA_8538h

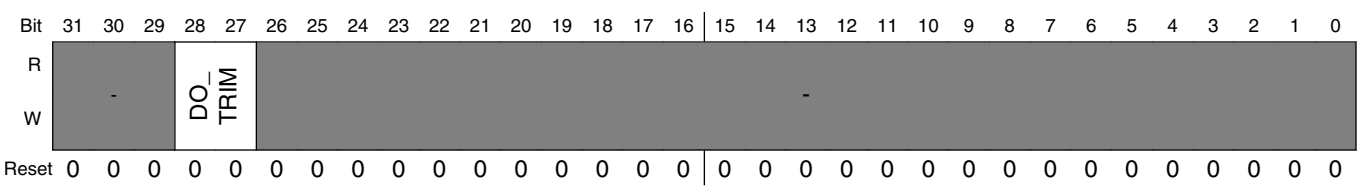


IOMUXC_SPADC_PDRAM_D27 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D27. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.336 SW_PAD_CTL_PAD_DRAM_D28 (IOMUXC_SPADC_PDRAM_D28)

Address: IOMUXC_SPADC_PDRAM_D28 is 53FA_8000h base + 53Ch offset = 53FA_853Ch



IOMUXC_SPADC_PDRAM_D28 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D28.

Table continues on the next page...

IOMUXC_SPADC_PDRAM_D28 field descriptions (continued)

Field	Description
	00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.337 SW_PAD_CTL_PAD_DRAM_D29 (IOMUXC_SPADC_PDRAM_D29)

Address: IOMUXC_SPADC_PDRAM_D29 is 53FA_8000h base + 540h offset = 53FA_8540h

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

IOMUXC_SPADC_PDRAM_D29 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D29. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.338 SW_PAD_CTL_PAD_DRAM_D30 (IOMUXC_SPADC_PDRAM_D30)

Address: IOMUXC_SPADC_PDRAM_D30 is 53FA_8000h base + 544h offset = 53FA_8544h

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

IOMUXC_SPADC_PDRAM_D30 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D30. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.339 SW_PAD_CTL_PAD_DRAM_D31 (IOMUXC_SPADC_PDRAM_D31)

Address: IOMUXC_SPADC_PDRAM_D31 is 53FA_8000h base + 548h offset = 53FA_8548h

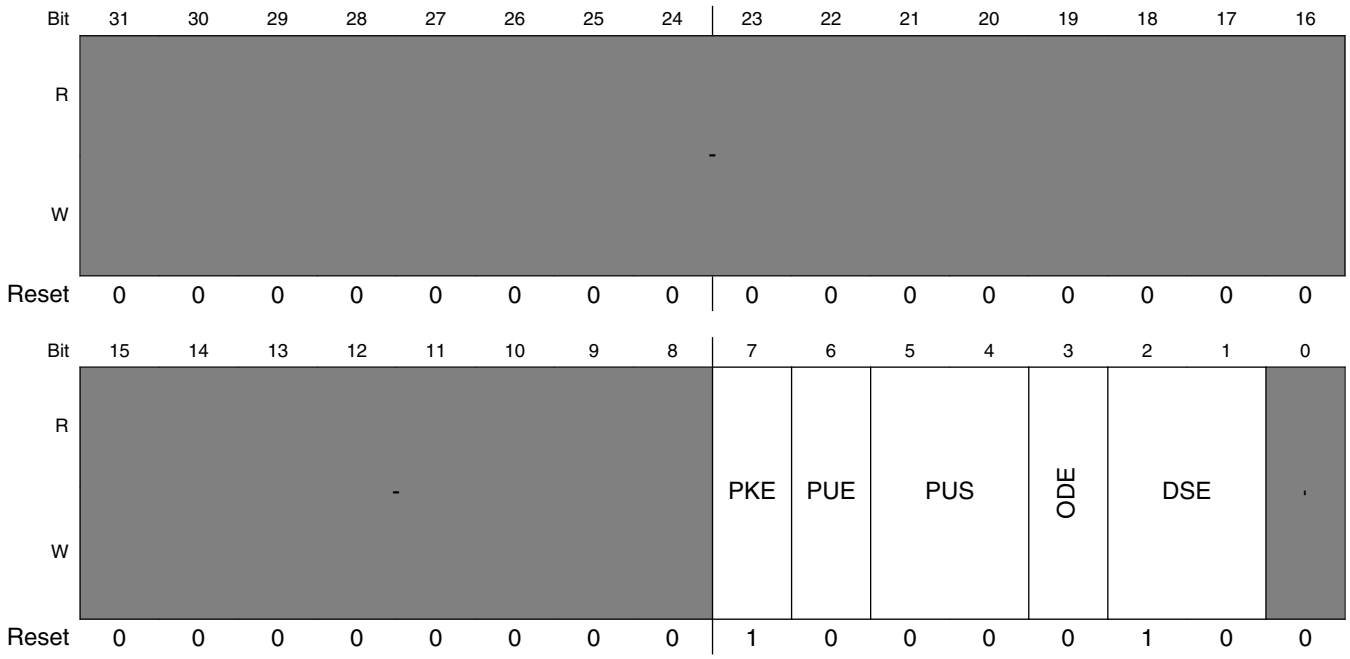
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-			DO_TRIM		-											-															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SPADC_PDRAM_D31 field descriptions

Field	Description
31–29 -	Reserved
28–27 DO_TRIM	do_trim Field Select one out of next values for pad: DRAM_D31. 00 min 01 +50ps 10 +100ps 11 +150ps
26–0 -	Reserved

35.4.340 SW_PAD_CTL_PAD_EPDC_D0 (IOMUXC_SPADC_PEPDC_D0)

Address: IOMUXC_SPADC_PEPDC_D0 is 53FA_8000h base + 54Ch offset = 53FA_854Ch



IOMUXC_SPADC_PEPDC_D0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_D0 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.341 SW_PAD_CTL_PAD_EPDC_D1 (IOMUXC_SPADC_PEPDC_D1)

Address: IOMUXC_SPADC_PEPDC_D1 is 53FA_8000h base + 550h offset = 53FA_8550h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

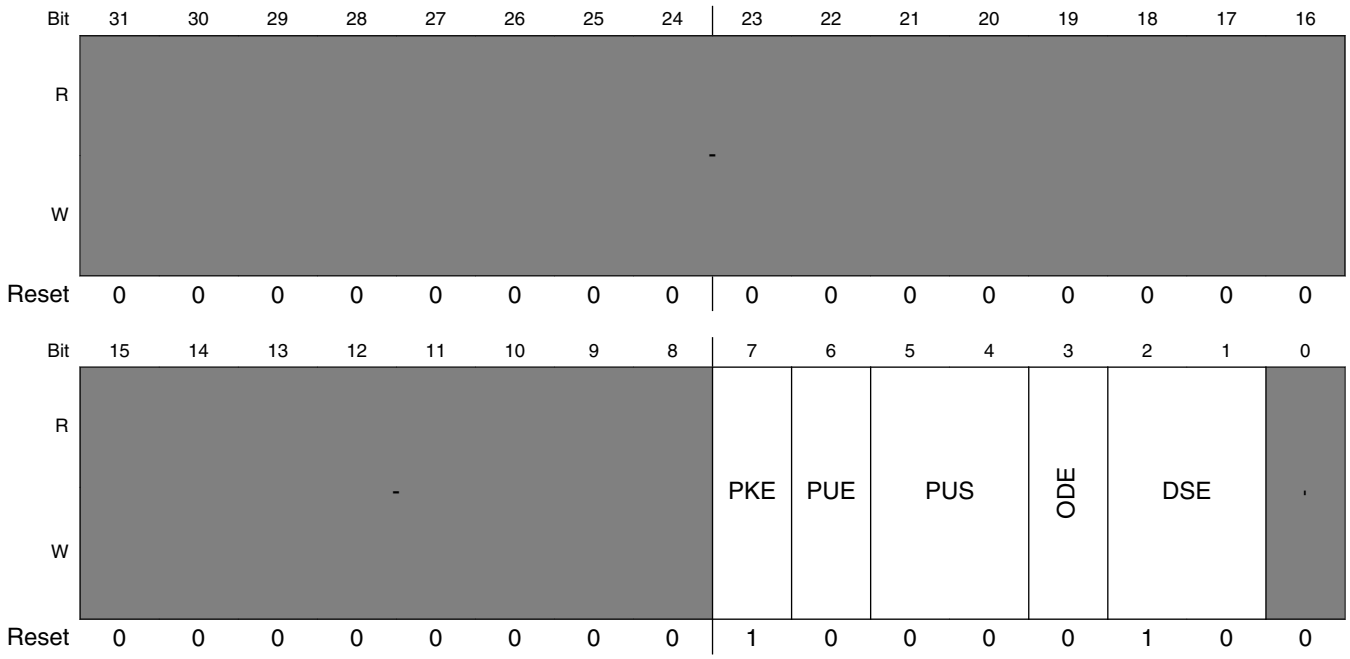
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_D1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.342 SW_PAD_CTL_PAD_EPDC_D2 (IOMUXC_SPADC_PEPDC_D2)

Address: IOMUXC_SPADC_PEPDC_D2 is 53FA_8000h base + 554h offset = 53FA_8554h



IOMUXC_SPADC_PEPDC_D2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_D2 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.343 SW_PAD_CTL_PAD_EPDC_D3 (IOMUXC_SPADC_PEPDC_D3)

Address: IOMUXC_SPADC_PEPDC_D3 is 53FA_8000h base + 558h offset = 53FA_8558h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

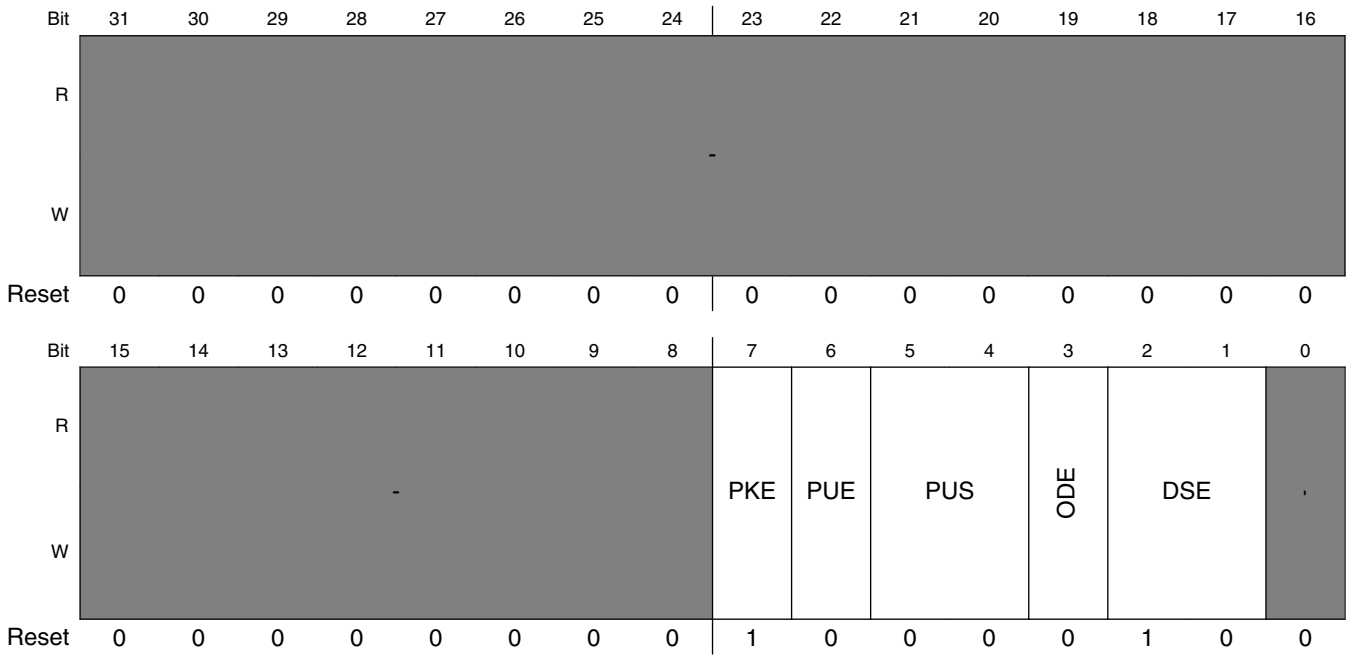
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_D3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D3. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.344 SW_PAD_CTL_PAD_EPDC_D4 (IOMUXC_SPADC_PEPDC_D4)

Address: IOMUXC_SPADC_PEPDC_D4 is 53FA_8000h base + 55Ch offset = 53FA_855Ch



IOMUXC_SPADC_PEPDC_D4 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D4. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D4. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D4. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_D4 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D4. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D4. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.345 SW_PAD_CTL_PAD_EPDC_D5 (IOMUXC_SPADC_PEPDC_D5)

Address: IOMUXC_SPADC_PEPDC_D5 is 53FA_8000h base + 560h offset = 53FA_8560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

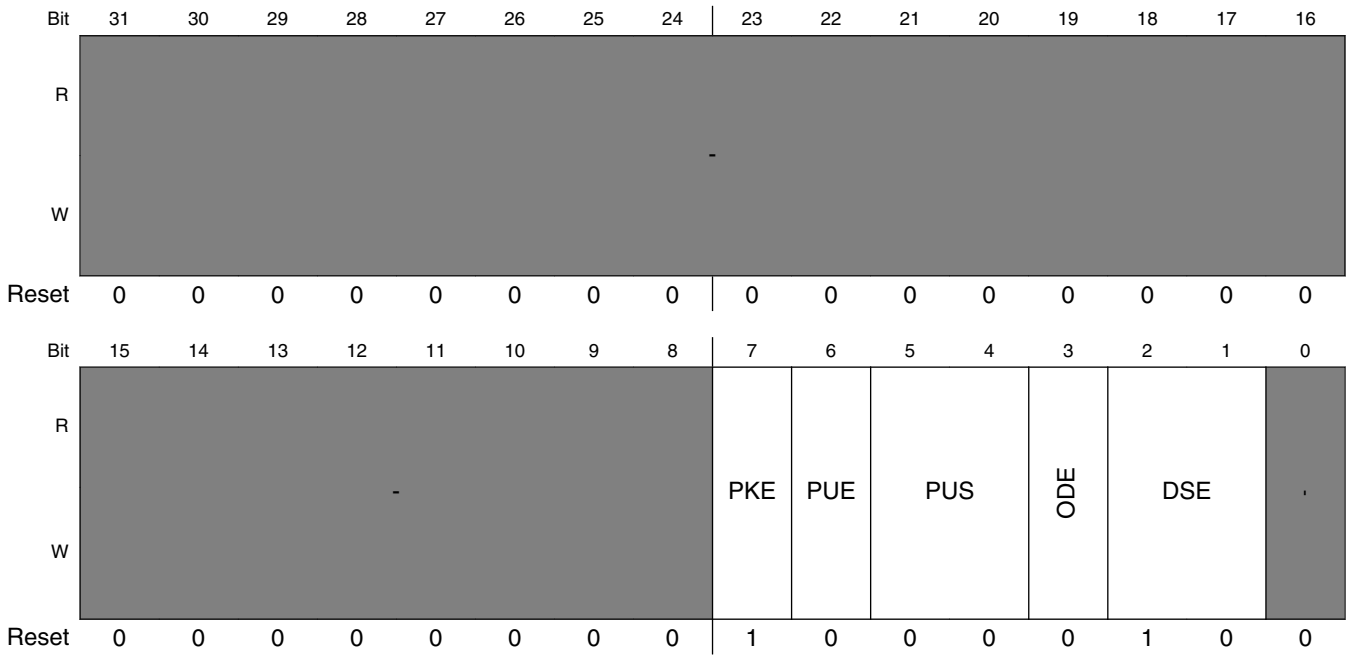
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_D5 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D5. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D5. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D5. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D5. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D5. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.346 SW_PAD_CTL_PAD_EPDC_D6 (IOMUXC_SPADC_PEPDC_D6)

Address: IOMUXC_SPADC_PEPDC_D6 is 53FA_8000h base + 564h offset = 53FA_8564h



IOMUXC_SPADC_PEPDC_D6 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D6. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D6. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D6. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_D6 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D6. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D6. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.347 SW_PAD_CTL_PAD_EPDC_D7 (IOMUXC_SPADC_PEPDC_D7)

Address: IOMUXC_SPADC_PEPDC_D7 is 53FA_8000h base + 568h offset = 53FA_8568h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

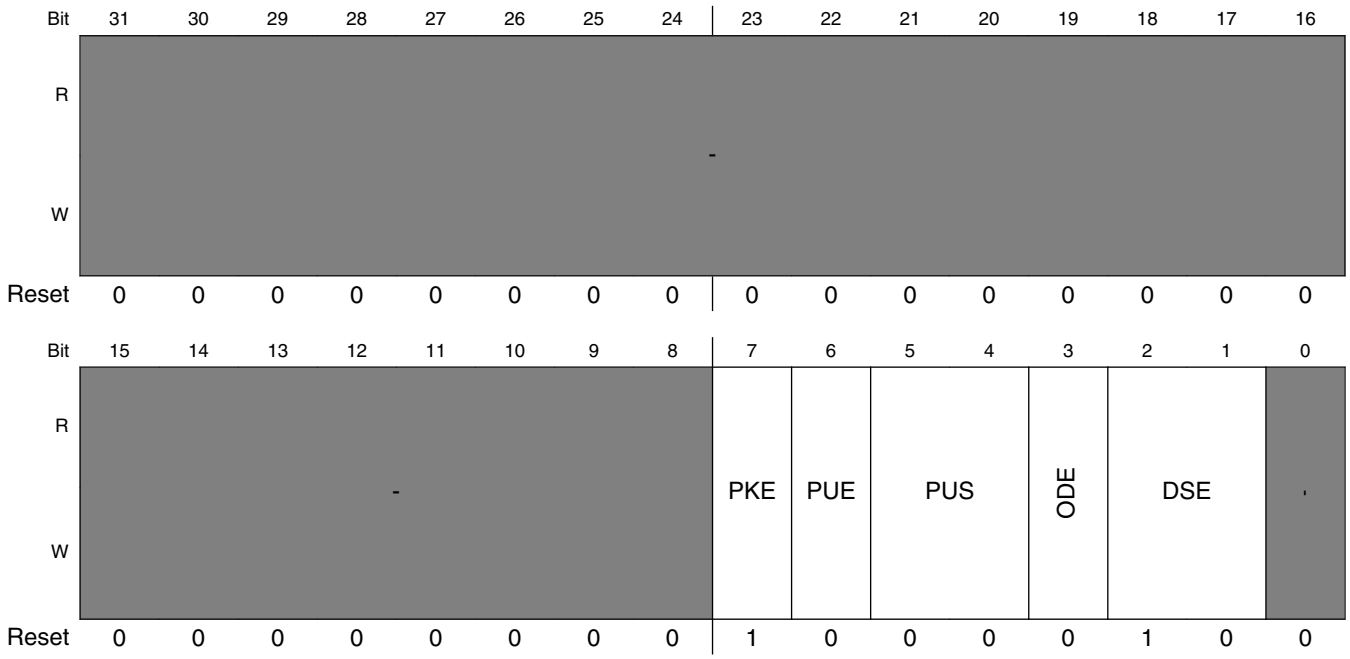
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_D7 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D7. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D7. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D7. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D7. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D7. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.348 SW_PAD_CTL_PAD_EPDC_D8 (IOMUXC_SPADC_PEPDC_D8)

Address: IOMUXC_SPADC_PEPDC_D8 is 53FA_8000h base + 56Ch offset = 53FA_856Ch



IOMUXC_SPADC_PEPDC_D8 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D8. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D8. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D8. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_D8 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D8. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D8. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.349 SW_PAD_CTL_PAD_EPDC_D9 (IOMUXC_SPADC_PEPDC_D9)

Address: IOMUXC_SPADC_PEPDC_D9 is 53FA_8000h base + 570h offset = 53FA_8570h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

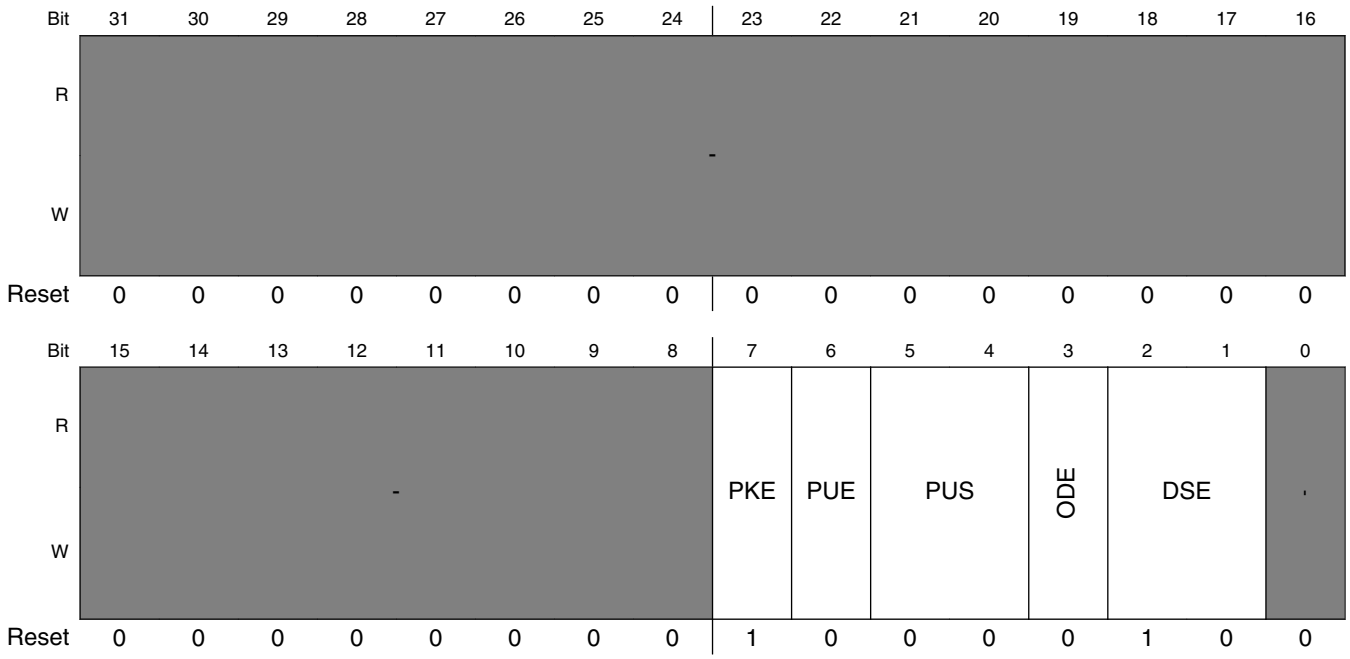
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_D9 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D9. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D9. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D9. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D9. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D9. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.350 SW_PAD_CTL_PAD_EPDC_D10 (IOMUXC_SPADC_PEPDC_D10)

Address: IOMUXC_SPADC_PEPDC_D10 is 53FA_8000h base + 574h offset = 53FA_8574h



IOMUXC_SPADC_PEPDC_D10 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D10. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D10. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D10. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_D10 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D10. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D10. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.351 SW_PAD_CTL_PAD_EPDC_D11 (IOMUXC_SPADC_PEPDC_D11)

Address: IOMUXC_SPADC_PEPDC_D11 is 53FA_8000h base + 578h offset = 53FA_8578h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

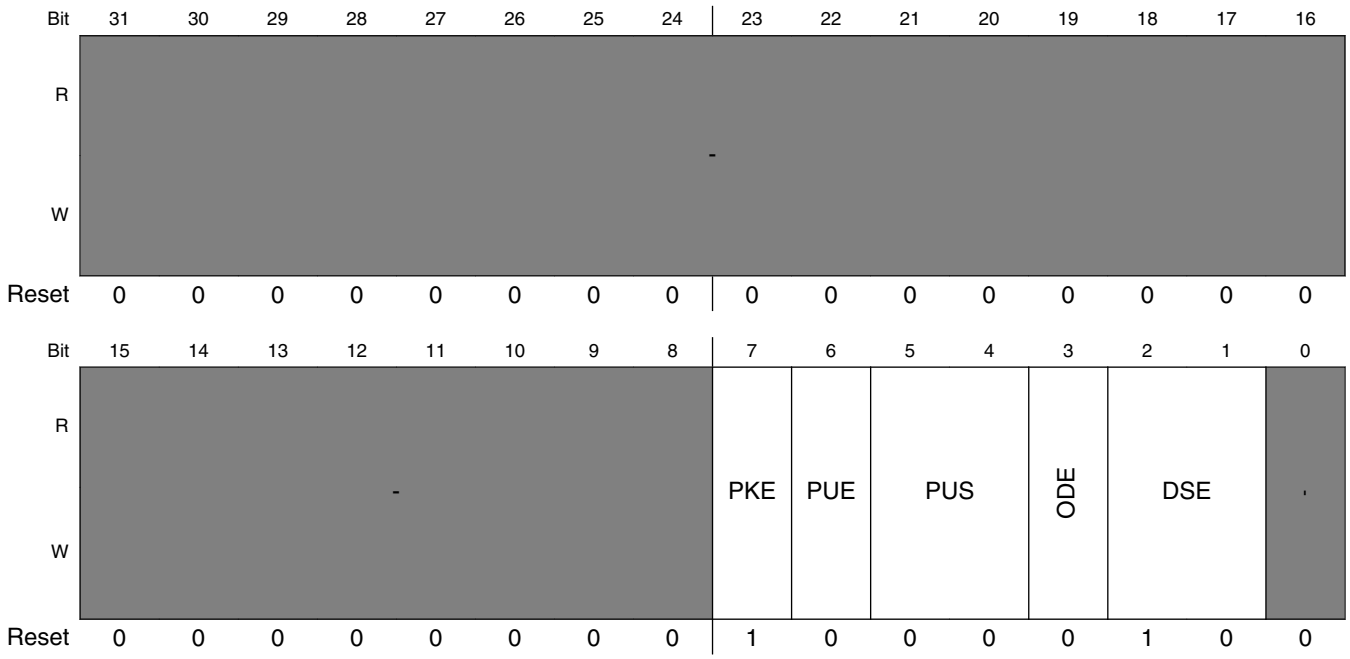
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_D11 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D11. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D11. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D11. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D11. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D11. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.352 SW_PAD_CTL_PAD_EPDC_D12 (IOMUXC_SPADC_PEPDC_D12)

Address: IOMUXC_SPADC_PEPDC_D12 is 53FA_8000h base + 57Ch offset = 53FA_857Ch



IOMUXC_SPADC_PEPDC_D12 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D12. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D12. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D12. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_D12 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D12. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D12. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.353 SW_PAD_CTL_PAD_EPDC_D13 (IOMUXC_SPADC_PEPDC_D13)

Address: IOMUXC_SPADC_PEPDC_D13 is 53FA_8000h base + 580h offset = 53FA_8580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

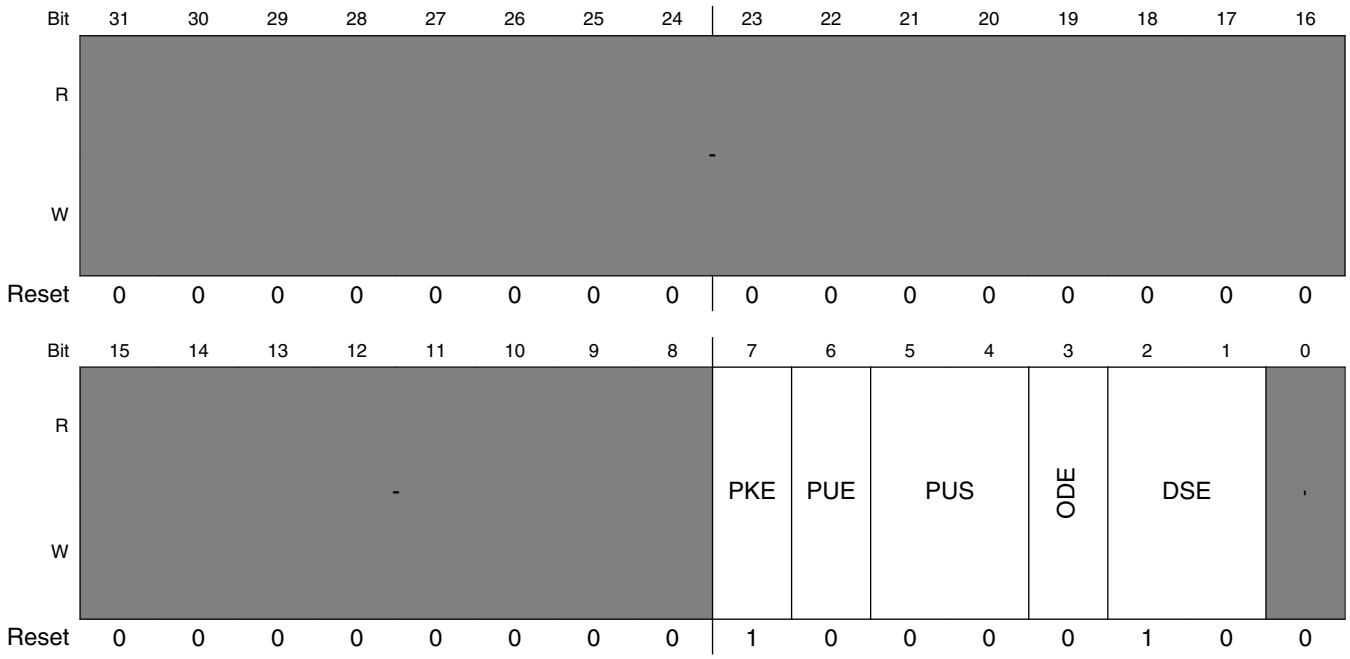
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_D13 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D13. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D13. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D13. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D13. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D13. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.354 SW_PAD_CTL_PAD_EPDC_D14 (IOMUXC_SPADC_PEPDC_D14)

Address: IOMUXC_SPADC_PEPDC_D14 is 53FA_8000h base + 584h offset = 53FA_8584h



IOMUXC_SPADC_PEPDC_D14 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D14. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D14. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D14. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_D14 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D14. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D14. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.355 SW_PAD_CTL_PAD_EPDC_D15 (IOMUXC_SPADC_PEPDC_D15)

Address: IOMUXC_SPADC_PEPDC_D15 is 53FA_8000h base + 588h offset = 53FA_8588h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

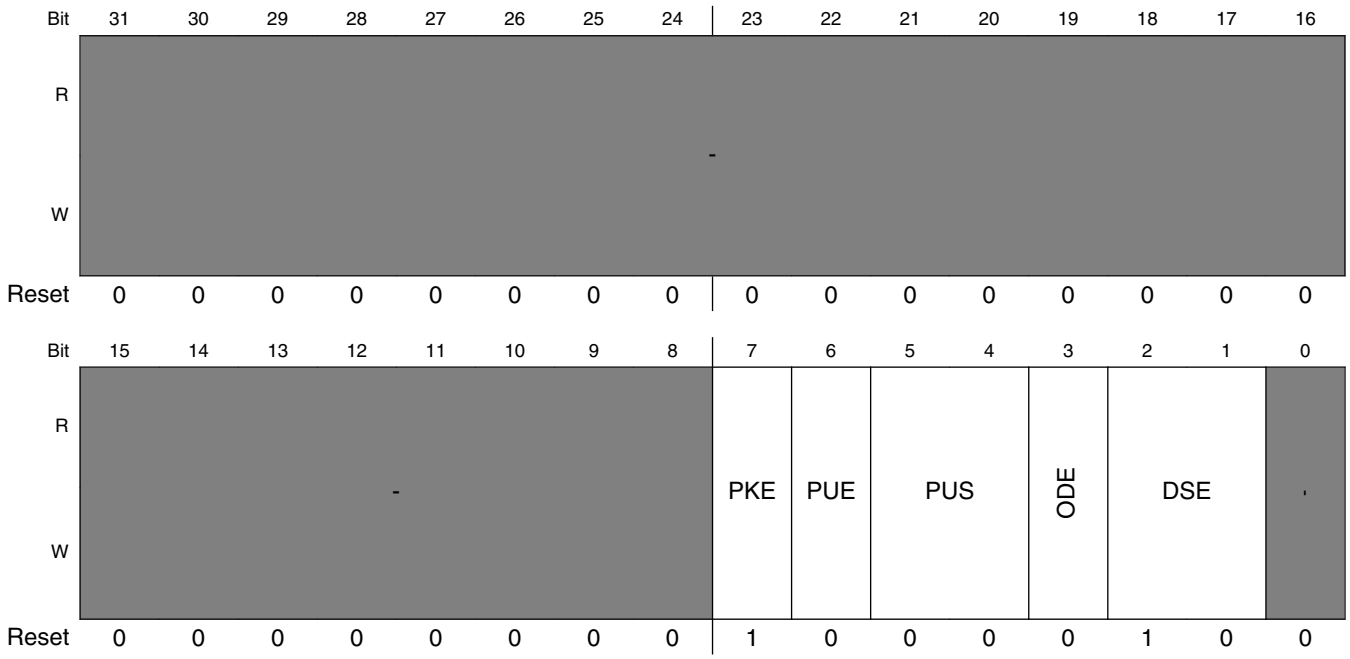
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_D15 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_D15. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_D15. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_D15. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_D15. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_D15. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.356 SW_PAD_CTL_PAD_EPDC_GDCLK (IOMUXC_SPADC_PEPDC_GDCLK)

Address: IOMUXC_SPADC_PEPDC_GDCLK is 53FA_8000h base + 58Ch offset = 53FA_858Ch



IOMUXC_SPADC_PEPDC_GDCLK field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_GDCLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_GDCLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_GDCLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_GDCLK field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_GDCLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_GDCLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.357 SW_PAD_CTL_PAD_EPDC_GDSP (IOMUXC_SPADC_PEPDC_GDSP)

Address: IOMUXC_SPADC_PEPDC_GDSP is 53FA_8000h base + 590h offset = 53FA_8590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_GDSP field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_GDSP. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_GDSP. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_GDSP. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_GDSP. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_GDSP. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.358 SW_PAD_CTL_PAD_EPDC_GDOE (IOMUXC_SPADC_PEPDC_GDOE)

Address: IOMUXC_SPADC_PEPDC_GDOE is 53FA_8000h base + 594h offset = 53FA_8594h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_GDOE field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_GDOE. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_GDOE. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_GDOE. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_GDOE field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_GDOE. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_GDOE. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.359 SW_PAD_CTL_PAD_EPDC_GDRL (IOMUXC_SPADC_PEPDC_GDRL)

Address: IOMUXC_SPADC_PEPDC_GDRL is 53FA_8000h base + 598h offset = 53FA_8598h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

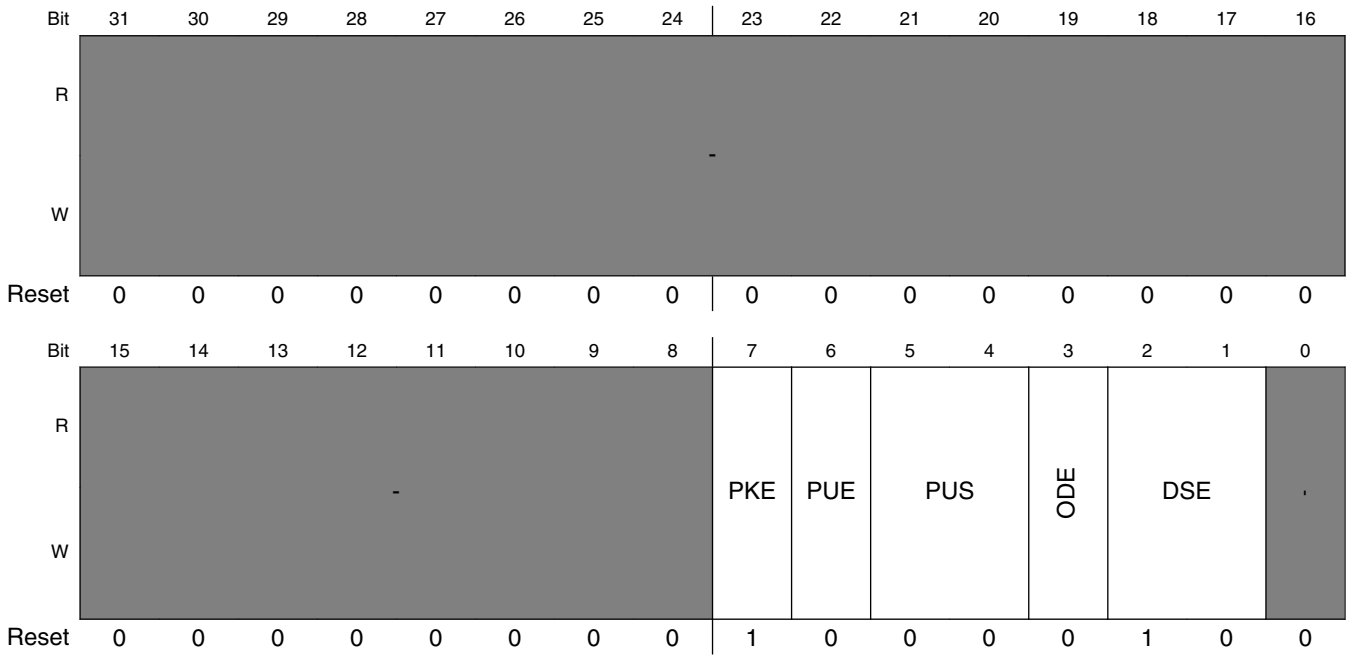
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_GDRL field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_GDRL. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_GDRL. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_GDRL. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_GDRL. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_GDRL. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.360 SW_PAD_CTL_PAD_EPDC_SDCLK (IOMUXC_SPADC_PEPDC_SDCLK)

Address: IOMUXC_SPADC_PEPDC_SDCLK is 53FA_8000h base + 59Ch offset = 53FA_859Ch



IOMUXC_SPADC_PEPDC_SDCLK field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDCLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDCLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDCLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_SDCLK field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDCLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.361 SW_PAD_CTL_PAD_EPDC_SDOEZ (IOMUXC_SPADC_PEPDC_SDOEZ)

Address: IOMUXC_SPADC_PEPDC_SDOEZ is 53FA_8000h base + 5A0h offset = 53FA_85A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_SDOEZ field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDOEZ. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDOEZ. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDOEZ. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDOEZ. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDOEZ. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.362 SW_PAD_CTL_PAD_EPDC_SDOED (IOMUXC_SPADC_PEPDC_SDOED)

Address: IOMUXC_SPADC_PEPDC_SDOED is 53FA_8000h base + 5A4h offset = 53FA_85A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_SDOED field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDOED. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDOED. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDOED. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_SDOED field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDOED. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDOED. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.363 SW_PAD_CTL_PAD_EPDC_SDOE (IOMUXC_SPADC_PEPDC_SDOE)

Address: IOMUXC_SPADC_PEPDC_SDOE is 53FA_8000h base + 5A8h offset = 53FA_85A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

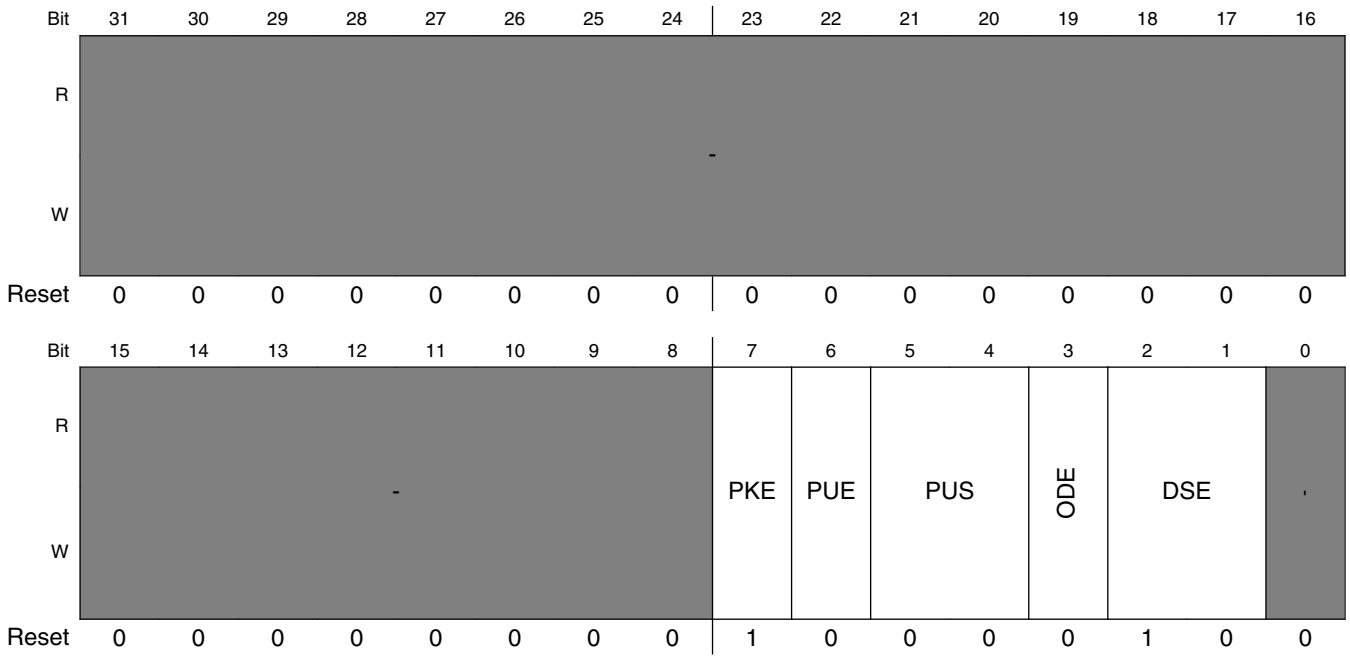
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_SDOE field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDOE. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDOE. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDOE. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDOE. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDOE. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.364 SW_PAD_CTL_PAD_EPDC_SDLE (IOMUXC_SPADC_PEPDC_SDLE)

Address: IOMUXC_SPADC_PEPDC_SDLE is 53FA_8000h base + 5ACh offset = 53FA_85ACh



IOMUXC_SPADC_PEPDC_SDLE field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDLE. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDLE. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDLE. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_SDLE field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDLE. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDLE. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.365 SW_PAD_CTL_PAD_EPDC_SDCLKN (IOMUXC_SPADC_PEPDC_SDCLKN)

Address: IOMUXC_SPADC_PEPDC_SDCLKN is 53FA_8000h base + 5B0h offset = 53FA_85B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_SDCLKN field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDCLKN. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDCLKN. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDCLKN. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDCLKN. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCLKN. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.366 SW_PAD_CTL_PAD_EPDC_SDSHR (IOMUXC_SPADC_PEPDC_SDSHR)

Address: IOMUXC_SPADC_PEPDC_SDSHR is 53FA_8000h base + 5B4h offset = 53FA_85B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_SDSHR field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDSHR. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDSHR. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDSHR. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_SDSHR field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDSHR. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDSHR. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.367 SW_PAD_CTL_PAD_EPDC_PWRCOM (IOMUXC_SPADC_PEPDC_PWRCOM)

Address: IOMUXC_SPADC_PEPDC_PWRCOM is 53FA_8000h base + 5B8h offset = 53FA_85B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_PWRCOM field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_PWRCOM. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_PWRCOM. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_PWRCOM. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_PWRCOM. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_PWRCOM. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.368 SW_PAD_CTL_PAD_EPDC_PWRSTAT (IOMUXC_SPADC_PEPDC_PWRSTAT)

Address: IOMUXC_SPADC_PEPDC_PWRSTAT is 53FA_8000h base + 5BCh offset = 53FA_85BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_PWRSTAT field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_PWRSTAT. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_PWRSTAT. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_PWRSTAT. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_PWRSTAT field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_PWRSTAT. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_PWRSTAT. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.369 SW_PAD_CTL_PAD_EPDC_PWRCTRL0 (IOMUXC_SPADC_PEPDC_PWRCTRL0)

Address: IOMUXC_SPADC_PEPDC_PWRCTRL0 is 53FA_8000h base + 5C0h offset = 53FA_85C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

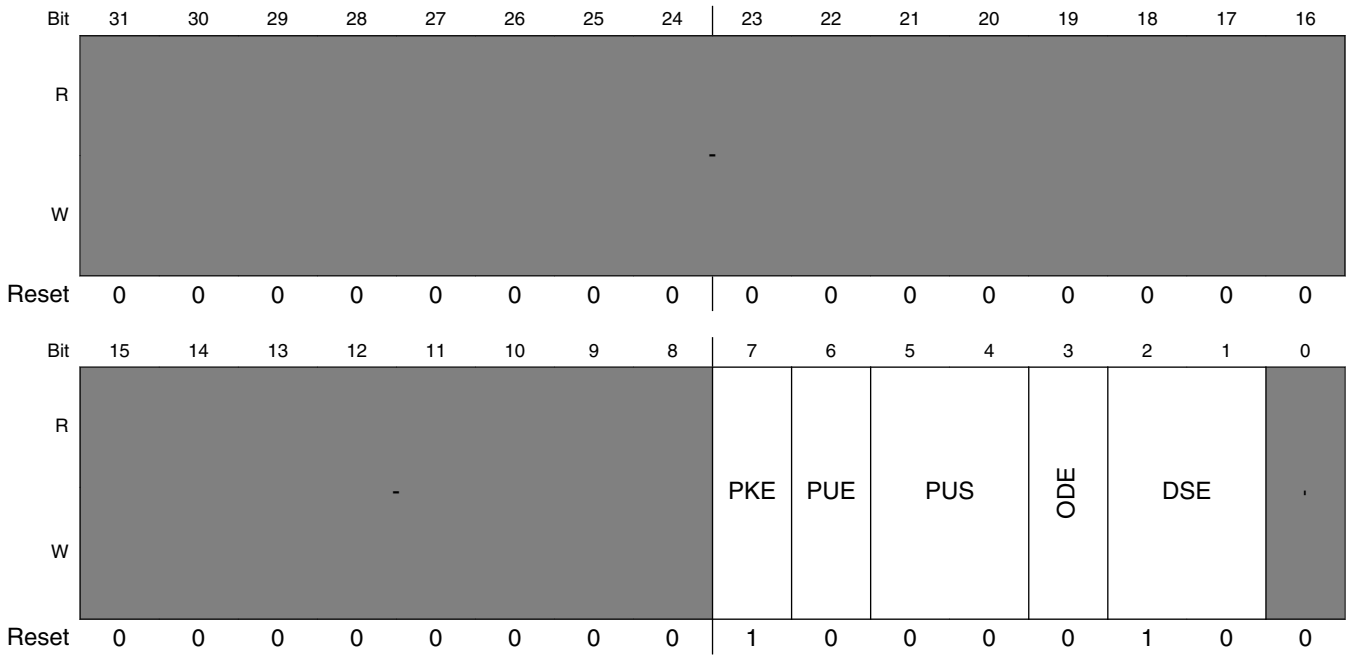
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_PWRCTRL0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_PWRCTRL0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_PWRCTRL0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_PWRCTRL0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_PWRCTRL0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_PWRCTRL0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.370 SW_PAD_CTL_PAD_EPDC_PWRCTRL1 (IOMUXC_SPADC_PEPDC_PWRCTRL1)

Address: IOMUXC_SPADC_PEPDC_PWRCTRL1 is 53FA_8000h base + 5C4h offset = 53FA_85C4h



IOMUXC_SPADC_PEPDC_PWRCTRL1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_PWRCTRL1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_PWRCTRL1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_PWRCTRL1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_PWRCTRL1 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_PWRCTRL1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_PWRCTRL1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.371 SW_PAD_CTL_PAD_EPDC_PWRCTRL2 (IOMUXC_SPADC_PEPDC_PWRCTRL2)

Address: IOMUXC_SPADC_PEPDC_PWRCTRL2 is 53FA_8000h base + 5C8h offset = 53FA_85C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

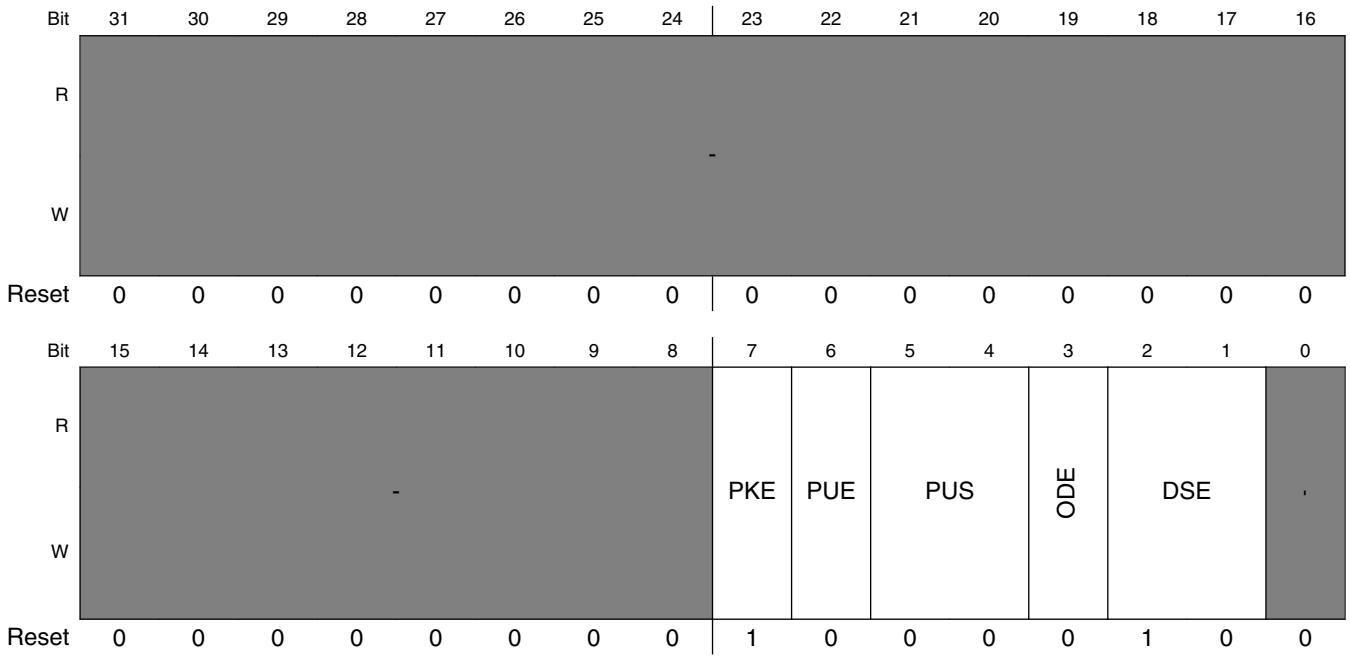
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_PWRCTRL2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_PWRCTRL2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_PWRCTRL2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_PWRCTRL2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_PWRCTRL2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_PWRCTRL2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.372 SW_PAD_CTL_PAD_EPDC_PWRCTRL3 (IOMUXC_SPADC_PEPDC_PWRCTRL3)

Address: IOMUXC_SPADC_PEPDC_PWRCTRL3 is 53FA_8000h base + 5CCh offset = 53FA_85CCh



IOMUXC_SPADC_PEPDC_PWRCTRL3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_PWRCTRL3. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_PWRCTRL3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_PWRCTRL3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_PWRCTRL3 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_PWRCTRL3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_PWRCTRL3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.373 SW_PAD_CTL_PAD_EPDC_VCOM0 (IOMUXC_SPADC_PEPDC_VCOM0)

Address: IOMUXC_SPADC_PEPDC_VCOM0 is 53FA_8000h base + 5D0h offset = 53FA_85D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

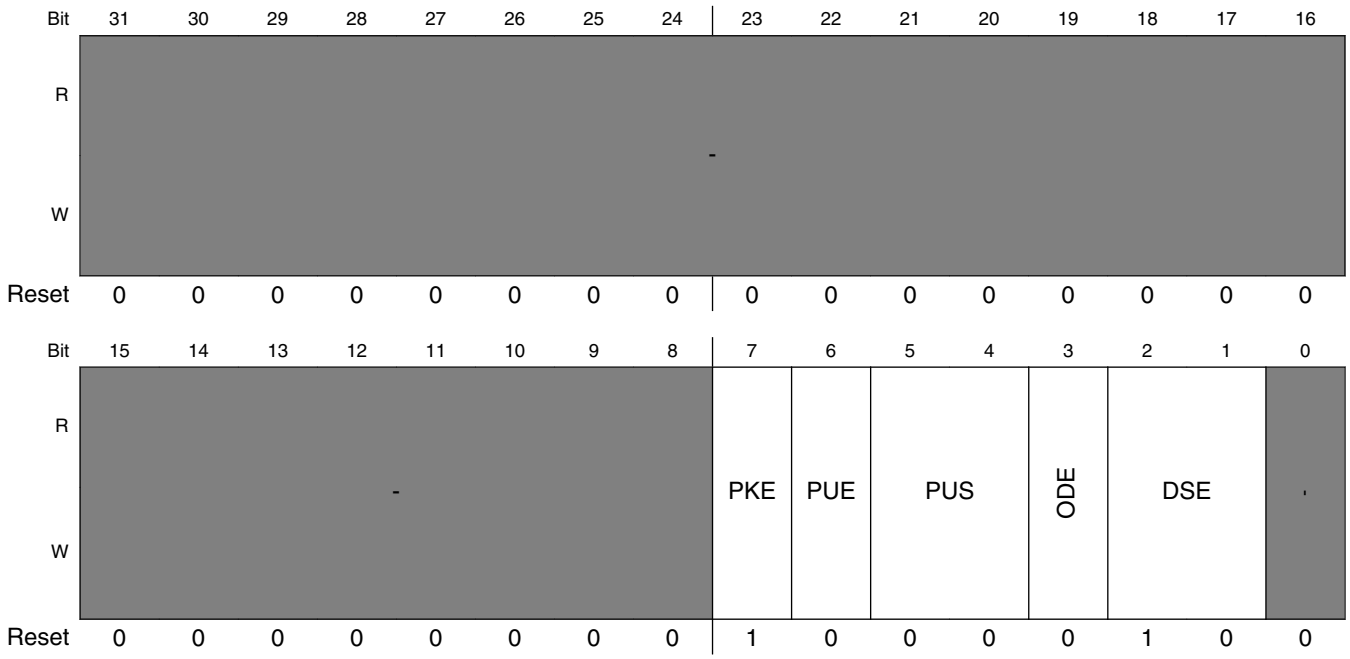
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_VCOM0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_VCOM0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_VCOM0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_VCOM0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_VCOM0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_VCOM0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.374 SW_PAD_CTL_PAD_EPDC_VCOM1 (IOMUXC_SPADC_PEPDC_VCOM1)

Address: IOMUXC_SPADC_PEPDC_VCOM1 is 53FA_8000h base + 5D4h offset = 53FA_85D4h



IOMUXC_SPADC_PEPDC_VCOM1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_VCOM1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_VCOM1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_VCOM1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_VCOM1 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_VCOM1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_VCOM1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.375 SW_PAD_CTL_PAD_EPDC_BDR0 (IOMUXC_SPADC_PEPDC_BDR0)

Address: IOMUXC_SPADC_PEPDC_BDR0 is 53FA_8000h base + 5D8h offset = 53FA_85D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

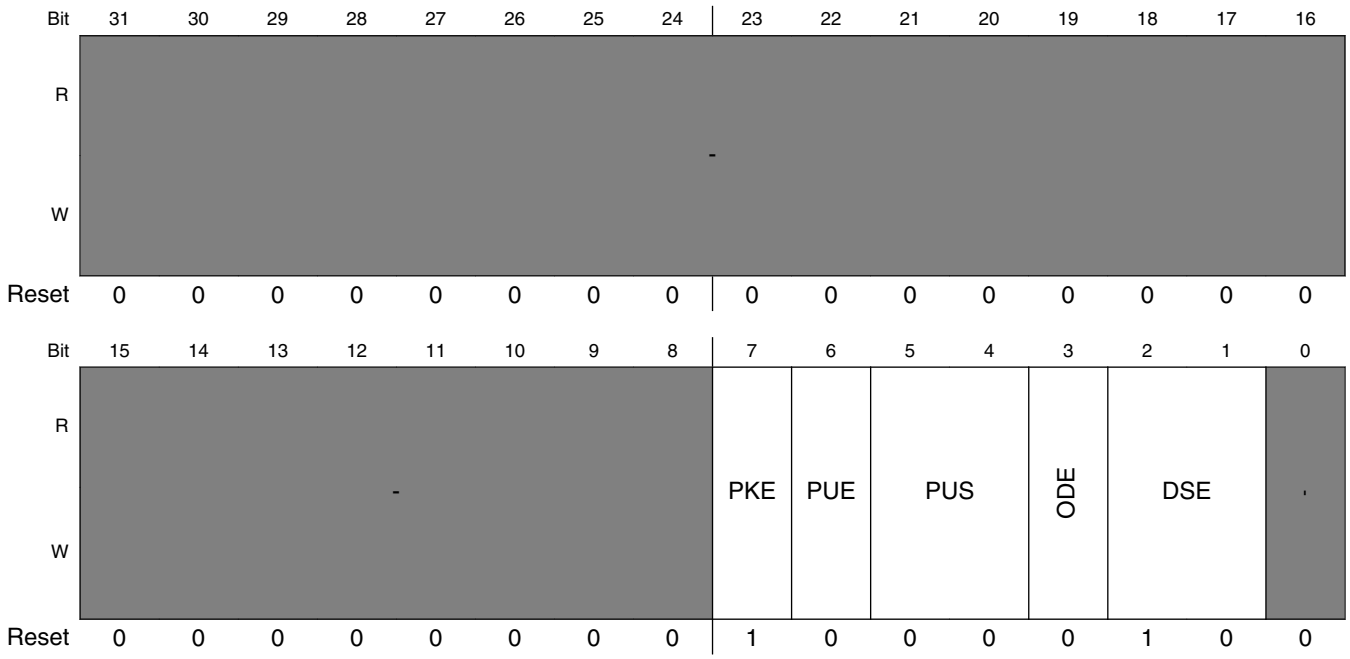
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_BDR0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_BDR0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_BDR0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_BDR0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_BDR0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_BDR0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.376 SW_PAD_CTL_PAD_EPDC_BDR1 (IOMUXC_SPADC_PEPDC_BDR1)

Address: IOMUXC_SPADC_PEPDC_BDR1 is 53FA_8000h base + 5DCh offset = 53FA_85DCh



IOMUXC_SPADC_PEPDC_BDR1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_BDR1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_BDR1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_BDR1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_BDR1 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_BDR1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_BDR1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.377 SW_PAD_CTL_PAD_EPDC_SDCE0 (IOMUXC_SPADC_PEPDC_SDCE0)

Address: IOMUXC_SPADC_PEPDC_SDCE0 is 53FA_8000h base + 5E0h offset = 53FA_85E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

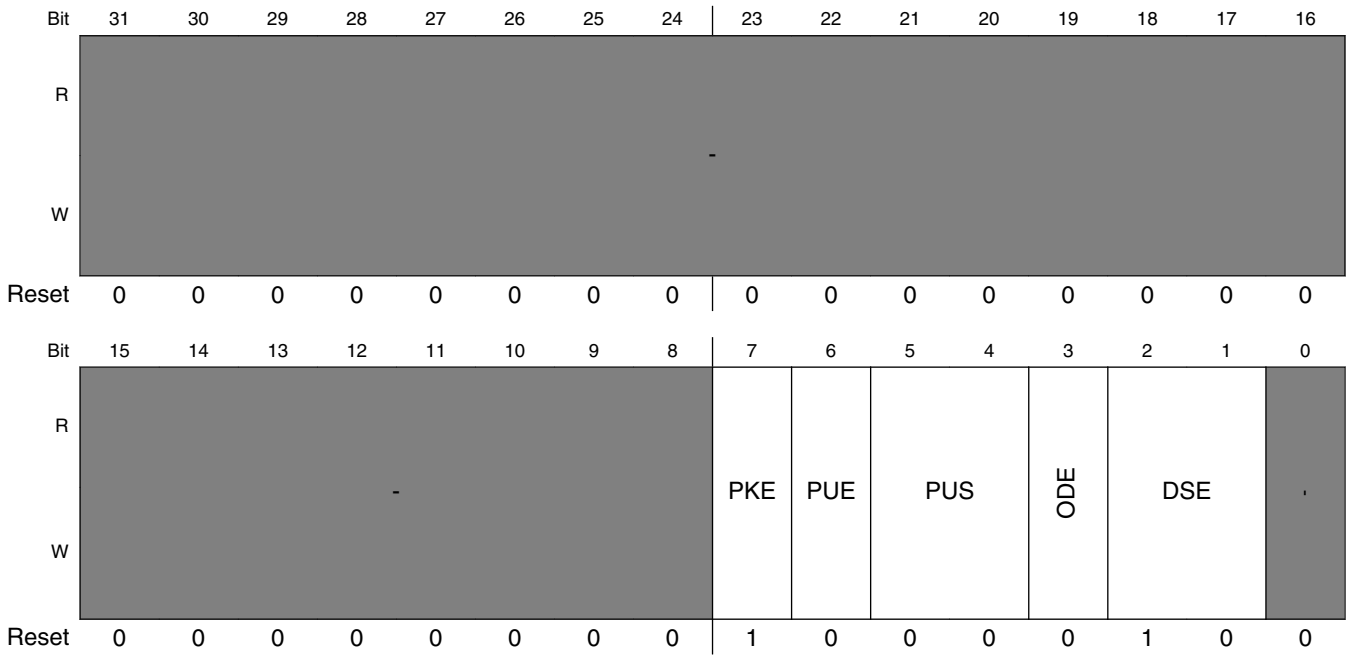
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_SDCE0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDCE0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDCE0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDCE0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDCE0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.378 SW_PAD_CTL_PAD_EPDC_SDCE1 (IOMUXC_SPADC_PEPDC_SDCE1)

Address: IOMUXC_SPADC_PEPDC_SDCE1 is 53FA_8000h base + 5E4h offset = 53FA_85E4h



IOMUXC_SPADC_PEPDC_SDCE1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDCE1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDCE1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDCE1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_SDCE1 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDCE1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.379 SW_PAD_CTL_PAD_EPDC_SDCE2 (IOMUXC_SPADC_PEPDC_SDCE2)

Address: IOMUXC_SPADC_PEPDC_SDCE2 is 53FA_8000h base + 5E8h offset = 53FA_85E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

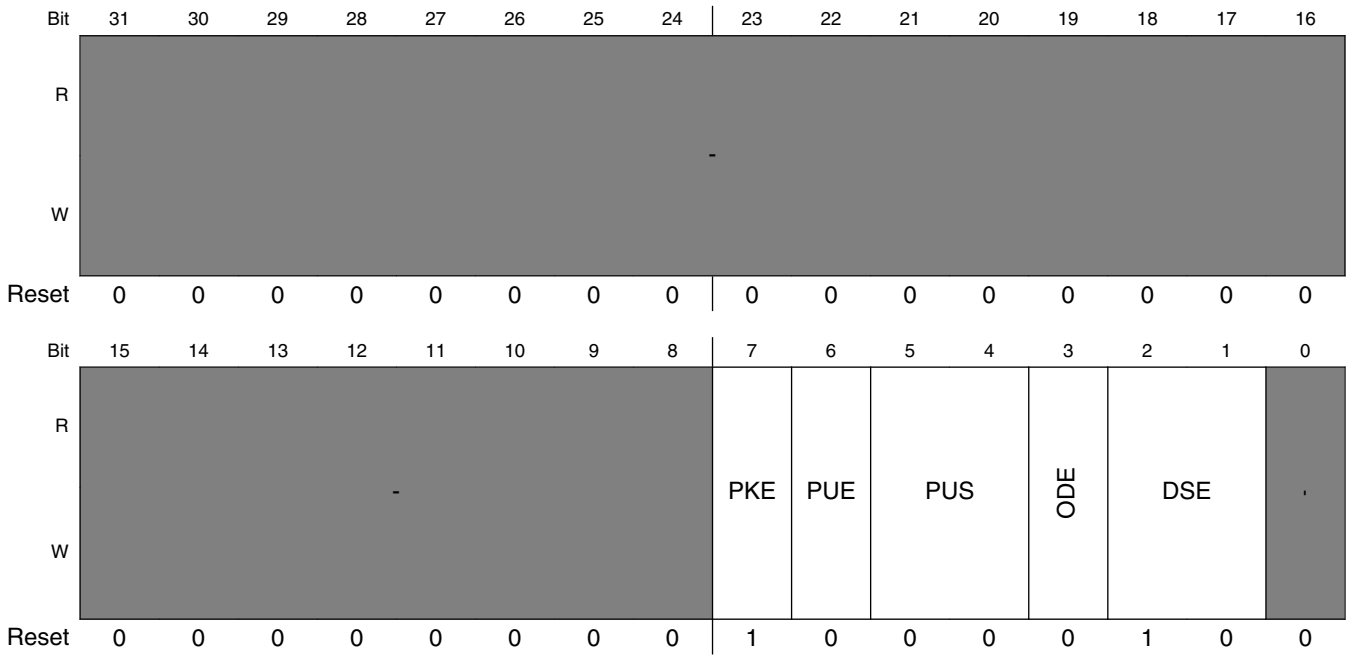
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_SDCE2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDCE2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDCE2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDCE2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDCE2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.380 SW_PAD_CTL_PAD_EPDC_SDCE3 (IOMUXC_SPADC_PEPDC_SDCE3)

Address: IOMUXC_SPADC_PEPDC_SDCE3 is 53FA_8000h base + 5ECh offset = 53FA_85ECh



IOMUXC_SPADC_PEPDC_SDCE3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDCE3. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDCE3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDCE3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_SDCE3 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDCE3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.381 SW_PAD_CTL_PAD_EPDC_SDCE4 (IOMUXC_SPADC_PEPDC_SDCE4)

Address: IOMUXC_SPADC_PEPDC_SDCE4 is 53FA_8000h base + 5F0h offset = 53FA_85F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

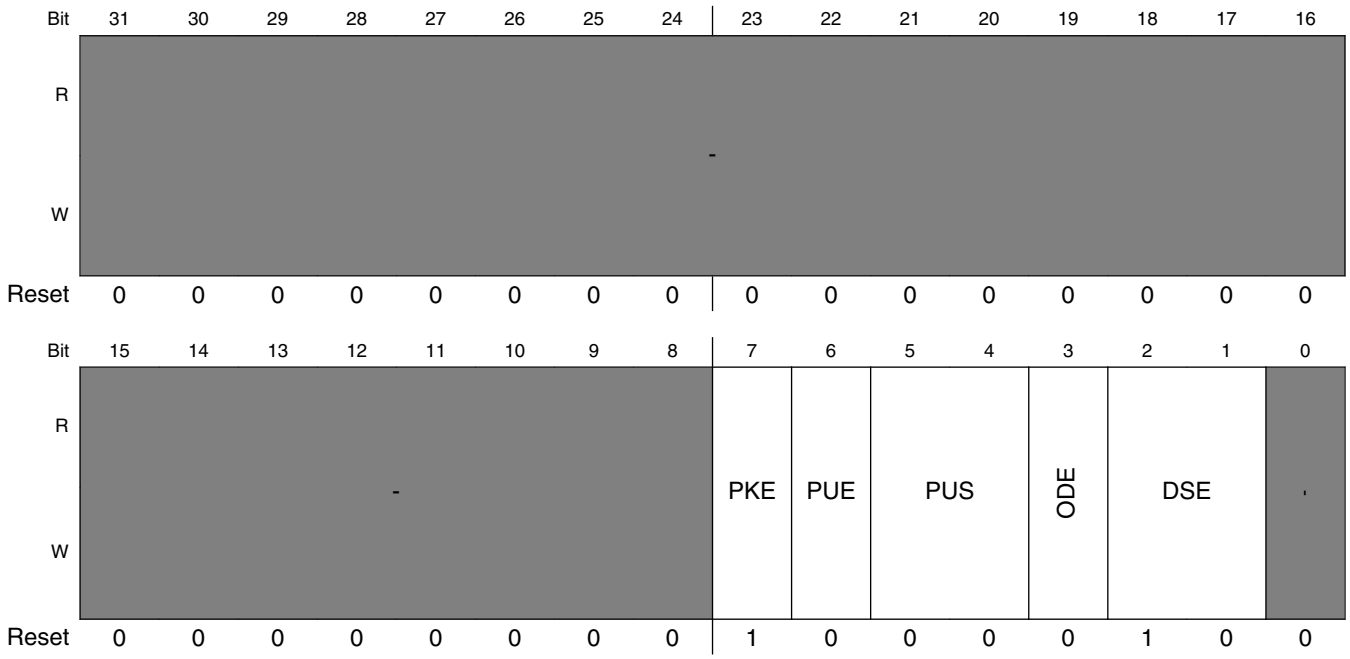
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

IOMUXC_SPADC_PEPDC_SDCE4 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDCE4. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDCE4. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDCE4. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDCE4. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE4. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.382 SW_PAD_CTL_PAD_EPDC_SDCE5 (IOMUXC_SPADC_PEPDC_SDCE5)

Address: IOMUXC_SPADC_PEPDC_SDCE5 is 53FA_8000h base + 5F4h offset = 53FA_85F4h



IOMUXC_SPADC_PEPDC_SDCE5 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EPDC_SDCE5. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EPDC_SDCE5. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EPDC_SDCE5. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEPDC_SDCE5 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EPDC_SDCE5. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EPDC_SDCE5. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.383 SW_PAD_CTL_PAD_EIM_DA0 (IOMUXC_SPADC_PEIM_DA0)

Address: IOMUXC_SPADC_PEIM_DA0 is 53FA_8000h base + 5F8h offset = 53FA_85F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

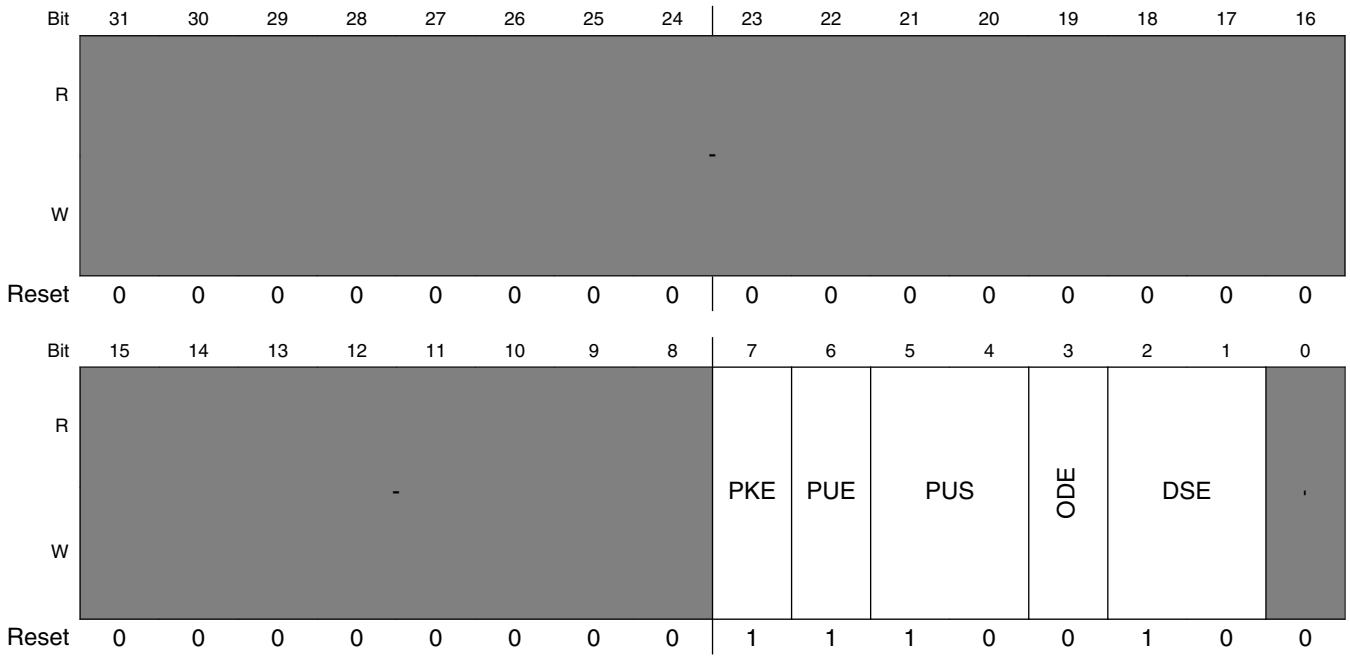
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.384 SW_PAD_CTL_PAD_EIM_DA1 (IOMUXC_SPADC_PEIM_DA1)

Address: IOMUXC_SPADC_PEIM_DA1 is 53FA_8000h base + 5FCh offset = 53FA_85FCh



IOMUXC_SPADC_PEIM_DA1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_DA1 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.385 SW_PAD_CTL_PAD_EIM_DA2 (IOMUXC_SPADC_PEIM_DA2)

Address: IOMUXC_SPADC_PEIM_DA2 is 53FA_8000h base + 600h offset = 53FA_8600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.386 SW_PAD_CTL_PAD_EIM_DA3 (IOMUXC_SPADC_PEIM_DA3)

Address: IOMUXC_SPADC_PEIM_DA3 is 53FA_8000h base + 604h offset = 53FA_8604h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA3 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA3. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA3. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA3. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_DA3 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA3. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA3. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.387 SW_PAD_CTL_PAD_EIM_DA4 (IOMUXC_SPADC_PEIM_DA4)

Address: IOMUXC_SPADC_PEIM_DA4 is 53FA_8000h base + 608h offset = 53FA_8608h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

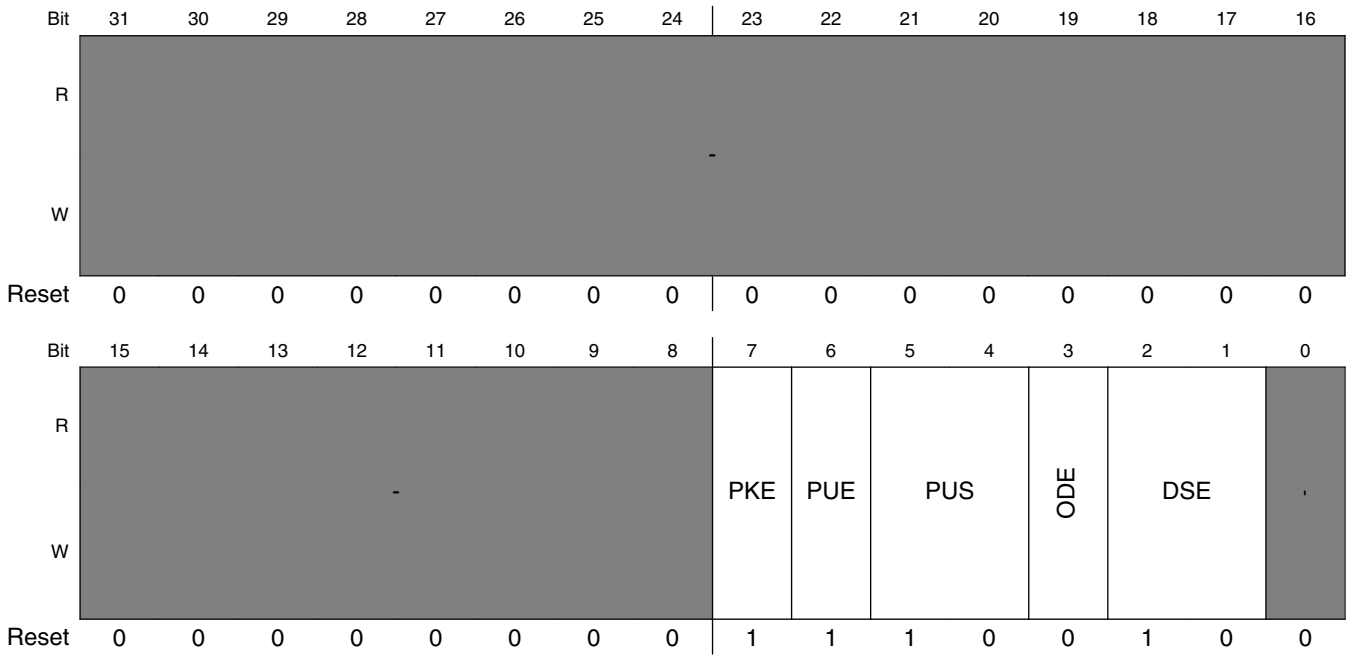
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA4 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA4. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA4. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA4. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA4. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA4. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.388 SW_PAD_CTL_PAD_EIM_DA5 (IOMUXC_SPADC_PEIM_DA5)

Address: IOMUXC_SPADC_PEIM_DA5 is 53FA_8000h base + 60Ch offset = 53FA_860Ch



IOMUXC_SPADC_PEIM_DA5 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA5. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA5. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA5. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_DA5 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA5. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA5. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.389 SW_PAD_CTL_PAD_EIM_DA6 (IOMUXC_SPADC_PEIM_DA6)

Address: IOMUXC_SPADC_PEIM_DA6 is 53FA_8000h base + 610h offset = 53FA_8610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA6 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA6. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA6. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA6. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA6. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA6. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.390 SW_PAD_CTL_PAD_EIM_DA7 (IOMUXC_SPADC_PEIM_DA7)

Address: IOMUXC_SPADC_PEIM_DA7 is 53FA_8000h base + 614h offset = 53FA_8614h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA7 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA7. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA7. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA7. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_DA7 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA7. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA7. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.391 SW_PAD_CTL_PAD_EIM_DA8 (IOMUXC_SPADC_PEIM_DA8)

Address: IOMUXC_SPADC_PEIM_DA8 is 53FA_8000h base + 618h offset = 53FA_8618h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

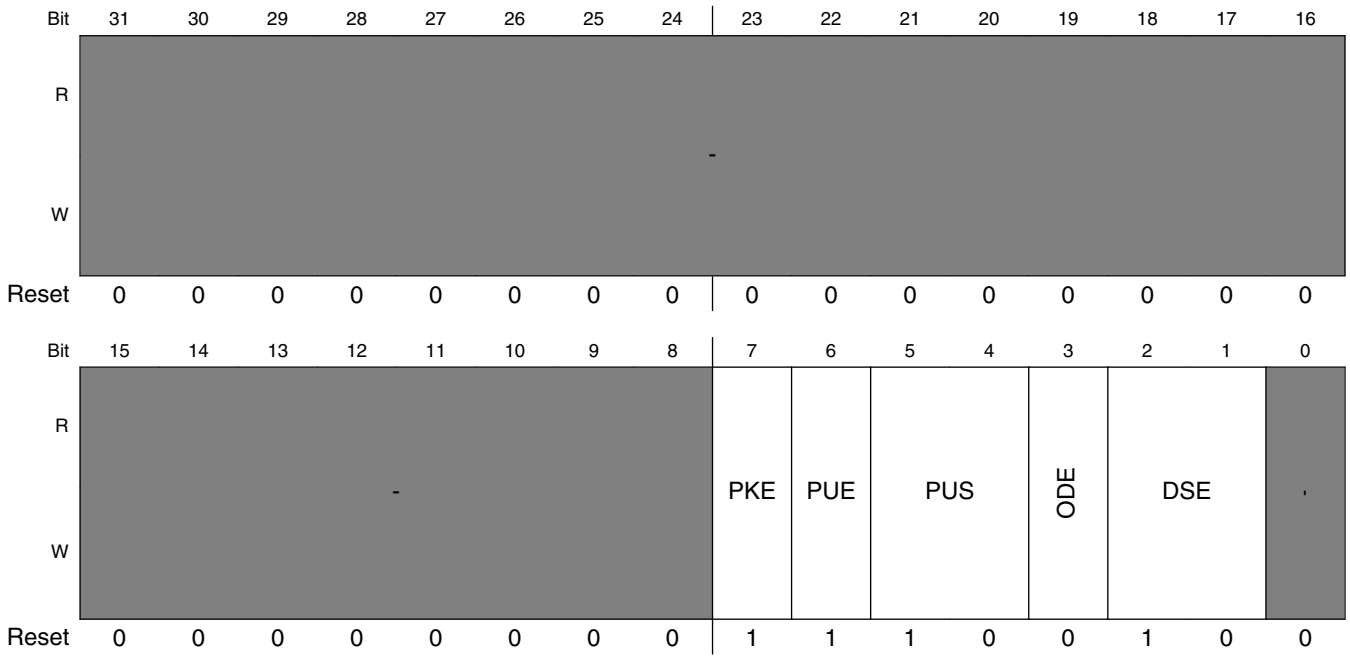
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA8 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA8. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA8. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA8. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA8. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA8. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.392 SW_PAD_CTL_PAD_EIM_DA9 (IOMUXC_SPADC_PEIM_DA9)

Address: IOMUXC_SPADC_PEIM_DA9 is 53FA_8000h base + 61Ch offset = 53FA_861Ch



IOMUXC_SPADC_PEIM_DA9 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA9. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA9. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA9. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_DA9 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA9. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA9. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.393 SW_PAD_CTL_PAD_EIM_DA10 (IOMUXC_SPADC_PEIM_DA10)

Address: IOMUXC_SPADC_PEIM_DA10 is 53FA_8000h base + 620h offset = 53FA_8620h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA10 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA10. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA10. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA10. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA10. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA10. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.394 SW_PAD_CTL_PAD_EIM_DA11 (IOMUXC_SPADC_PEIM_DA11)

Address: IOMUXC_SPADC_PEIM_DA11 is 53FA_8000h base + 624h offset = 53FA_8624h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA11 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA11. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA11. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA11. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_DA11 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA11. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA11. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.395 SW_PAD_CTL_PAD_EIM_DA12 (IOMUXC_SPADC_PEIM_DA12)

Address: IOMUXC_SPADC_PEIM_DA12 is 53FA_8000h base + 628h offset = 53FA_8628h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

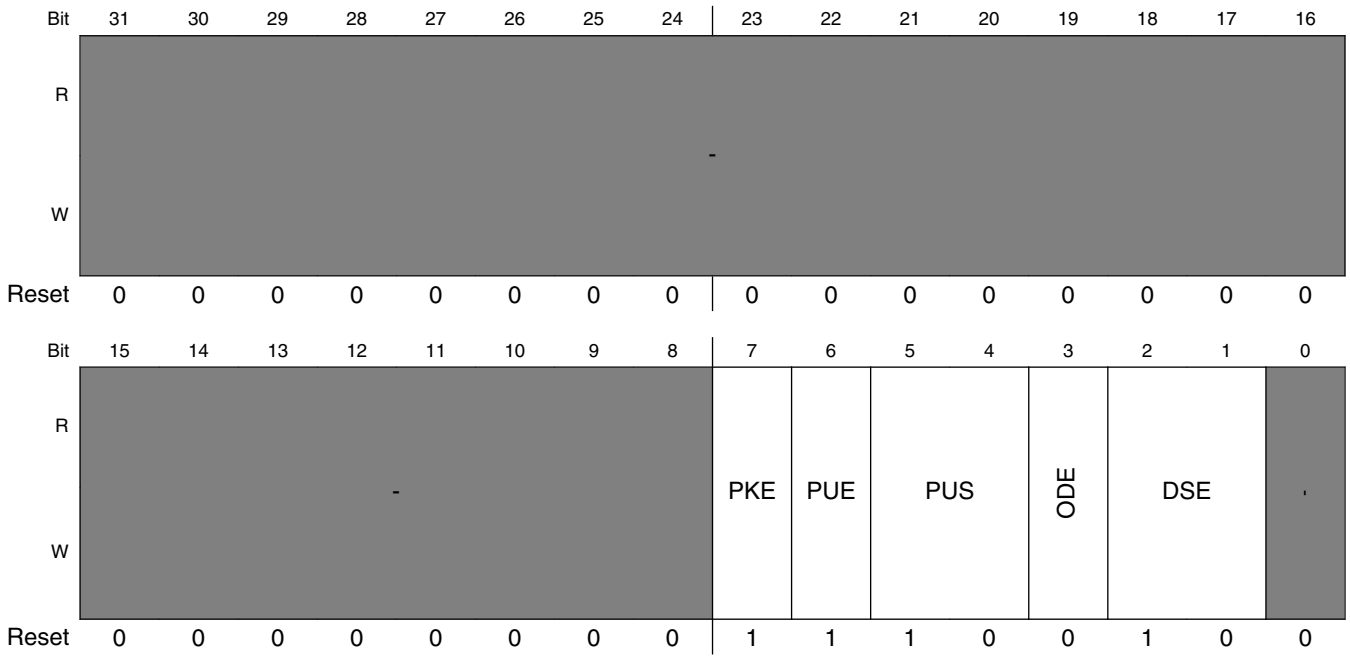
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA12 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA12. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA12. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA12. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA12. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA12. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.396 SW_PAD_CTL_PAD_EIM_DA13 (IOMUXC_SPADC_PEIM_DA13)

Address: IOMUXC_SPADC_PEIM_DA13 is 53FA_8000h base + 62Ch offset = 53FA_862Ch



IOMUXC_SPADC_PEIM_DA13 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA13. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA13. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA13. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_DA13 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA13. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA13. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.397 SW_PAD_CTL_PAD_EIM_DA14 (IOMUXC_SPADC_PEIM_DA14)

Address: IOMUXC_SPADC_PEIM_DA14 is 53FA_8000h base + 630h offset = 53FA_8630h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA14 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA14. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA14. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_DA14. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA14. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA14. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.398 SW_PAD_CTL_PAD_EIM_DA15 (IOMUXC_SPADC_PEIM_DA15)

Address: IOMUXC_SPADC_PEIM_DA15 is 53FA_8000h base + 634h offset = 53FA_8634h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_DA15 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_DA15. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_DA15. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Slect one out of next values for pad: EIM_DA15. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_DA15 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_DA15. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_DA15. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.399 SW_PAD_CTL_PAD_EIM_CS2 (IOMUXC_SPADC_PEIM_CS2)

Address: IOMUXC_SPADC_PEIM_CS2 is 53FA_8000h base + 638h offset = 53FA_8638h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

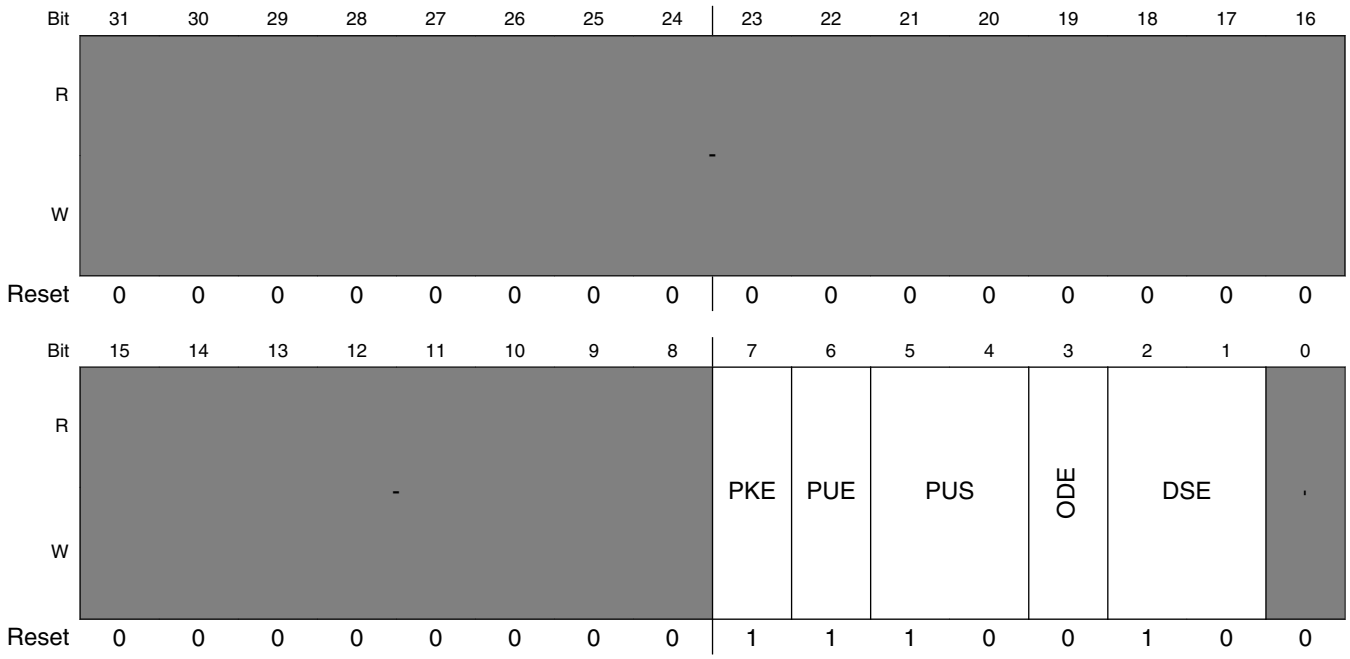
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_CS2 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_CS2. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_CS2. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_CS2. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_CS2. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_CS2. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.400 SW_PAD_CTL_PAD_EIM_CS1 (IOMUXC_SPADC_PEIM_CS1)

Address: IOMUXC_SPADC_PEIM_CS1 is 53FA_8000h base + 63Ch offset = 53FA_863Ch



IOMUXC_SPADC_PEIM_CS1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_CS1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_CS1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_CS1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_CS1 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_CS1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_CS1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.401 SW_PAD_CTL_PAD_EIM_CS0 (IOMUXC_SPADC_PEIM_CS0)

Address: IOMUXC_SPADC_PEIM_CS0 is 53FA_8000h base + 640h offset = 53FA_8640h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

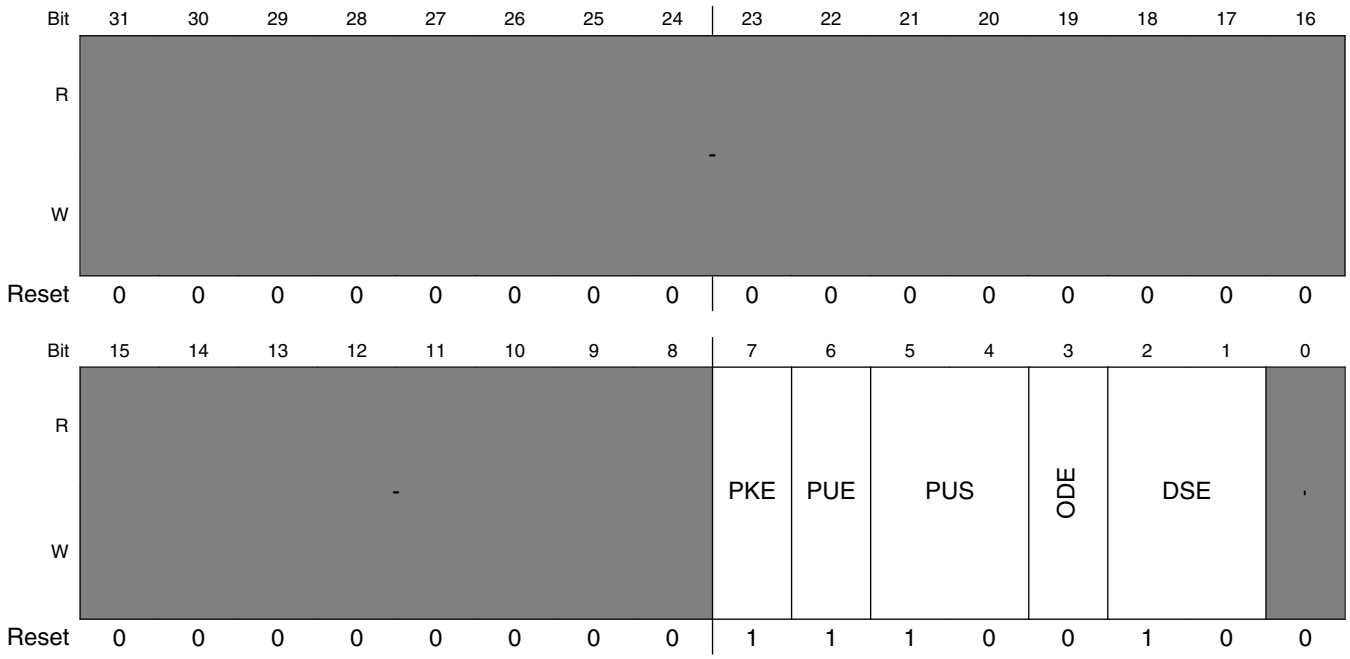
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_CS0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_CS0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_CS0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_CS0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_CS0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_CS0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.402 SW_PAD_CTL_PAD_EIM_EB0 (IOMUXC_SPADC_PEIM_EB0)

Address: IOMUXC_SPADC_PEIM_EB0 is 53FA_8000h base + 644h offset = 53FA_8644h



IOMUXC_SPADC_PEIM_EB0 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_EB0. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_EB0. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_EB0. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_EB0 field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_EB0. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_EB0. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.403 SW_PAD_CTL_PAD_EIM_EB1 (IOMUXC_SPADC_PEIM_EB1)

Address: IOMUXC_SPADC_PEIM_EB1 is 53FA_8000h base + 648h offset = 53FA_8648h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

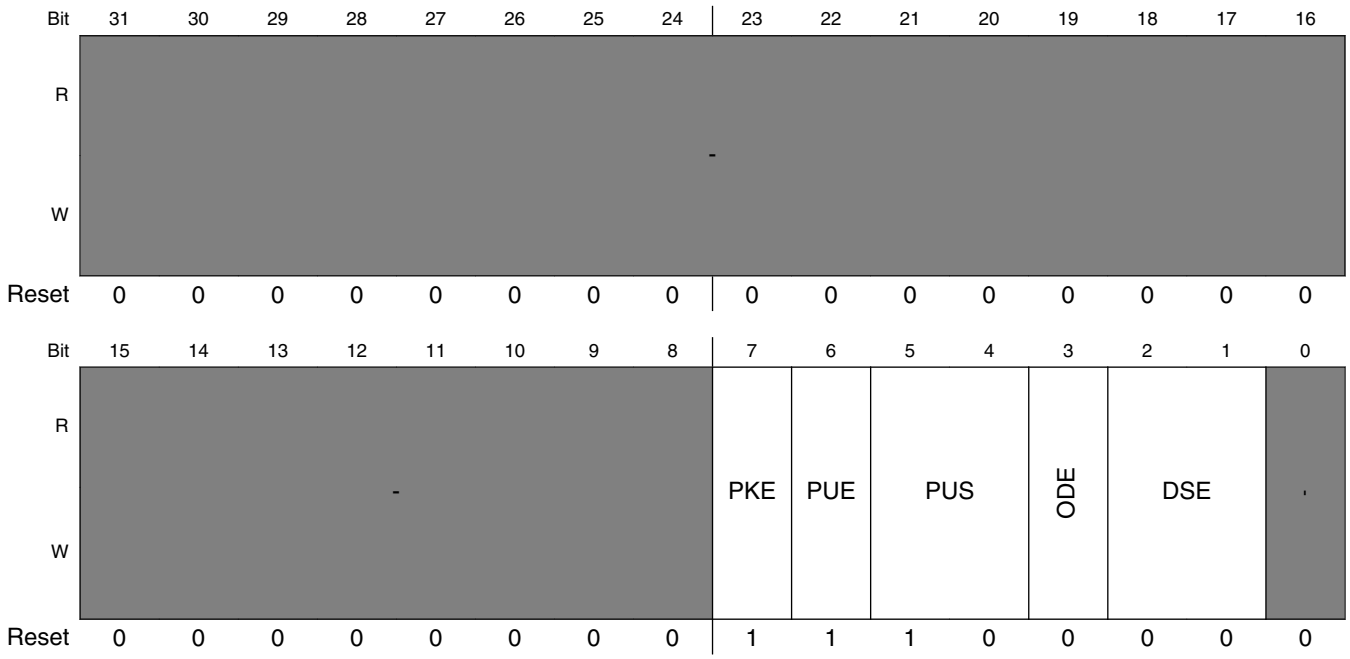
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								PKE	PUE	PUS		ODE	DSE		-
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_EB1 field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_EB1. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_EB1. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_EB1. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_EB1. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_EB1. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.404 SW_PAD_CTL_PAD_EIM_WAIT (IOMUXC_SPADC_PEIM_WAIT)

Address: IOMUXC_SPADC_PEIM_WAIT is 53FA_8000h base + 64Ch offset = 53FA_864Ch



IOMUXC_SPADC_PEIM_WAIT field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_WAIT. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_WAIT. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_WAIT. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_WAIT field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_WAIT. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_WAIT. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.405 SW_PAD_CTL_PAD_EIM_BCLK (IOMUXC_SPADC_PEIM_BCLK)

Address: IOMUXC_SPADC_PEIM_BCLK is 53FA_8000h base + 650h offset = 53FA_8650h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

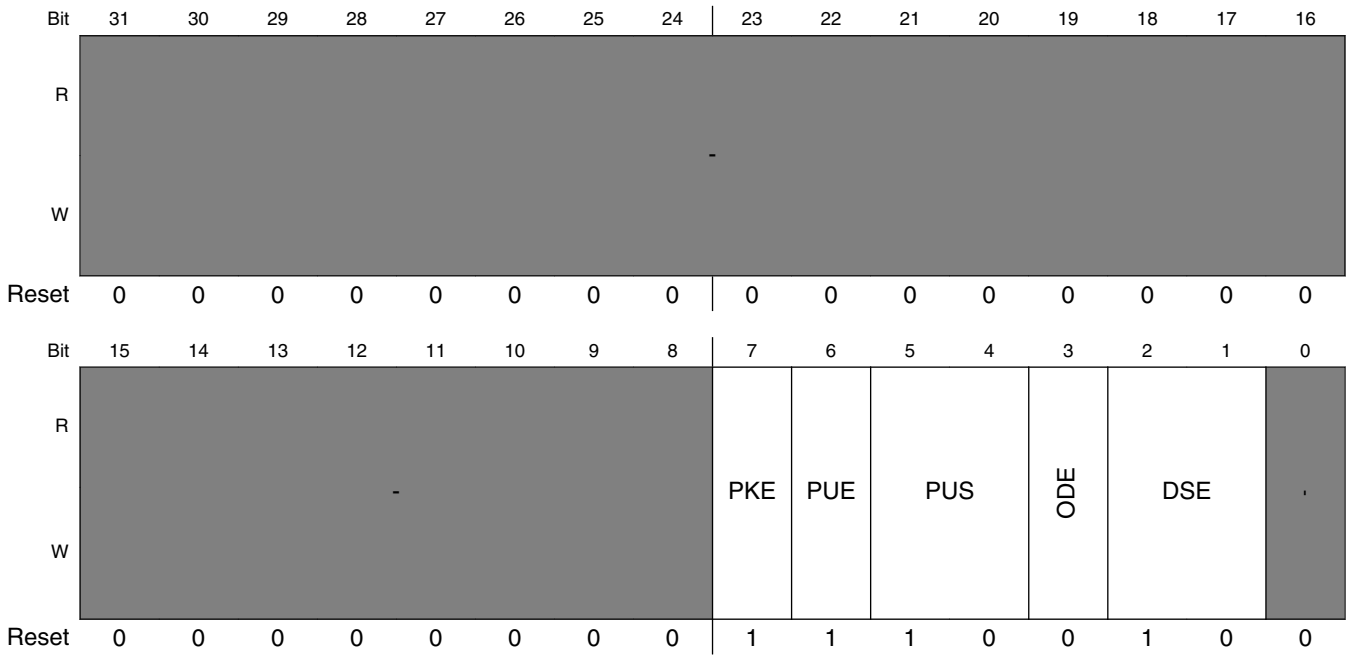
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_BCLK field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_BCLK. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_BCLK. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_BCLK. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_BCLK. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_BCLK. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.406 SW_PAD_CTL_PAD_EIM_RDY (IOMUXC_SPADC_PEIM_RDY)

Address: IOMUXC_SPADC_PEIM_RDY is 53FA_8000h base + 654h offset = 53FA_8654h



IOMUXC_SPADC_PEIM_RDY field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_RDY. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_RDY. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_RDY. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

Table continues on the next page...

IOMUXC_SPADC_PEIM_RDY field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_RDY. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_RDY. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.407 SW_PAD_CTL_PAD_EIM_OE (IOMUXC_SPADC_PEIM_OE)

Address: IOMUXC_SPADC_PEIM_OE is 53FA_8000h base + 658h offset = 53FA_8658h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_OE field descriptions

Field	Description
31–8 -	Reserved

Table continues on the next page...

IOMUXC_SPADC_PEIM_OE field descriptions (continued)

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_OE. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_OE. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_OE. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_OE. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_OE. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.408 SW_PAD_CTL_PAD_EIM_RW (IOMUXC_SPADC_PEIM_RW)

Address: IOMUXC_SPADC_PEIM_RW is 53FA_8000h base + 65Ch offset = 53FA_865Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_RW field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_RW. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_RW. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_RW. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field

Table continues on the next page...

IOMUXC_SPADC_PEIM_RW field descriptions (continued)

Field	Description
	Select one out of next values for pad: EIM_RW. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_RW. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.409 SW_PAD_CTL_PAD_EIM_LBA (IOMUXC_SPADC_PEIM_LBA)

Address: IOMUXC_SPADC_PEIM_LBA is 53FA_8000h base + 660h offset = 53FA_8660h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-								PKE	PUE	PUS		ODE	DSE		-
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0

IOMUXC_SPADC_PEIM_LBA field descriptions

Field	Description
31–8 -	Reserved

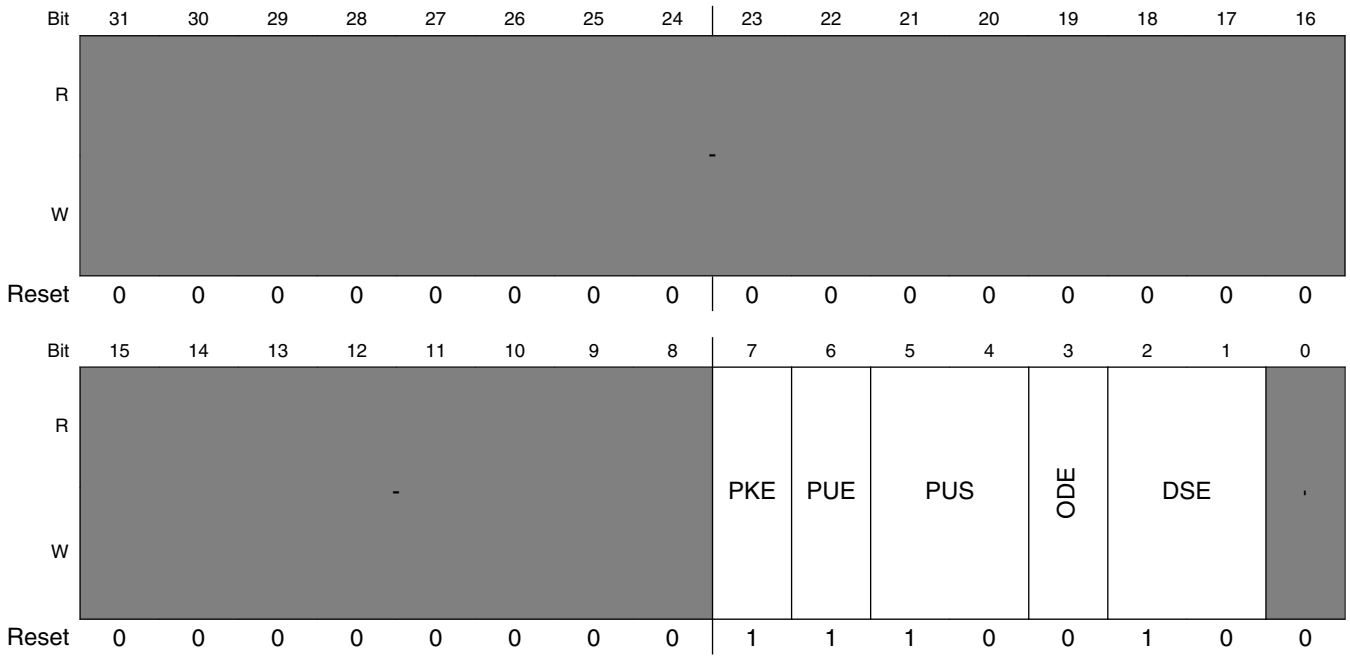
Table continues on the next page...

IOMUXC_SPADC_PEIM_LBA field descriptions (continued)

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_LBA. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_LBA. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_LBA. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_LBA. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_LBA. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.410 SW_PAD_CTL_PAD_EIM_CRE (IOMUXC_SPADC_PEIM_CRE)

Address: IOMUXC_SPADC_PEIM_CRE is 53FA_8000h base + 664h offset = 53FA_8664h



IOMUXC_SPADC_PEIM_CRE field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EIM_CRE. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: EIM_CRE. 0 Keeper 1 Pull
5–4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: EIM_CRE. 00 100 kΩ Pull Down 01 47 kΩ Pull Up 10 100 kΩ Pull Up 11 22 kΩ Pull Up

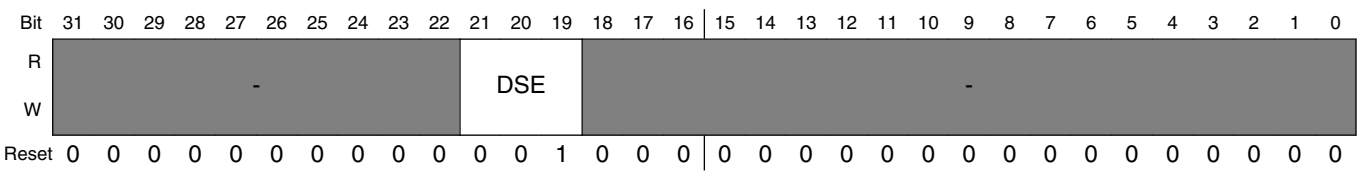
Table continues on the next page...

IOMUXC_SPADC_PEIM_CRE field descriptions (continued)

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: EIM_CRE. 0 Open Drain Disabled 1 Open Drain Enabled
2–1 DSE	Drive Strength Field Select one out of next values for pad: EIM_CRE. 00 Low Drive Strength 01 Medium Drive Strength 10 High Drive Strength (same as MAX if strength_mode=0, 3 level strength) 11 Max Drive Strength
0 -	Reserved

35.4.411 SW_PAD_CTL_GRP_ADDDS (IOMUXC_SPAD_GADDDS)

Address: IOMUXC_SPAD_GADDDS is 53FA_8000h base + 668h offset = 53FA_8668h

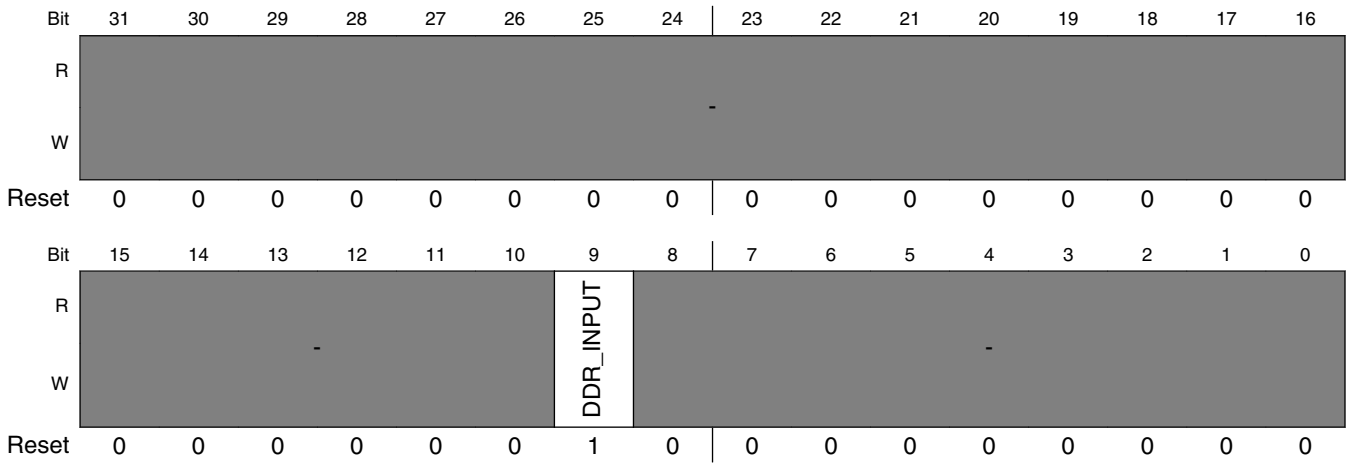


IOMUXC_SPAD_GADDDS field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for group: ADDDS (Pads: DRAM_A0 DRAM_A1 DRAM_A10 DRAM_A11 DRAM_A12 DRAM_A13 DRAM_A14 DRAM_A2 DRAM_A3 DRAM_A4 DRAM_A5 DRAM_A6 DRAM_A7 DRAM_A8 DRAM_A9 DRAM_SDBA0 DRAM_SDBA1 DRAM_SDBA2). 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.412 SW_PAD_CTL_GRP_DDRMODE_CTL (IOMUXC_SPAD_GDDRMODE_CTL)

Address: IOMUXC_SPAD_GDDRMODE_CTL is 53FA_8000h base + 66Ch offset = 53FA_866Ch

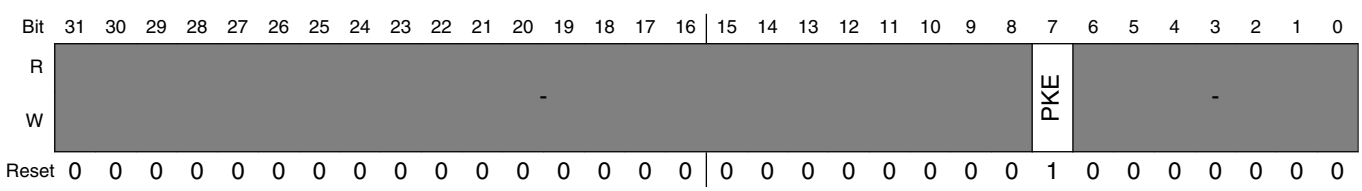


IOMUXC_SPAD_GDDRMODE_CTL field descriptions

Field	Description
31–10 -	Reserved
9 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for group: DDRMODE_CTL (Pads: DRAM_SDQS0 DRAM_SDQS1 DRAM_SDQS2 DRAM_SDQS3). 0 CMOS input type 1 differential input mode
8–0 -	Reserved

35.4.413 SW_PAD_CTL_GRP_DDRPKE (IOMUXC_SPAD_GDDRPKE)

Address: IOMUXC_SPAD_GDDRPKE is 53FA_8000h base + 670h offset = 53FA_8670h

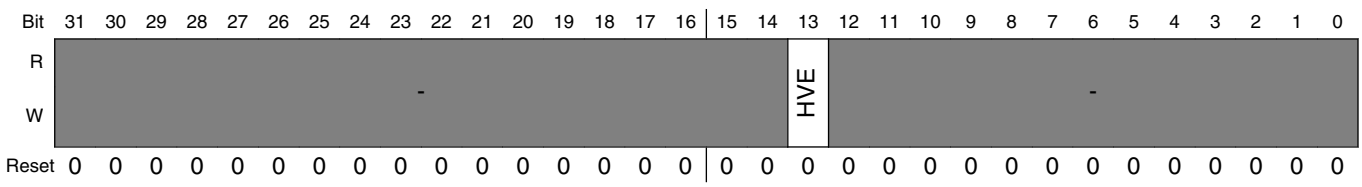


IOMUXC_SPAD_GDDRPKE field descriptions

Field	Description
31–8 -	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for group: DDRPKE (Pads: DRAM_A0 DRAM_A1 DRAM_A10 DRAM_A11 DRAM_A12 DRAM_A13 DRAM_A14 DRAM_A2 DRAM_A3 DRAM_A4 DRAM_A5 DRAM_A6 DRAM_A7 DRAM_A8 DRAM_A9 DRAM_CAS DRAM_CS0 DRAM_CS1 DRAM_D0 DRAM_D1 DRAM_D10 DRAM_D11 DRAM_D12 DRAM_D13 DRAM_D14 DRAM_D15 DRAM_D16 DRAM_D17 DRAM_D18 DRAM_D19 DRAM_D2 DRAM_D20 DRAM_D21 DRAM_D22 DRAM_D23 DRAM_D24 DRAM_D25 DRAM_D26 DRAM_D27 DRAM_D28 DRAM_D29 DRAM_D3 DRAM_D30 DRAM_D31 DRAM_D4 DRAM_D5 DRAM_D6 DRAM_D7 DRAM_D8 DRAM_D9 DRAM_DQM0 DRAM_DQM1 DRAM_DQM2 DRAM_DQM3 DRAM_OPEN DRAM_OPENFB DRAM_RAS DRAM_SDBA0 DRAM_SDBA1 DRAM_SDBA2 DRAM_SDCLK_0 DRAM_SDCLK_1 DRAM_SDWE). 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6–0 -	Reserved

35.4.414 SW_PAD_CTL_GRP_EIM (IOMUXC_SPAD_GEIM)

Address: IOMUXC_SPAD_GEIM is 53FA_8000h base + 674h offset = 53FA_8674h

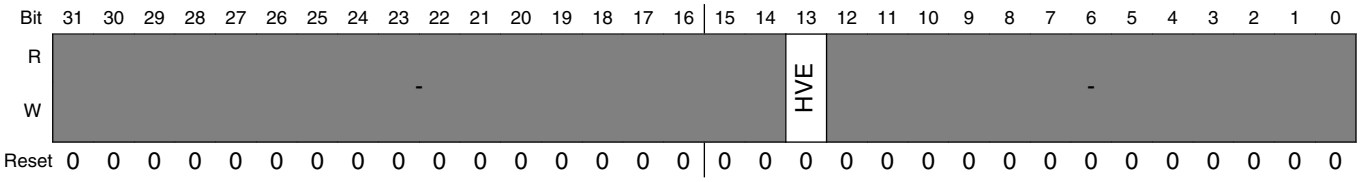


IOMUXC_SPAD_GEIM field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: EIM (Pads: EIM_BCLK EIM_CRE EIM_CS0 EIM_CS1 EIM_CS2 EIM_DA0 EIM_DA1 EIM_DA10 EIM_DA11 EIM_DA12 EIM_DA13 EIM_DA14 EIM_DA15 EIM_DA2 EIM_DA3 EIM_DA4 EIM_DA5 EIM_DA6 EIM_DA7 EIM_DA8 EIM_DA9 EIM_EB0 EIM_EB1 EIM_LBA EIM_OE EIM_RDY EIM_RW EIM_WAIT). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.415 SW_PAD_CTL_GRP_EPDC (IOMUXC_SPAD_GEPDC)

Address: IOMUXC_SPAD_GEPDC is 53FA_8000h base + 678h offset = 53FA_8678h

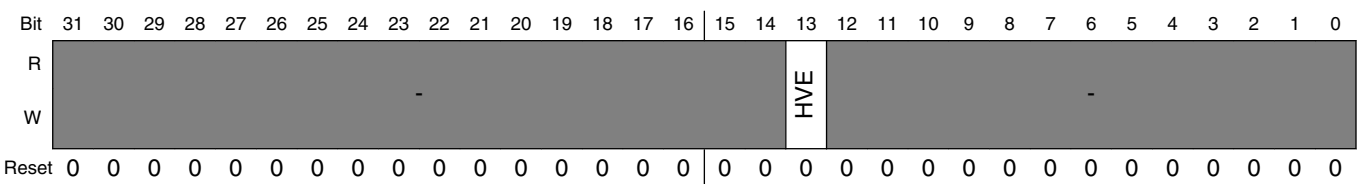


IOMUXC_SPAD_GEPDC field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: EPDC (Pads: EPDC_BDR0 EPDC_BDR1 EPDC_D0 EPDC_D1 EPDC_D10 EPDC_D11 EPDC_D12 EPDC_D13 EPDC_D14 EPDC_D15 EPDC_D2 EPDC_D3 EPDC_D4 EPDC_D5 EPDC_D6 EPDC_D7 EPDC_D8 EPDC_D9 EPDC_GDCLK EPDC_GDOE EPDC_GDRL EPDC_GDSP EPDC_PWRCOM EPDC_PWRCTRL0 EPDC_PWRCTRL1 EPDC_PWRCTRL2 EPDC_PWRCTRL3 EPDC_PWRSTAT EPDC_SDCE0 EPDC_SDCE1 EPDC_SDCE2 EPDC_SDCE3 EPDC_SDCE4 EPDC_SDCE5 EPDC_SDCLK EPDC_SDCLKN EPDC_SDLE EPDC_SDOE EPDC_SDOED EPDC_SDOEZ EPDC_SDSHR EPDC_VCOM0 EPDC_VCOM1). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.416 SW_PAD_CTL_GRP_UART (IOMUXC_SPAD_GUART)

Address: IOMUXC_SPAD_GUART is 53FA_8000h base + 67Ch offset = 53FA_867Ch



IOMUXC_SPAD_GUART field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field

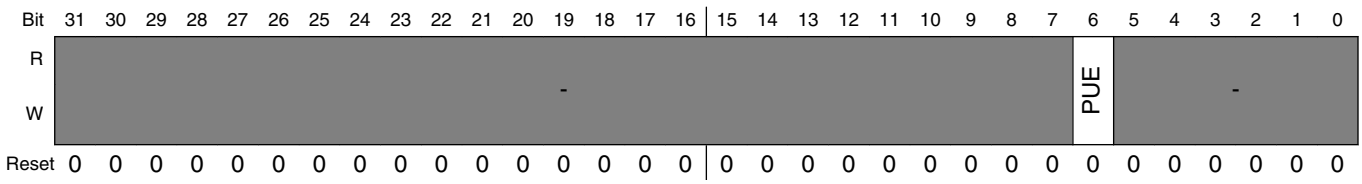
Table continues on the next page...

IOMUXC_SPAD_GUART field descriptions (continued)

Field	Description
	Select one out of next values for group: UART (Pads: UART1_CTS UART1_RTS UART1_RXD UART1_TXD UART2_CTS UART2_RTS UART2_RXD UART2_TXD UART3_RXD UART3_TXD UART4_RXD UART4_TXD). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.417 SW_PAD_CTL_GRP_DDRPK (IOMUXC_SPAD_GDDRPK)

Address: IOMUXC_SPAD_GDDRPK is 53FA_8000h base + 680h offset = 53FA_8680h

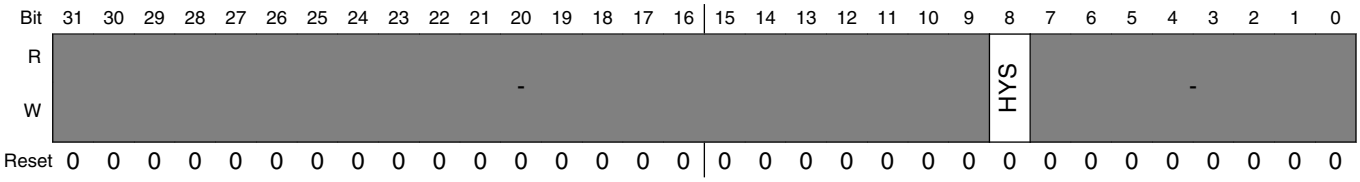


IOMUXC_SPAD_GDDRPK field descriptions

Field	Description
31–7 -	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for group: DDRPK (Pads: DRAM_A0 DRAM_A1 DRAM_A10 DRAM_A11 DRAM_A12 DRAM_A13 DRAM_A14 DRAM_A2 DRAM_A3 DRAM_A4 DRAM_A5 DRAM_A6 DRAM_A7 DRAM_A8 DRAM_A9 DRAM_CAS DRAM_CS0 DRAM_CS1 DRAM_D0 DRAM_D1 DRAM_D10 DRAM_D11 DRAM_D12 DRAM_D13 DRAM_D14 DRAM_D15 DRAM_D16 DRAM_D17 DRAM_D18 DRAM_D19 DRAM_D2 DRAM_D20 DRAM_D21 DRAM_D22 DRAM_D23 DRAM_D24 DRAM_D25 DRAM_D26 DRAM_D27 DRAM_D28 DRAM_D29 DRAM_D3 DRAM_D30 DRAM_D31 DRAM_D4 DRAM_D5 DRAM_D6 DRAM_D7 DRAM_D8 DRAM_D9 DRAM_DQM0 DRAM_DQM1 DRAM_DQM2 DRAM_DQM3 DRAM_OPEN DRAM_OPENFB DRAM_RAS DRAM_SDBA0 DRAM_SDBA1 DRAM_SDBA2 DRAM_SDCLK_0 DRAM_SDCLK_1 DRAM_SDWE). 0 Keeper 1 Pull
5–0 -	Reserved

35.4.418 SW_PAD_CTL_GRP_DDRHYS (IOMUXC_SPAD_GDDRHYS)

Address: IOMUXC_SPAD_GDDRHYS is 53FA_8000h base + 684h offset = 53FA_8684h

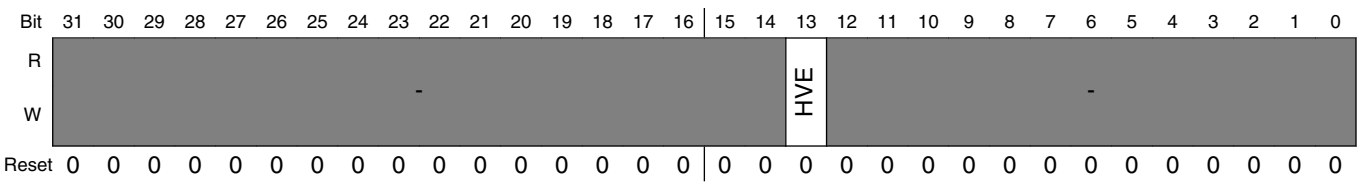


IOMUXC_SPAD_GDDRHYS field descriptions

Field	Description
31–9 -	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for group: DDRHYS (Pads: DRAM_D0 DRAM_D1 DRAM_D10 DRAM_D11 DRAM_D12 DRAM_D13 DRAM_D14 DRAM_D15 DRAM_D16 DRAM_D17 DRAM_D18 DRAM_D19 DRAM_D2 DRAM_D20 DRAM_D21 DRAM_D22 DRAM_D23 DRAM_D24 DRAM_D25 DRAM_D26 DRAM_D27 DRAM_D28 DRAM_D29 DRAM_D3 DRAM_D30 DRAM_D31 DRAM_D4 DRAM_D5 DRAM_D6 DRAM_D7 DRAM_D8 DRAM_D9 DRAM_OPENFB DRAM_SDQS0 DRAM_SDQS1 DRAM_SDQS2 DRAM_SDQS3). 0 Hysteresis Disabled 1 Hysteresis Enabled
7–0 -	Reserved

35.4.419 SW_PAD_CTL_GRP_KEYPAD (IOMUXC_SPAD_GKEYPAD)

Address: IOMUXC_SPAD_GKEYPAD is 53FA_8000h base + 688h offset = 53FA_8688h



IOMUXC_SPAD_GKEYPAD field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: KEYPAD (Pads: KEY_COL0 KEY_COL1 KEY_COL2 KEY_COL3 KEY_ROW0 KEY_ROW1 KEY_ROW2 KEY_ROW3).

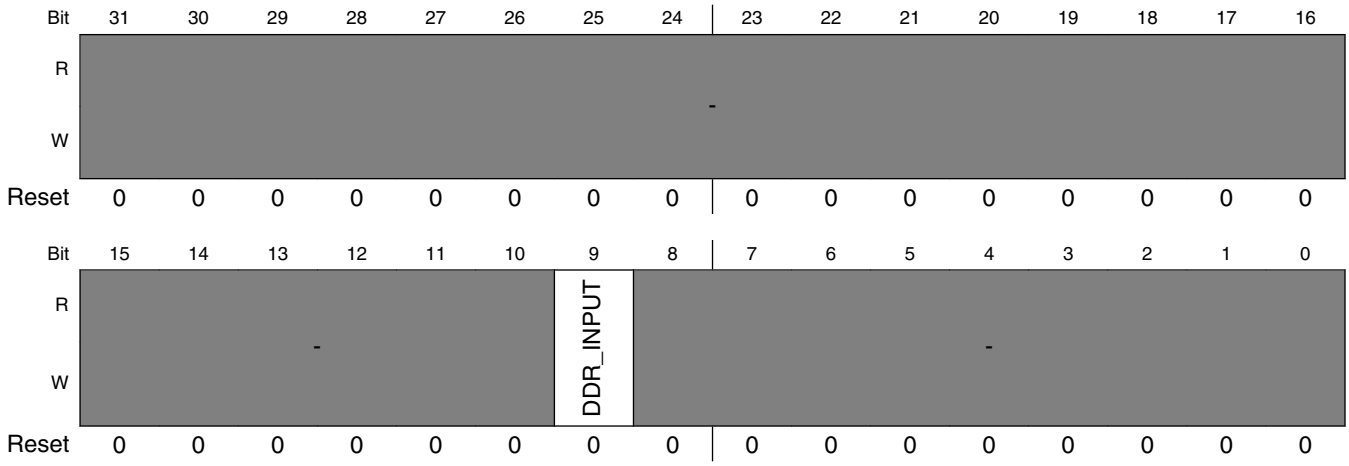
Table continues on the next page...

IOMUXC_SPAD_GKEYPAD field descriptions (continued)

Field	Description
	0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.420 SW_PAD_CTL_GRP_DDRMODE (IOMUXC_SPAD_GDDRMODE)

Address: IOMUXC_SPAD_GDDRMODE is 53FA_8000h base + 68Ch offset = 53FA_868Ch



IOMUXC_SPAD_GDDRMODE field descriptions

Field	Description
31–10 -	Reserved
9 DDR_INPUT	DDR / CMOS Input Mode Field Select one out of next values for group: DDRMODE (Pads: DRAM_D0 DRAM_D1 DRAM_D10 DRAM_D11 DRAM_D12 DRAM_D13 DRAM_D14 DRAM_D15 DRAM_D16 DRAM_D17 DRAM_D18 DRAM_D19 DRAM_D2 DRAM_D20 DRAM_D21 DRAM_D22 DRAM_D23 DRAM_D24 DRAM_D25 DRAM_D26 DRAM_D27 DRAM_D28 DRAM_D29 DRAM_D3 DRAM_D30 DRAM_D31 DRAM_D4 DRAM_D5 DRAM_D6 DRAM_D7 DRAM_D8 DRAM_D9). 0 CMOS input type 1 differential input mode
8–0 -	Reserved

35.4.421 SW_PAD_CTL_GRP_SSI (IOMUXC_SPAD_GSSI)

Address: IOMUXC_SPAD_GSSI is 53FA_8000h base + 690h offset = 53FA_8690h

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

IOMUXC_SPAD_GSSI field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: SSI (Pads: SSI_RXC SSI_RXD SSI_RXFS SSI_TXC SSI_TXD SSI_TXFS). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.422 SW_PAD_CTL_GRP_SD1 (IOMUXC_SPAD_GSD1)

Address: IOMUXC_SPAD_GSD1 is 53FA_8000h base + 694h offset = 53FA_8694h

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

IOMUXC_SPAD_GSD1 field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: SD1 (Pads: SD1_CLK SD1_CMD SD1_D0 SD1_D1 SD1_D2 SD1_D3). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.423 SW_PAD_CTL_GRP_B0DS (IOMUXC_SPAD_GB0DS)

Address: IOMUXC_SPAD_GB0DS is 53FA_8000h base + 698h offset = 53FA_8698h

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

IOMUXC_SPAD_GB0DS field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for group: B0DS (Pads: DRAM_D0 DRAM_D1 DRAM_D2 DRAM_D3 DRAM_D4 DRAM_D5 DRAM_D6 DRAM_D7). 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.424 SW_PAD_CTL_GRP_SD2 (IOMUXC_SPAD_GSD2)

Address: IOMUXC_SPAD_GSD2 is 53FA_8000h base + 69Ch offset = 53FA_869Ch

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

IOMUXC_SPAD_GSD2 field descriptions

Field	Description
31–14 -	Reserved

Table continues on the next page...

IOMUXC_SPAD_GSD2 field descriptions (continued)

Field	Description
13 HVE	low/high output voltage Field Select one out of next values for group: SD2 (Pads: SD2_CD SD2_CLK SD2_CMD SD2_D0 SD2_D1 SD2_D2 SD2_D3 SD2_D4 SD2_D5 SD2_D6 SD2_D7 SD2_WP). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.425 SW_PAD_CTL_GRP_B1DS (IOMUXC_SPAD_GB1DS)

Address: IOMUXC_SPAD_GB1DS is 53FA_8000h base + 6A0h offset = 53FA_86A0h

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

IOMUXC_SPAD_GB1DS field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for group: B1DS (Pads: DRAM_D10 DRAM_D11 DRAM_D12 DRAM_D13 DRAM_D14 DRAM_D15 DRAM_D8 DRAM_D9). 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.426 SW_PAD_CTL_GRP_CTLDS (IOMUXC_SPAD_GCTLDS)

Address: IOMUXC_SPAD_GCTLDS is 53FA_8000h base + 6A4h offset = 53FA_86A4h

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

IOMUXC_SPAD_GCTLDS field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for group: CTLDS (Pads: DRAM_CAS DRAM_CS0 DRAM_CS1 DRAM_RAS DRAM_SDCKE DRAM_SDWE). 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.427 SW_PAD_CTL_GRP_B2DS (IOMUXC_SPAD_GB2DS)

Address: IOMUXC_SPAD_GB2DS is 53FA_8000h base + 6A8h offset = 53FA_86A8h

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

IOMUXC_SPAD_GB2DS field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field

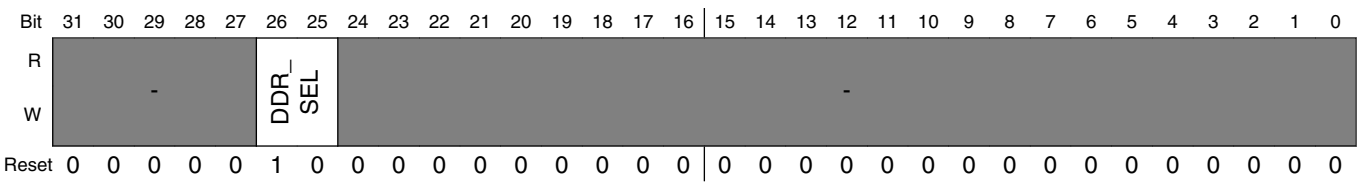
Table continues on the next page...

IOMUXC_SPAD_GB2DS field descriptions (continued)

Field	Description
	Select one out of next values for group: B2DS (Pads: DRAM_D16 DRAM_D17 DRAM_D18 DRAM_D19 DRAM_D20 DRAM_D21 DRAM_D22 DRAM_D23). 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.428 SW_PAD_CTL_GRP_DDR_TYPE (IOMUXC_SPAD_GDDR_TYPE)

Address: IOMUXC_SPAD_GDDR_TYPE is 53FA_8000h base + 6ACh offset = 53FA_86ACh



IOMUXC_SPAD_GDDR_TYPE field descriptions

Field	Description
31–27 -	Reserved
26–25 DDR_SEL	ddr_sel Field Select one out of next values for group: DDR_TYPE (Pads: DRAM_A0 DRAM_A1 DRAM_A10 DRAM_A11 DRAM_A12 DRAM_A13 DRAM_A14 DRAM_A2 DRAM_A3 DRAM_A4 DRAM_A5 DRAM_A6 DRAM_A7 DRAM_A8 DRAM_A9 DRAM_CAS DRAM_CS0 DRAM_CS1 DRAM_D0 DRAM_D1 DRAM_D10 DRAM_D11 DRAM_D12 DRAM_D13 DRAM_D14 DRAM_D15 DRAM_D16 DRAM_D17 DRAM_D18 DRAM_D19 DRAM_D2 DRAM_D20 DRAM_D21 DRAM_D22 DRAM_D23 DRAM_D24 DRAM_D25 DRAM_D26 DRAM_D27 DRAM_D28 DRAM_D29 DRAM_D3 DRAM_D30 DRAM_D31 DRAM_D4 DRAM_D5 DRAM_D6 DRAM_D7 DRAM_D8 DRAM_D9 DRAM_DQM0 DRAM_DQM1 DRAM_DQM2 DRAM_DQM3 DRAM_OPEN DRAM_OPENFB DRAM_RAS DRAM_SDBA0 DRAM_SDBA1 DRAM_SDBA2 DRAM_SDCKE DRAM_SDCLK_0 DRAM_SDCLK_1 DRAM_SDODT0 DRAM_SDODT1 DRAM_SDQS0 DRAM_SDQS1 DRAM_SDQS2 DRAM_SDQS3 DRAM_SDWE). 00 LPDDR1/ DDR2 ODT mode at 1.8V power (7x300Ohm driver unit calibration, additional driver unit is NOT connected) 01 DDR2 drive mode at 1.8V power (7x300Ohm driver unit calibration, additional driver unit is connected)

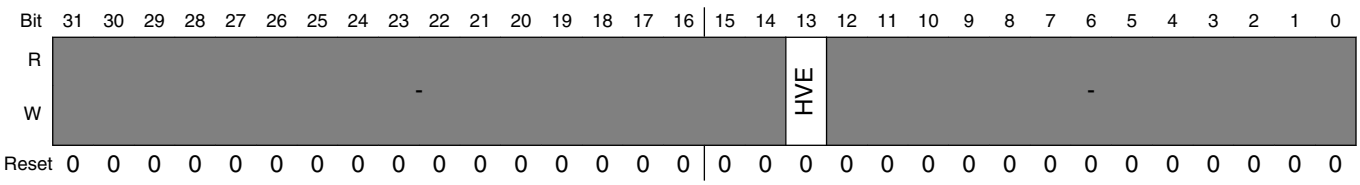
Table continues on the next page...

IOMUXC_SPAD_GDDR_TYPE field descriptions (continued)

Field	Description
	10 LPDDR2 at 1.2V power (7x240Ohm driver unit calibration, additional driver unit is connected) 11 Reserved
24–0 -	Reserved

35.4.429 SW_PAD_CTL_GRP_LCD (IOMUXC_SPAD_GLCD)

Address: IOMUXC_SPAD_GLCD is 53FA_8000h base + 6B0h offset = 53FA_86B0h

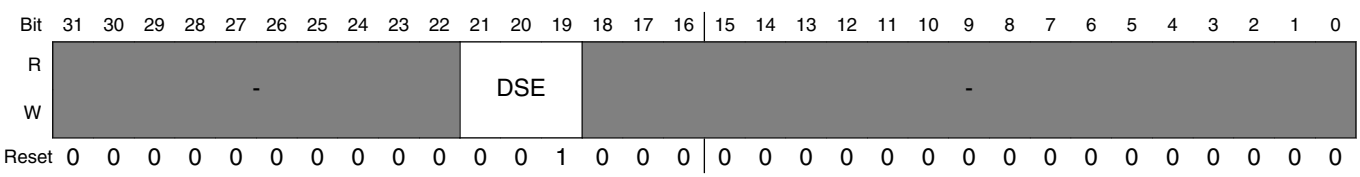


IOMUXC_SPAD_GLCD field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: LCD (Pads: DISP_BUSY DISP_CS DISP_D0 DISP_D1 DISP_D2 DISP_D3 DISP_D4 DISP_D5 DISP_D6 DISP_D7 DISP_RD DISP_RESET DISP_RS DISP_WR). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.430 SW_PAD_CTL_GRP_B3DS (IOMUXC_SPAD_GB3DS)

Address: IOMUXC_SPAD_GB3DS is 53FA_8000h base + 6B4h offset = 53FA_86B4h



IOMUXC_SPAD_GB3DS field descriptions

Field	Description
31–22 -	Reserved
21–19 DSE	Drive Strength Field Select one out of next values for group: B3DS (Pads: DRAM_D24 DRAM_D25 DRAM_D26 DRAM_D27 DRAM_D28 DRAM_D29 DRAM_D30 DRAM_D31). 000 output driver disabled 001 LPDDR2: 240 Ohm ; LPDDR1: ; DDR2: 010 LPDDR2: 240/2 = 120 Ohm; LPDDR1: ; DDR2: 011 LPDDR2: 240/3 = 80 Ohm; LPDDR1: ; DDR2: 100 LPDDR2: 240/4 = 60 Ohm; LPDDR1: 300/4 =75 Ohm; DDR2: 45 Ohm 101 LPDDR2: 240/5 = 48 Ohm; LPDDR1: ; DDR2: 110 LPDDR2: 240/6 = 40 Ohm; LPDDR1: 300/6 =50 Ohm; DDR2: 111 LPDDR2: 240/7 = 34 Ohm; LPDDR1: ; DDR2:
18–0 -	Reserved

35.4.431 SW_PAD_CTL_GRP_MISC (IOMUXC_SPAD_GMISC)

Address: IOMUXC_SPAD_GMISC is 53FA_8000h base + 6B8h offset = 53FA_86B8h

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

IOMUXC_SPAD_GMISC field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: MISC (Pads: EPITO I2C1_SCL I2C1_SDA I2C2_SCL I2C2_SDA I2C3_SCL I2C3_SDA OWIRE PWM1 PWM2 WDOG). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.432 SW_PAD_CTL_GRP_SPI (IOMUXC_SPAD_GSPI)

Address: IOMUXC_SPAD_GSPI is 53FA_8000h base + 6BCh offset = 53FA_86BCh

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

IOMUXC_SPAD_GSPI field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: SPI (Pads: CSPI_MISO CSPI_MOSI CSPI_SCLK CSPI_SS0 ECSPI1_MISO ECSPI1_MOSI ECSPI1_SCLK ECSPI1_SS0 ECSPI2_MISO ECSPI2_MOSI ECSPI2_SCLK ECSPI2_SS0). 0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.433 SW_PAD_CTL_GRP_NANDF (IOMUXC_SPAD_GNANDF)

Address: IOMUXC_SPAD_GNANDF is 53FA_8000h base + 6C0h offset = 53FA_86C0h

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

IOMUXC_SPAD_GNANDF field descriptions

Field	Description
31–14 -	Reserved
13 HVE	low/high output voltage Field Select one out of next values for group: NANDF (Pads: DISP_D10 DISP_D11 DISP_D12 DISP_D13 DISP_D14 DISP_D15 DISP_D8 DISP_D9 SD3_CLK SD3_CMD SD3_D0 SD3_D1 SD3_D2 SD3_D3 SD3_D4 SD3_D5 SD3_D6 SD3_D7 SD3_WP).

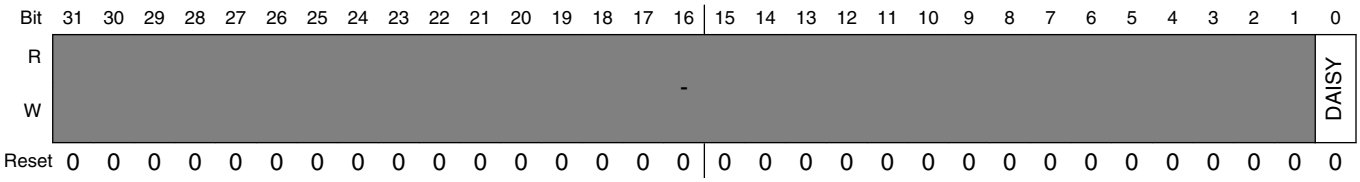
Table continues on the next page...

IOMUXC_SPAD_GNANDF field descriptions (continued)

Field	Description
	0 High output voltage (2.7V-3.3V) 1 Low output voltage (1.65V-1.95V)
12–0 -	Reserved

35.4.434 AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT (IOMUXC_API_DA_AMX_SI)

Address: IOMUXC_API_DA_AMX_SI is 53FA_8000h base + 6C4h offset = 53FA_86C4h

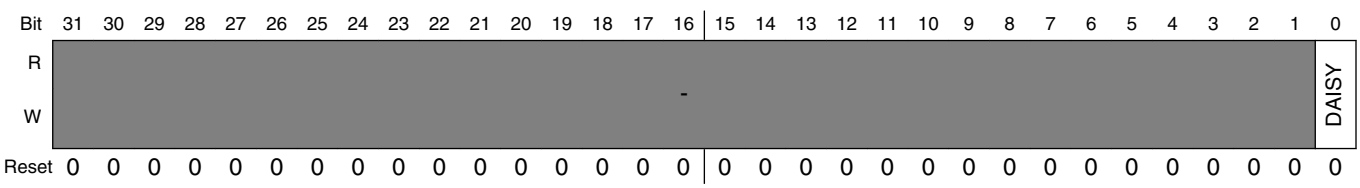


IOMUXC_API_DA_AMX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_da_amx 0 Selecting Pad: SD2_D6 for Mode: ALT2. 1 Selecting Pad: EPDC_PWRCTRL0 for Mode: ALT4.

35.4.435 AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT (IOMUXC_API_DB_AMX_SI)

Address: IOMUXC_API_DB_AMX_SI is 53FA_8000h base + 6C8h offset = 53FA_86C8h



IOMUXC_API_DB_AMX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_db_amx 0 Selecting Pad: SD2_WP for Mode: ALT2. 1 Selecting Pad: EPDC_SDSHR for Mode: ALT4.

35.4.436 AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT (IOMUXC_API_RXCLK_AMX_SI)

Address: IOMUXC_API_RXCLK_AMX_SI is 53FA_8000h base + 6CCh offset = 53FA_86CCh

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

IOMUXC_API_RXCLK_AMX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_rxclk_amx 0 Selecting Pad: SD2_D5 for Mode: ALT2. 1 Selecting Pad: EPDC_PWRCTRL1 for Mode: ALT4.

35.4.437 AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT (IOMUXC_API_RXFS_AMX_SI)

Address: IOMUXC_API_RXFS_AMX_SI is 53FA_8000h base + 6D0h offset = 53FA_86D0h

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

IOMUXC_API_RXFS_AMX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_rxfs_amx 0 Selecting Pad: SD2_D4 for Mode: ALT2. 1 Selecting Pad: EPDC_PWRCTRL2 for Mode: ALT4.

35.4.438 AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT (IOMUXC_API_TXCLK_AMX_SI)

Address: IOMUXC_API_TXCLK_AMX_SI is 53FA_8000h base + 6D4h offset = 53FA_86D4h

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

IOMUXC_API_TXCLK_AMX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_txclk_amx 0 Selecting Pad: SD2_CD for Mode: ALT2. 1 Selecting Pad: EPDC_PWRCOM for Mode: ALT4.

35.4.439 AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT (IOMUXC_API_TXFS_AMX_SI)

Address: IOMUXC_API_TXFS_AMX_SI is 53FA_8000h base + 6D8h offset = 53FA_86D8h

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

IOMUXC_API_TXFS_AMX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_txfs_amx 0 Selecting Pad: SD2_D7 for Mode: ALT2. 1 Selecting Pad: EPDC_PWRSTAT for Mode: ALT4.

35.4.440 CCM_PLL1_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL1_BYPASS_CLK_SI)

Address: IOMUXC_CCM_PLL1_BYPASS_CLK_SI is 53FA_8000h base + 6DCh offset = 53FA_86DCh

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

IOMUXC_CCM_PLL1_BYPASS_CLK_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ccm, In Pin: pll1_bypass_clk 0 Selecting Pad: SD1_D0 for Mode: ALT7. 1 Selecting Pad: SD3_D0 for Mode: ALT7.

35.4.441 CCM_PLL2_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL2_BYPASS_CLK_SI)

Address: IOMUXC_CCM_PLL2_BYPASS_CLK_SI is 53FA_8000h base + 6E0h offset = 53FA_86E0h

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

IOMUXC_CCM_PLL2_BYPASS_CLK_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ccm, In Pin: pll2_bypass_clk 0 Selecting Pad: SD1_D1 for Mode: ALT7. 1 Selecting Pad: SD3_D1 for Mode: ALT7.

35.4.442 CCM_PLL3_BYPASS_CLK_SELECT_INPUT (IOMUXC_CCM_PLL3_BYPASS_CLK_SI)

Address: IOMUXC_CCM_PLL3_BYPASS_CLK_SI is 53FA_8000h base + 6E4h offset = 53FA_86E4h

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

IOMUXC_CCM_PLL3_BYPASS_CLK_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: ccm, In Pin: pll3_bypass_clk 0 Selecting Pad: SD1_D2 for Mode: ALT7. 1 Selecting Pad: SD3_D2 for Mode: ALT7.

35.4.443 CSPI_IPP_IND_DATAREADY_B_SELECT_INPUT (IOMUXC_CSPI_DRDY_SI)

Address: IOMUXC_CSPI_DRDY_SI is 53FA_8000h base + 6E8h offset = 53FA_86E8h

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

IOMUXC_CSPI_DRDY_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi, In Pin: ipp_ind_dataready_b 0 Selecting Pad: SSI_TXD for Mode: ALT4. 1 Selecting Pad: ECSPI1_SCLK for Mode: ALT2.

35.4.444 CSPI_IPP_IND_SS1_B_SELECT_INPUT (IOMUXC_CSPI_SS1_SI)

Address: IOMUXC_CSPI_SS1_SI is 53FA_8000h base + 6ECh offset = 53FA_86ECh

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

IOMUXC_CSPI_SS1_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi, In Pin: ipp_ind_ss1_b 0 Selecting Pad: SSI_RXC for Mode: ALT4. 1 Selecting Pad: ECSPI1_MOSI for Mode: ALT2.

35.4.445 CSPI_IPP_IND_SS2_B_SELECT_INPUT (IOMUXC_CSPI_SS2_SI)

Address: IOMUXC_CSPI_SS2_SI is 53FA_8000h base + 6F0h offset = 53FA_86F0h

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

IOMUXC_CSPI_SS2_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi, In Pin: ipp_ind_ss2_b 0 Selecting Pad: SSI_RXFS for Mode: ALT4. 1 Selecting Pad: ECSPI1_MISO for Mode: ALT2.

35.4.446 CSPI_IPP_IND_SS3_B_SELECT_INPUT (IOMUXC_CSPI_SS3_SI)

Address: IOMUXC_CSPI_SS3_SI is 53FA_8000h base + 6F4h offset = 53FA_86F4h

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

IOMUXC_CSPI_SS3_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi, In Pin: ipp_ind_ss3_b 0 Selecting Pad: SSI_RXD for Mode: ALT4. 1 Selecting Pad: ECSPI1_SS0 for Mode: ALT2.

35.4.447 ELCDIF_LCDIF_BUSY_SELECT_INPUT (IOMUXC_ELCDIFL_BUSY_SI)

Address: IOMUXC_ELCDIFL_BUSY_SI is 53FA_8000h base + 6F8h offset = 53FA_86F8h

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

IOMUXC_ELCDIFL_BUSY_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_busy 00 Selecting Pad: ECSPI2_SS0 for Mode: ALT5. 01 Selecting Pad: DISP_CS for Mode: ALT2. 10 Selecting Pad: DISP_BUSY for Mode: ALT0. 11 Selecting Pad: EPDC_D3 for Mode: ALT4.

35.4.448 ELCDIF_LCDIF_RXDATA_0_SELECT_INPUT (IOMUXC_ELCDIFL_R0_SI)

Address: IOMUXC_ELCDIFL_R0_SI is 53FA_8000h base + 6FCh offset = 53FA_86FCh

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

IOMUXC_ELCDIFL_R0_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[0] 0 Selecting Pad: DISP_D0 for Mode: ALT0. 1 Selecting Pad: EPDC_SDCE5 for Mode: ALT3.

35.4.449 ELCDIF_LCDIF_RXDATA_1_SELECT_INPUT (IOMUXC_ELCDIFL_R1_SI)

Address: IOMUXC_ELCDIFL_R1_SI is 53FA_8000h base + 700h offset = 53FA_8700h

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

IOMUXC_ELCDIFL_R1_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[1] 0 Selecting Pad: DISP_D1 for Mode: ALT0. 1 Selecting Pad: EPDC_SDCE4 for Mode: ALT3.

35.4.450 ELCDIF_LCDIF_RXDATA_2_SELECT_INPUT (IOMUXC_ELCDIFL_R2_SI)

Address: IOMUXC_ELCDIFL_R2_SI is 53FA_8000h base + 704h offset = 53FA_8704h

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

IOMUXC_ELCDIFL_R2_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[2] 0 Selecting Pad: DISP_D2 for Mode: ALT0. 1 Selecting Pad: EPDC_SDCE3 for Mode: ALT3.

35.4.451 ELCDIF_LCDIF_RXDATA_3_SELECT_INPUT (IOMUXC_ELCDIFL_R3_SI)

Address: IOMUXC_ELCDIFL_R3_SI is 53FA_8000h base + 708h offset = 53FA_8708h

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

IOMUXC_ELCDIFL_R3_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[3] 0 Selecting Pad: DISP_D3 for Mode: ALT0. 1 Selecting Pad: EPDC_SDCE2 for Mode: ALT3.

35.4.452 ELCDIF_LCDIF_RXDATA_4_SELECT_INPUT (IOMUXC_ELCDIFL_R4_SI)

Address: IOMUXC_ELCDIFL_R4_SI is 53FA_8000h base + 70Ch offset = 53FA_870Ch

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

IOMUXC_ELCDIFL_R4_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[4] 0 Selecting Pad: DISP_D4 for Mode: ALT0. 1 Selecting Pad: EPDC_SDCE1 for Mode: ALT3.

35.4.453 ELCDIF_LCDIF_RXDATA_5_SELECT_INPUT (IOMUXC_ELCDIFL_R5_SI)

Address: IOMUXC_ELCDIFL_R5_SI is 53FA_8000h base + 710h offset = 53FA_8710h

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

IOMUXC_ELCDIFL_R5_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[5] 0 Selecting Pad: DISP_D5 for Mode: ALT0. 1 Selecting Pad: EPDC_SDCE0 for Mode: ALT3.

35.4.454 ELCDIF_LCDIF_RXDATA_6_SELECT_INPUT (IOMUXC_ELCDIFL_R6_SI)

Address: IOMUXC_ELCDIFL_R6_SI is 53FA_8000h base + 714h offset = 53FA_8714h

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

IOMUXC_ELCDIFL_R6_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[6] 0 Selecting Pad: DISP_D6 for Mode: ALT0. 1 Selecting Pad: EPDC_BDR1 for Mode: ALT3.

35.4.455 ELCDIF_LCDIF_RXDATA_7_SELECT_INPUT (IOMUXC_ELCDIFL_R7_SI)

Address: IOMUXC_ELCDIFL_R7_SI is 53FA_8000h base + 718h offset = 53FA_8718h

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

IOMUXC_ELCDIFL_R7_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[7] 0 Selecting Pad: DISP_D7 for Mode: ALT0. 1 Selecting Pad: EPDC_BDR0 for Mode: ALT3.

35.4.456 ELCDIF_LCDIF_RXDATA_8_SELECT_INPUT (IOMUXC_ELCDIFL_R8_SI)

Address: IOMUXC_ELCDIFL_R8_SI is 53FA_8000h base + 71Ch offset = 53FA_871Ch

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

IOMUXC_ELCDIFL_R8_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[8] 0 Selecting Pad: DISP_D8 for Mode: ALT0. 1 Selecting Pad: EPDC_SDLE for Mode: ALT3.

35.4.457 C_ELCDIF_LCDIF_RXDATA_9_SELECT_INPUT (IOMUXC_C_ELCDIFL_R9_SI)

Address: IOMUXC_C_ELCDIFL_R9_SI is 53FA_8000h base + 720h offset = 53FA_8720h

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

IOMUXC_C_ELCDIFL_R9_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[9] 0 Selecting Pad: DISP_D9 for Mode: ALT0. 1 Selecting Pad: EPDC_SDCLKN for Mode: ALT3.

35.4.458 ELCDIF_LCDIF_RXDATA_10_SELECT_INPUT (IOMUXC_ELCDIFL_R10_SI)

Address: IOMUXC_ELCDIFL_R10_SI is 53FA_8000h base + 724h offset = 53FA_8724h

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

IOMUXC_ELCDIFL_R10_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[10] 0 Selecting Pad: DISP_D10 for Mode: ALT0. 1 Selecting Pad: EPDC_SDSHR for Mode: ALT3.

35.4.459 ELCDIF_LCDIF_RXDATA_11_SELECT_INPUT (IOMUXC_ELCDIFL_R11_SI)

Address: IOMUXC_ELCDIFL_R11_SI is 53FA_8000h base + 728h offset = 53FA_8728h

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

IOMUXC_ELCDIFL_R11_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[11] 0 Selecting Pad: DISP_D11 for Mode: ALT0. 1 Selecting Pad: EPDC_PWRCOM for Mode: ALT3.

35.4.460 ELCDIF_LCDIF_RXDATA_12_SELECT_INPUT (IOMUXC_ELCDIFL_R12_SI)

Address: IOMUXC_ELCDIFL_R12_SI is 53FA_8000h base + 72Ch offset = 53FA_872Ch

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

IOMUXC_ELCDIFL_R12_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[12] 0 Selecting Pad: DISP_D12 for Mode: ALT0. 1 Selecting Pad: EPDC_PWRSTAT for Mode: ALT3.

35.4.461 ELCDIF_LCDIF_RXDATA_13_SELECT_INPUT (IOMUXC_ELCDIFL_R13_SI)

Address: IOMUXC_ELCDIFL_R13_SI is 53FA_8000h base + 730h offset = 53FA_8730h

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

IOMUXC_ELCDIFL_R13_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[13] 0 Selecting Pad: DISP_D13 for Mode: ALT0. 1 Selecting Pad: EPDC_PWRCTRL0 for Mode: ALT3.

35.4.462 ELCDIF_LCDIF_RXDATA_14_SELECT_INPUT (IOMUXC_ELCDIFL_R14_SI)

Address: IOMUXC_ELCDIFL_R14_SI is 53FA_8000h base + 734h offset = 53FA_8734h

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

IOMUXC_ELCDIFL_R14_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[14] 0 Selecting Pad: DISP_D14 for Mode: ALT0. 1 Selecting Pad: EPDC_PWRCTRL1 for Mode: ALT3.

35.4.463 ELCDIF_LCDIF_RXDATA_15_SELECT_INPUT (IOMUXC_ELCDIFL_R15_SI)

Address: IOMUXC_ELCDIFL_R15_SI is 53FA_8000h base + 738h offset = 53FA_8738h

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

IOMUXC_ELCDIFL_R15_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: lcdif_rxdata[15] 0 Selecting Pad: DISP_D15 for Mode: ALT0. 1 Selecting Pad: EPDC_PWRCTRL2 for Mode: ALT3.

35.4.464 ELCDIF_VSYNC_I_SELECT_INPUT (IOMUXC_ELCDIF_VSYNC_I_SI)

Address: IOMUXC_ELCDIF_VSYNC_I_SI is 53FA_8000h base + 73Ch offset = 53FA_873Ch

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

IOMUXC_ELCDIF_VSYNC_I_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: elcdif, In Pin: vsync_i 00 Selecting Pad: ECSPI2_MISO for Mode: ALT5. 01 Selecting Pad: DISP_RS for Mode: ALT2. 10 Selecting Pad: EPDC_D2 for Mode: ALT4.

35.4.465 ESDHC2_IPP_CARD_DET_SELECT_INPUT (IOMUXC_ESDHC2_CDET_SI)

Address: IOMUXC_ESDHC2_CDET_SI is 53FA_8000h base + 740h offset = 53FA_8740h

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

IOMUXC_ESDHC2_CDET_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_card_det 0 Selecting Pad: UART3_RXD for Mode: ALT5. 1 Selecting Pad: SD2_CD for Mode: ALT0.

35.4.466 ESDHC2_IPP_WP_ON_SELECT_INPUT (IOMUXC_ESDHC2_WP_ON_SI)

Address: IOMUXC_ESDHC2_WP_ON_SI is 53FA_8000h base + 744h offset = 53FA_8744h

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

IOMUXC_ESDHC2_WP_ON_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_wp_on 0 Selecting Pad: UART3_TXD for Mode: ALT5. 1 Selecting Pad: SD2_WP for Mode: ALT0.

35.4.467 ESDHC4_IPP_CARD_CLK_IN_SELECT_INPUT (IOMUXC_ESDHC4_CCLK_IN_SI)

Address: IOMUXC_ESDHC4_CCLK_IN_SI is 53FA_8000h base + 748h offset = 53FA_8748h

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

IOMUXC_ESDHC4_CCLK_IN_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_card_clk_in 00 Selecting Pad: UART1_RTS for Mode: ALT5. 01 Selecting Pad: UART2_RTS for Mode: ALT4. 10 Selecting Pad: DISP_D9 for Mode: ALT4

35.4.468 ESDHC4_IPP_CMD_IN_SELECT_INPUT (IOMUXC_ESDHC4_CMD_IN_SI)

Address: IOMUXC_ESDHC4_CMD_IN_SI is 53FA_8000h base + 74Ch offset = 53FA_874Ch

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

IOMUXC_ESDHC4_CMD_IN_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_cmd_in 00 Selecting Pad: UART1_CTS for Mode: ALT5. 01 Selecting Pad: UART2_CTS for Mode: ALT4. 10 Selecting Pad: DISP_D8 for Mode: ALT4.

35.4.469 ESDHC4_IPP_DAT0_IN_SELECT_INPUT (IOMUXC_ESDHC4_DAT0_IN_SI)

Address: IOMUXC_ESDHC4_DAT0_IN_SI is 53FA_8000h base + 750h offset = 53FA_8750h

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

IOMUXC_ESDHC4_DAT0_IN_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_dat0_in 0 Selecting Pad: UART3_TXD for Mode: ALT4. 1 Selecting Pad: DISP_D10 for Mode: ALT4.

35.4.470 XC_ESDHC4_IPP_DAT1_IN_SELECT_INPUT (IOMUXC_XC_ESDHC4_1_IN_SI)

Address: IOMUXC_XC_ESDHC4_1_IN_SI is 53FA_8000h base + 754h offset = 53FA_8754h

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

IOMUXC_XC_ESDHC4_1_IN_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_dat1_in 0 Selecting Pad: UART3_RXD for Mode: ALT4. 1 Selecting Pad: DISP_D11 for Mode: ALT4.

35.4.471 ESDHC4_IPP_DAT2_IN_SELECT_INPUT (IOMUXC_ESDHC4_2_IN_SI)

Address: IOMUXC_ESDHC4_2_IN_SI is 53FA_8000h base + 758h offset = 53FA_8758h

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

IOMUXC_ESDHC4_2_IN_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_dat2_in 0 Selecting Pad: UART4_TXD for Mode: ALT4. 1 Selecting Pad: DISP_D12 for Mode: ALT4.

35.4.472 ESDHC4_IPP_DAT3_IN_SELECT_INPUT (IOMUXC_ESDHC4_3_IN_SI)

Address: IOMUXC_ESDHC4_3_IN_SI is 53FA_8000h base + 75Ch offset = 53FA_875Ch

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

IOMUXC_ESDHC4_3_IN_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_dat3_in 0 Selecting Pad: UART4_RXD for Mode: ALT4. 1 Selecting Pad: DISP_D13 for Mode: ALT4.

35.4.473 ESDHC4_IPP_DAT4_IN_SELECT_INPUT (IOMUXC_ESDHC4_4_IN_SI)

Address: IOMUXC_ESDHC4_4_IN_SI is 53FA_8000h base + 760h offset = 53FA_8760h

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

IOMUXC_ESDHC4_4_IN_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_dat4_in 0 Selecting Pad: UART1_CTS for Mode: ALT4. 1 Selecting Pad: UART2_TXD for Mode: ALT5.

35.4.474 ESDHC4_IPP_DAT5_IN_SELECT_INPUT (IOMUXC_ESDHC4_5_IN_SI)

Address: IOMUXC_ESDHC4_5_IN_SI is 53FA_8000h base + 764h offset = 53FA_8764h

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

IOMUXC_ESDHC4_5_IN_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_dat5_in 0 Selecting Pad: UART1_RTS for Mode: ALT4. 1 Selecting Pad: UART2_RXD for Mode: ALT5.

35.4.475 ESDHC4_IPP_DAT6_IN_SELECT_INPUT (IOMUXC_ESDHC4_6_IN_SI)

Address: IOMUXC_ESDHC4_6_IN_SI is 53FA_8000h base + 768h offset = 53FA_8768h

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

IOMUXC_ESDHC4_6_IN_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_dat6_in 0 Selecting Pad: UART2_TXD for Mode: ALT4. 1 Selecting Pad: UART2_CTS for Mode: ALT5.

35.4.476 ESDHC4_IPP_DAT7_IN_SELECT_INPUT (IOMUXC_ESDHC4_7_IN_SI)

Address: IOMUXC_ESDHC4_7_IN_SI is 53FA_8000h base + 76Ch offset = 53FA_876Ch

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

IOMUXC_ESDHC4_7_IN_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc4, In Pin: ipp_dat7_in 0 Selecting Pad: UART2_RXD for Mode: ALT4. 1 Selecting Pad: UART2_RTS for Mode: ALT5.

35.4.477 FEC_FEC_COL_SELECT_INPUT (IOMUXC_FEC_COL_SI)

Address: IOMUXC_FEC_COL_SI is 53FA_8000h base + 770h offset = 53FA_8770h

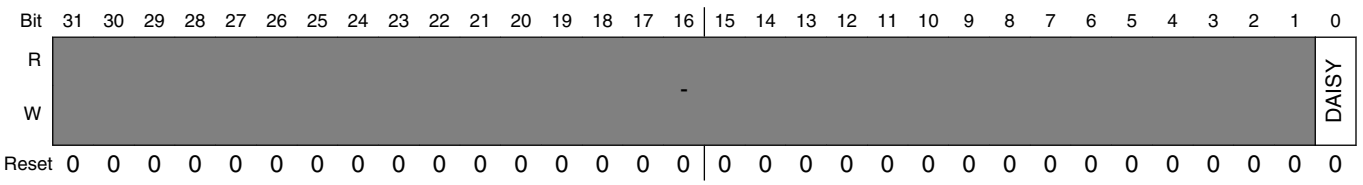
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_FEC_COL_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_col 0 Selecting Pad: SSI_RXFS for Mode: ALT5. 1 Selecting Pad: DISP_D3 for Mode: ALT4.

35.4.478 FEC_FEC_MDI_SELECT_INPUT (IOMUXC_FEC_MDI_SI)

Address: IOMUXC_FEC_MDI_SI is 53FA_8000h base + 774h offset = 53FA_8774h

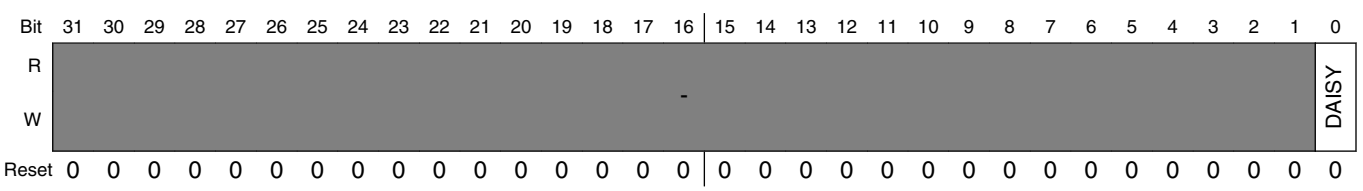


IOMUXC_FEC_MDI_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_mdi 0 Selecting Pad: I2C3_SDA for Mode: ALT2. 1 Selecting Pad: SSI_RXC for Mode: ALT6

35.4.479 FEC_FEC_RDATA_0_SELECT_INPUT (IOMUXC_FEC_RD0_SI)

Address: IOMUXC_FEC_RD0_SI is 53FA_8000h base + 778h offset = 53FA_8778h



IOMUXC_FEC_RD0_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rdata[0] 0 Selecting Pad: DISP_D4 for Mode: ALT2. 1 Selecting Pad: DISP_D12 for Mode: ALT6.

35.4.480 FEC_FEC_RDATA_1_SELECT_INPUT (IOMUXC_FEC_RD1_SI)

Address: IOMUXC_FEC_RD1_SI is 53FA_8000h base + 77Ch offset = 53FA_877Ch

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

IOMUXC_FEC_RD1_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rdata[1] 0 Selecting Pad: DISP_D3 for Mode: ALT2. 1 Selecting Pad: DISP_D11 for Mode: ALT6.

35.4.481 FEC_FEC_RX_CLK_SELECT_INPUT (IOMUXC_FEC_RX_CLK_SI)

Address: IOMUXC_FEC_RX_CLK_SI is 53FA_8000h base + 780h offset = 53FA_8780h

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

IOMUXC_FEC_RX_CLK_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rx_clk 0 Selecting Pad: SSI_RXC for Mode: ALT5. 1 Selecting Pad: DISP_D6 for Mode: ALT4.

35.4.482 FEC_FEC_RX_DV_SELECT_INPUT (IOMUXC_FEC_RX_DV_SI)

Address: IOMUXC_FEC_RX_DV_SI is 53FA_8000h base + 784h offset = 53FA_8784h

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

IOMUXC_FEC_RX_DV_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rx_dv 0 Selecting Pad: DISP_D2 for Mode: ALT2. 1 Selecting Pad: DISP_D10 for Mode: ALT6.

35.4.483 FEC_FEC_RX_ER_SELECT_INPUT (IOMUXC_FEC_RX_ER_SI)

Address: IOMUXC_FEC_RX_ER_SI is 53FA_8000h base + 788h offset = 53FA_8788h

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

IOMUXC_FEC_RX_ER_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rx_er 0 Selecting Pad: DISP_D1 for Mode: ALT2. 1 Selecting Pad: DISP_D9 for Mode: ALT6.

35.4.484 FEC_FEC_TX_CLK_SELECT_INPUT (IOMUXC_FEC_TX_CLK_SI)

Address: IOMUXC_FEC_TX_CLK_SI is 53FA_8000h base + 78Ch offset = 53FA_878Ch

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

IOMUXC_FEC_TX_CLK_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_tx_clk 0 Selecting Pad: DISP_D0 for Mode: ALT2. 1 Selecting Pad: DISP_D8 for Mode: ALT6.

35.4.485 KPP_IPP_IND_COL_4_SELECT_INPUT (IOMUXC_KPP_FC4_SI)

Address: IOMUXC_KPP_FC4_SI is 53FA_8000h base + 790h offset = 53FA_8790h

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

IOMUXC_KPP_FC4_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_col[4] 00 Selecting Pad: SD2_D0 for Mode: ALT3. 01 Selecting Pad: DISP_D8 for Mode: ALT5. 10 Selecting Pad: EIM_DA0 for Mode: ALT3.

35.4.486 KPP_IPP_IND_COL_5_SELECT_INPUT (IOMUXC_KPP_FC5_SI)

Address: IOMUXC_KPP_FC5_SI is 53FA_8000h base + 794h offset = 53FA_8794h

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

IOMUXC_KPP_FC5_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_col[5] 00 Selecting Pad: SD2_D2 for Mode: ALT3. 01 Selecting Pad: DISP_D10 for Mode: ALT5. 10 Selecting Pad: EIM_DA2 for Mode: ALT3.

35.4.487 KPP_IPP_IND_COL_6_SELECT_INPUT (IOMUXC_KPP_FC6_SI)

Address: IOMUXC_KPP_FC6_SI is 53FA_8000h base + 798h offset = 53FA_8798h

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

IOMUXC_KPP_FC6_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_col[6] 00 Selecting Pad: SD2_D4 for Mode: ALT3. 01 Selecting Pad: DISP_D12 for Mode: ALT5. 10 Selecting Pad: EIM_DA4 for Mode: ALT3.

35.4.488 KPP_IPP_IND_COL_7_SELECT_INPUT (IOMUXC_KPP_FC7_SI)

Address: IOMUXC_KPP_FC7_SI is 53FA_8000h base + 79Ch offset = 53FA_879Ch

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

IOMUXC_KPP_FC7_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_col[7] 00 Selecting Pad: SD2_D6 for Mode: ALT3. 01 Selecting Pad: DISP_D14 for Mode: ALT5. 10 Selecting Pad: EIM_DA6 for Mode: ALT3.

35.4.489 KPP_IPP_IND_ROW_4_SELECT_INPUT (IOMUXC_KPP_FR4_SI)

Address: IOMUXC_KPP_FR4_SI is 53FA_8000h base + 7A0h offset = 53FA_87A0h

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

IOMUXC_KPP_FR4_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_row[4] 00 Selecting Pad: SD2_D1 for Mode: ALT3. 01 Selecting Pad: DISP_D9 for Mode: ALT5. 10 Selecting Pad: EIM_DA1 for Mode: ALT3.

35.4.490 KPP_IPP_IND_ROW_5_SELECT_INPUT (IOMUXC_KPP_FR5_SI)

Address: IOMUXC_KPP_FR5_SI is 53FA_8000h base + 7A4h offset = 53FA_87A4h

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

IOMUXC_KPP_FR5_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_row[5] 00 Selecting Pad: SD2_D3 for Mode: ALT3. 01 Selecting Pad: DISP_D11 for Mode: ALT5. 10 Selecting Pad: EIM_DA3 for Mode: ALT3.

35.4.491 KPP_IPP_IND_ROW_6_SELECT_INPUT (IOMUXC_KPP_FR6_SI)

Address: IOMUXC_KPP_FR6_SI is 53FA_8000h base + 7A8h offset = 53FA_87A8h

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

IOMUXC_KPP_FR6_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_row[6] 00 Selecting Pad: SD2_D5 for Mode: ALT3. 01 Selecting Pad: DISP_D13 for Mode: ALT5. 10 Selecting Pad: EIM_DA5 for Mode: ALT3.

35.4.492 KPP_IPP_IND_ROW_7_SELECT_INPUT (IOMUXC_KPP_FR7_SI)

Address: IOMUXC_KPP_FR7_SI is 53FA_8000h base + 7ACh offset = 53FA_87ACh

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

IOMUXC_KPP_FR7_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_row[7] 00 Selecting Pad: SD2_D7 for Mode: ALT3. 01 Selecting Pad: DISP_D15 for Mode: ALT5. 10 Selecting Pad: EIM_DA7 for Mode: ALT3.

35.4.493 RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SELECT_INPUT (IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SI)

Address: IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SI is 53FA_8000h base + 7B0h offset = 53FA_87B0h

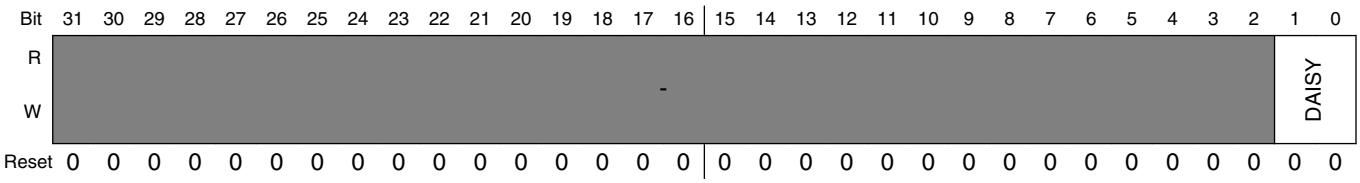
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_DQS_IN_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: rawnand, In Pin: u_gpmi_input_gpmi_dqs_in 00 Selecting Pad: KEY_ROW3 for Mode: ALT2. 01 Selecting Pad: DISP_D15 for Mode: ALT2. 10 Selecting Pad: EIM_DA15 for Mode: ALT2.

35.4.494 RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SELECT_INPUT (IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SI)

Address: IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SI is 53FA_8000h base + 7B4h offset = 53FA_87B4h

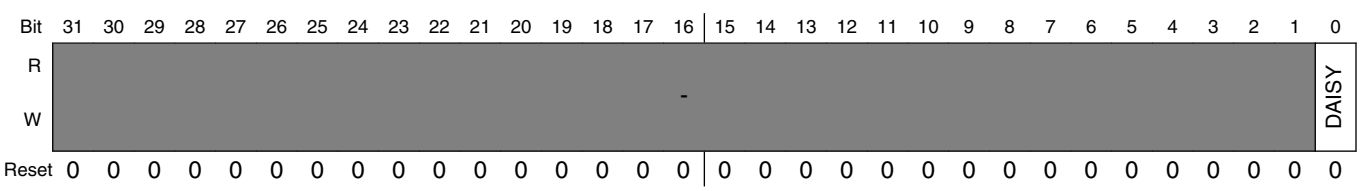


IOMUXC_RAWNAND_U_GPMI_INPUT_GPMI_RDY0_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: rawnand, In Pin: u_gpmi_input_gpmi_rdy0 00 Selecting Pad: KEY_COL3 for Mode: ALT2. 01 Selecting Pad: DISP_D14 for Mode: ALT2. 10 Selecting Pad: EIM_DA14 for Mode: ALT2.

35.4.495 SDMA_EVENTS_14_SELECT_INPUT (IOMUXC_SDMA_EVENTS_14_SI)

Address: IOMUXC_SDMA_EVENTS_14_SI is 53FA_8000h base + 7B8h offset = 53FA_87B8h



IOMUXC_SDMA_EVENTS_14_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sdma, In Pin: events[14] 0 Selecting Pad: KEY_COL3 for Mode: ALT6. 1 Selecting Pad: EPDC_PWRCTRL2 for Mode: ALT6.

35.4.496 SDMA_EVENTS_15_SELECT_INPUT (IOMUXC_SDMA_EVENTS_15_SI)

Address: IOMUXC_SDMA_EVENTS_15_SI is 53FA_8000h base + 7BCh offset = 53FA_87BCh

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

IOMUXC_SDMA_EVENTS_15_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sdma, In Pin: events[15] 0 Selecting Pad: KEY_ROW3 for Mode: ALT6. 1 Selecting Pad: EPDC_PWRCTRL3 for Mode: ALT6.

35.4.497 UART1_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U1U_RTS_B_SI)

Address: IOMUXC_U1U_RTS_B_SI is 53FA_8000h base + 7C0h offset = 53FA_87C0h

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

IOMUXC_U1U_RTS_B_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart1, In Pin: ipp_uart_rts_b 0 Selecting Pad: UART1_CTS for Mode: ALT0. 1 Selecting Pad: UART1_RTS for Mode: ALT0.

35.4.498 UART1_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U1U_RXD_MUX_SI)

Address: IOMUXC_U1U_RXD_MUX_SI is 53FA_8000h base + 7C4h offset = 53FA_87C4h

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

IOMUXC_U1U_RXD_MUX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart1, In Pin: ipp_uart_rxd_mux 0 Selecting Pad: UART1_TXD for Mode: ALT0. 1 Selecting Pad: UART1_RXD for Mode: ALT0.

35.4.499 UART2_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U2U_RTS_B_SI)

Address: IOMUXC_U2U_RTS_B_SI is 53FA_8000h base + 7C8h offset = 53FA_87C8h

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

IOMUXC_U2U_RTS_B_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart2, In Pin: ipp_uart_rts_b 00 Selecting Pad: I2C2_SCL for Mode: ALT2. 01 Selecting Pad: I2C2_SDA for Mode: ALT2. 10 Selecting Pad: UART2_CTS for Mode: ALT0. 11 Selecting Pad: UART2_RTS for Mode: ALT0.

35.4.500 UART2_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U2U_RXD_MUX_SI)

Address: IOMUXC_U2U_RXD_MUX_SI is 53FA_8000h base + 7CCh offset = 53FA_87CCh

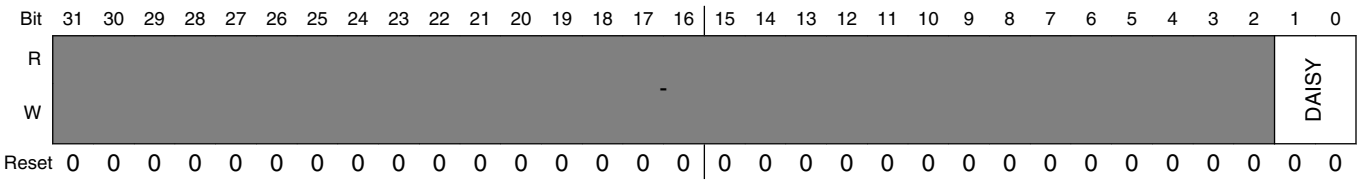
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																														DAISY	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_U2U_RXD_MUX_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart2, In Pin: ipp_uart_rxd_mux 00 Selecting Pad: I2C1_SCL for Mode: ALT2. 01 Selecting Pad: I2C1_SDA for Mode: ALT2. 10 Selecting Pad: UART2_TXD for Mode: ALT0. 11 Selecting Pad: UART2_RXD for Mode: ALT0.

35.4.501 UART3_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U3U_RTS_B_SI)

Address: IOMUXC_U3U_RTS_B_SI is 53FA_8000h base + 7D0h offset = 53FA_87D0h

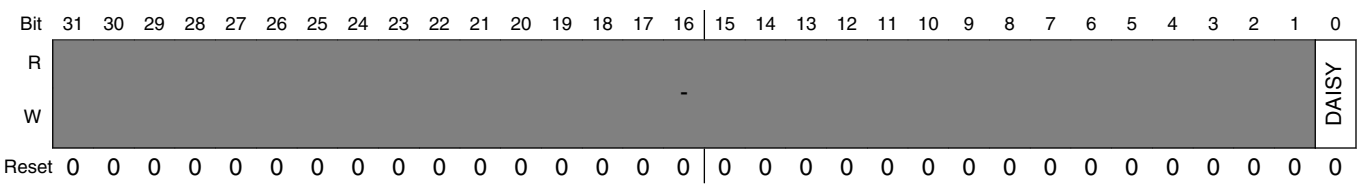


IOMUXC_U3U_RTS_B_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart3, In Pin: ipp_uart_rts_b 00 Selecting Pad: UART4_TXD for Mode: ALT2. 01 Selecting Pad: UART4_RXD for Mode: ALT2. 10 Selecting Pad: ECSPI1_SCLK for Mode: ALT4. 11 Selecting Pad: ECSPI1_MOSI for Mode: ALT4.

35.4.502 UART3_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U3U_RXD_MUX_SI)

Address: IOMUXC_U3U_RXD_MUX_SI is 53FA_8000h base + 7D4h offset = 53FA_87D4h

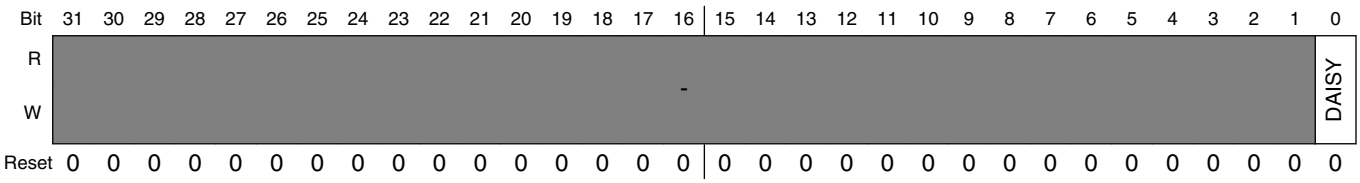


IOMUXC_U3U_RXD_MUX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart3, In Pin: ipp_uart_rxd_mux 0 Selecting Pad: UART3_TXD for Mode: ALT0. 1 Selecting Pad: UART3_RXD for Mode: ALT0.

35.4.503 UART4_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U4U_RTS_B_SI)

Address: IOMUXC_U4U_RTS_B_SI is 53FA_8000h base + 7D8h offset = 53FA_87D8h

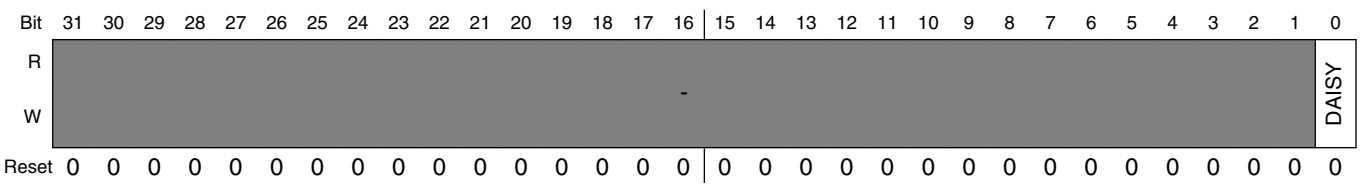


IOMUXC_U4U_RTS_B_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart4, In Pin: ipp_uart_rts_b 0 Selecting Pad: ECSPI1_MISO for Mode: ALT4. 1 Selecting Pad: ECSPI1_SS0 for Mode: ALT4.

35.4.504 UART4_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U4U_RXD_MUX_SI)

Address: IOMUXC_U4U_RXD_MUX_SI is 53FA_8000h base + 7DCh offset = 53FA_87DCh



IOMUXC_U4U_RXD_MUX_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart4, In Pin: ipp_uart_rxd_mux 0 Selecting Pad: UART4_TXD for Mode: ALT0. 1 Selecting Pad: UART4_RXD for Mode: ALT0.

35.4.505 UART5_IPP_UART_RTS_B_SELECT_INPUT (IOMUXC_U5U_RTS_B_SI)

Address: IOMUXC_U5U_RTS_B_SI is 53FA_8000h base + 7E0h offset = 53FA_87E0h

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

IOMUXC_U5U_RTS_B_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart5, In Pin: ipp_uart_rts_b 0 Selecting Pad: ECSPI2_SCLK for Mode: ALT4. 1 Selecting Pad: ECSPI2_MOSI for Mode: ALT4.

35.4.506 UART5_IPP_UART_RXD_MUX_SELECT_INPUT (IOMUXC_U5U_RXD_MUX_SI)

Address: IOMUXC_U5U_RXD_MUX_SI is 53FA_8000h base + 7E4h offset = 53FA_87E4h

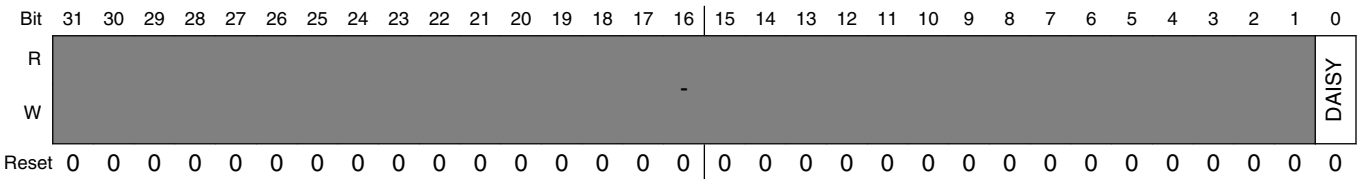
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																															DAISY
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_U5U_RXD_MUX_SI field descriptions

Field	Description
31–3 -	Reserved
2–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart5, In Pin: ipp_uart_rxd_mux 000 Selecting Pad: SSI_RXFS for Mode: ALT2. 001 Selecting Pad: SSI_RXC for Mode: ALT2. 010 Selecting Pad: UART1_CTS for Mode: ALT2. 011 Selecting Pad: UART1_RTS for Mode: ALT2. 100 Selecting Pad: ECSPI2_MISO for Mode: ALT4. 101 Selecting Pad: ECSPI2_SS0 for Mode: ALT4.

35.4.507 USBOH1_IPP_IND_OTG_OC_SELECT_INPUT (IOMUXC_USBOH1_OTG_OC_SI)

Address: IOMUXC_USBOH1_OTG_OC_SI is 53FA_8000h base + 7E8h offset = 53FA_87E8h

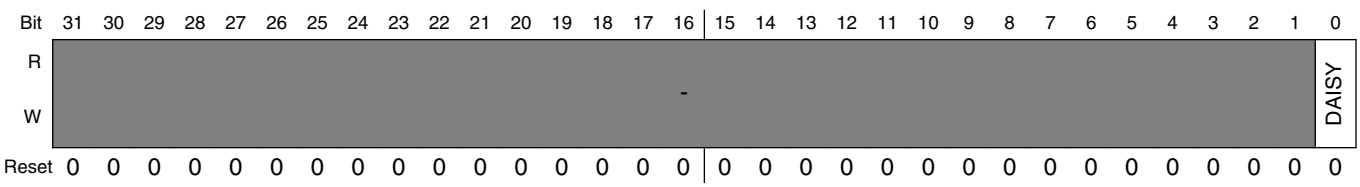


IOMUXC_USBOH1_OTG_OC_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usboh1, In Pin: ipp_ind_otg_oc 0 Selecting Pad: I2C3_SCL for Mode: ALT7. 1 Selecting Pad: PWM1 for Mode: ALT2.

35.4.508 WEIMV2_IPP_IND_READ_DATA_0_SELECT_INPUT (IOMUXC_WEIMV2_RD0_SI)

Address: IOMUXC_WEIMV2_RD0_SI is 53FA_8000h base + 7ECh offset = 53FA_87ECh



IOMUXC_WEIMV2_RD0_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[0] 0 Selecting Pad: SD2_D4 for Mode: ALT4. 1 Selecting Pad: EPDC_D0 for Mode: ALT2.

35.4.509 WEIMV2_IPP_IND_READ_DATA_1_SELECT_INPUT (IOMUXC_WEIMV2_RD1_SI)

Address: IOMUXC_WEIMV2_RD1_SI is 53FA_8000h base + 7F0h offset = 53FA_87F0h

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

IOMUXC_WEIMV2_RD1_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[1] 0 Selecting Pad: SD2_D5 for Mode: ALT4. 1 Selecting Pad: EPDC_D1 for Mode: ALT2.

35.4.510 WEIMV2_IPP_IND_READ_DATA_2_SELECT_INPUT (IOMUXC_WEIMV2_RD2_SI)

Address: IOMUXC_WEIMV2_RD2_SI is 53FA_8000h base + 7F4h offset = 53FA_87F4h

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

IOMUXC_WEIMV2_RD2_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[2] 0 Selecting Pad: SD2_D6 for Mode: ALT4. 1 Selecting Pad: EPDC_D2 for Mode: ALT2.

35.4.511 WEIMV2_IPP_IND_READ_DATA_3_SELECT_INPUT (IOMUXC_WEIMV2_RD3_SI)

Address: IOMUXC_WEIMV2_RD3_SI is 53FA_8000h base + 7F8h offset = 53FA_87F8h

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

IOMUXC_WEIMV2_RD3_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[3] 0 Selecting Pad: SD2_D7 for Mode: ALT4. 1 Selecting Pad: EPDC_D3 for Mode: ALT2.

35.4.512 WEIMV2_IPP_IND_READ_DATA_4_SELECT_INPUT (IOMUXC_WEIMV2_RD4_SI)

Address: IOMUXC_WEIMV2_RD4_SI is 53FA_8000h base + 7FCh offset = 53FA_87FCh

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

IOMUXC_WEIMV2_RD4_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[4] 0 Selecting Pad: SD2_WP for Mode: ALT4. 1 Selecting Pad: EPDC_D4 for Mode: ALT2.

35.4.513 WEIMV2_IPP_IND_READ_DATA_5_SELECT_INPUT (IOMUXC_WEIMV2_RD5_SI)

Address: IOMUXC_WEIMV2_RD5_SI is 53FA_8000h base + 800h offset = 53FA_8800h

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

IOMUXC_WEIMV2_RD5_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[5] 0 Selecting Pad: SD2_CD for Mode: ALT4. 1 Selecting Pad: EPDC_D5 for Mode: ALT2.

35.4.514 WEIMV2_IPP_IND_READ_DATA_6_SELECT_INPUT (IOMUXC_WEIMV2_RD6_SI)

Address: IOMUXC_WEIMV2_RD6_SI is 53FA_8000h base + 804h offset = 53FA_8804h

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

IOMUXC_WEIMV2_RD6_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[6] 0 Selecting Pad: SSI_RXFS for Mode: ALT3. 1 Selecting Pad: EPDC_D6 for Mode: ALT2.

35.4.515 WEIMV2_IPP_IND_READ_DATA_7_SELECT_INPUT (IOMUXC_WEIMV2_RD7_SI)

Address: IOMUXC_WEIMV2_RD7_SI is 53FA_8000h base + 808h offset = 53FA_8808h

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

IOMUXC_WEIMV2_RD7_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[7] 0 Selecting Pad: SSI_RXC for Mode: ALT3. 1 Selecting Pad: EPDC_D7 for Mode: ALT2.

35.4.516 WEIMV2_IPP_IND_READ_DATA_8_SELECT_INPUT (IOMUXC_WEIMV2_RD8_SI)

Address: IOMUXC_WEIMV2_RD8_SI is 53FA_8000h base + 80Ch offset = 53FA_880Ch

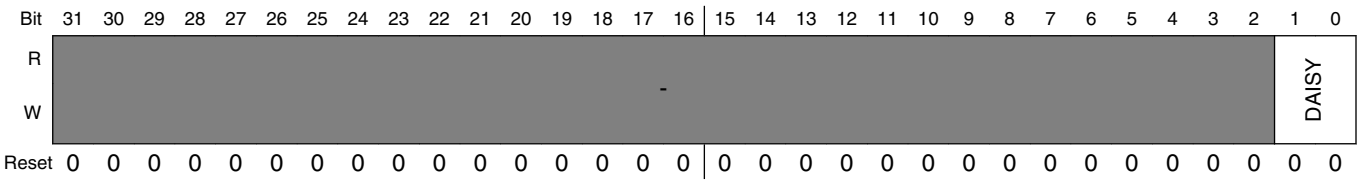
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																															DAISY
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_WEIMV2_RD8_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[8] 00 Selecting Pad: ECSPI1_SCLK for Mode: ALT7. 01 Selecting Pad: ECSPI2_SCLK for Mode: ALT7. 10 Selecting Pad: EPDC_D8 for Mode: ALT2.

35.4.517 WEIMV2_IPP_IND_READ_DATA_9_SELECT_INPUT (IOMUXC_WEIMV2_RD9_SI)

Address: IOMUXC_WEIMV2_RD9_SI is 53FA_8000h base + 810h offset = 53FA_8810h

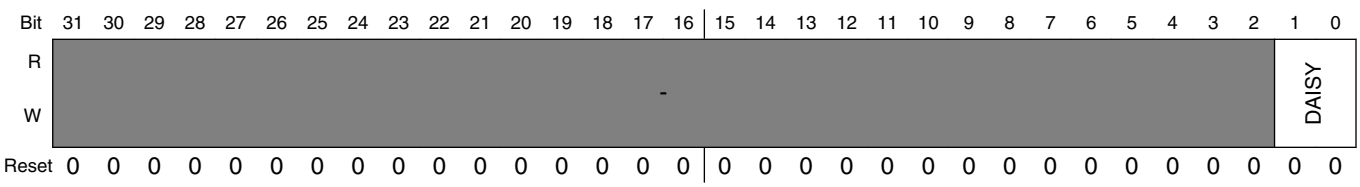


IOMUXC_WEIMV2_RD9_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[9] 00 Selecting Pad: ECSPI1_MOSI for Mode: ALT7. 01 Selecting Pad: ECSPI2_MOSI for Mode: ALT7. 10 Selecting Pad: EPDC_D9 for Mode: ALT2.

35.4.518 WEIMV2_IPP_IND_READ_DATA_10_SELECT_INPUT (IOMUXC_WEIMV2_RD10_SI)

Address: IOMUXC_WEIMV2_RD10_SI is 53FA_8000h base + 814h offset = 53FA_8814h

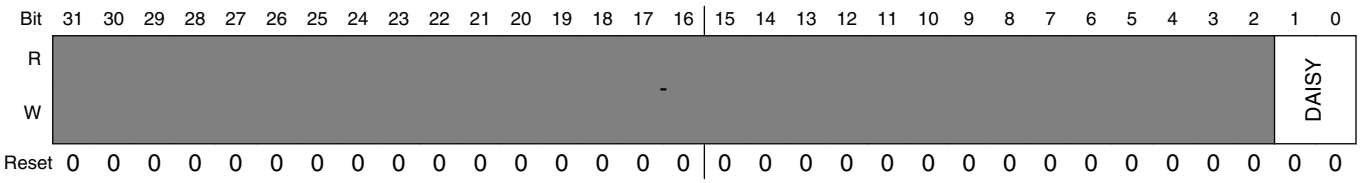


IOMUXC_WEIMV2_RD10_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[10] 00 Selecting Pad: ECSPI1_MISO for Mode: ALT7. 01 Selecting Pad: ECSPI2_MISO for Mode: ALT7. 10 Selecting Pad: EPDC_D10 for Mode: ALT2.

35.4.519 WEIMV2_IPP_IND_READ_DATA_11_SELECT_INPUT (IOMUXC_WEIMV2_RD11_SI)

Address: IOMUXC_WEIMV2_RD11_SI is 53FA_8000h base + 818h offset = 53FA_8818h

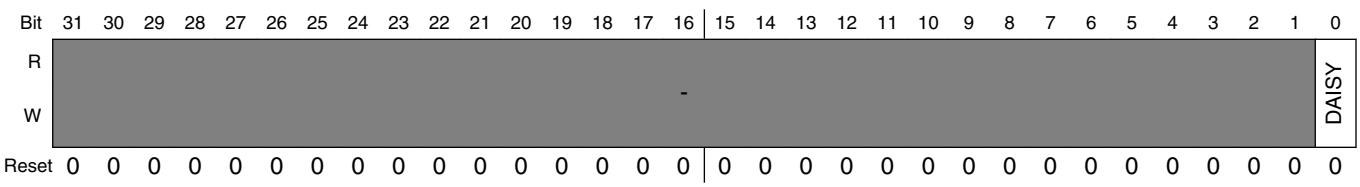


IOMUXC_WEIMV2_RD11_SI field descriptions

Field	Description
31–2 -	Reserved
1–0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[11] 00 Selecting Pad: ECSPI1_SS0 for Mode: ALT7. 01 Selecting Pad: ECSPI2_SS0 for Mode: ALT7. 10 Selecting Pad: EPDC_D11 for Mode: ALT2.

35.4.520 WEIMV2_IPP_IND_READ_DATA_12_SELECT_INPUT (IOMUXC_WEIMV2_RD12_SI)

Address: IOMUXC_WEIMV2_RD12_SI is 53FA_8000h base + 81Ch offset = 53FA_881Ch

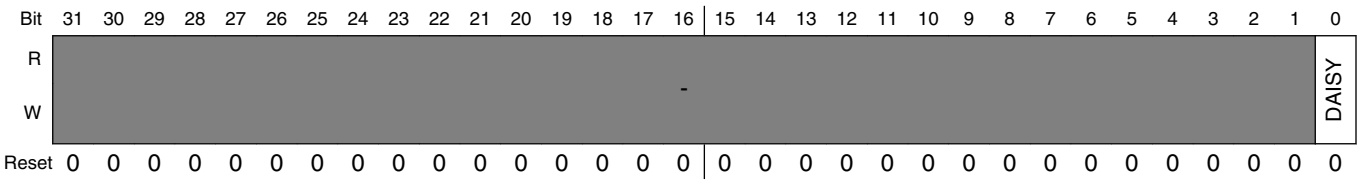


IOMUXC_WEIMV2_RD12_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[12] 0 Selecting Pad: UART3_TXD for Mode: ALT6. 1 Selecting Pad: EPDC_D12 for Mode: ALT2.

35.4.521 WEIMV2_IPP_IND_READ_DATA_13_SELECT_INPUT (IOMUXC_WEIMV2_RD13_SI)

Address: IOMUXC_WEIMV2_RD13_SI is 53FA_8000h base + 820h offset = 53FA_8820h

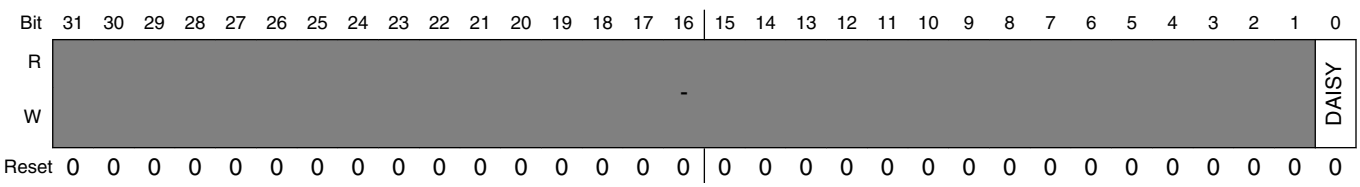


IOMUXC_WEIMV2_RD13_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[13] 0 Selecting Pad: UART3_RXD for Mode: ALT6. 1 Selecting Pad: EPDC_D13 for Mode: ALT2.

35.4.522 WEIMV2_IPP_IND_READ_DATA_14_SELECT_INPUT (IOMUXC_WEIMV2_RD14_SI)

Address: IOMUXC_WEIMV2_RD14_SI is 53FA_8000h base + 824h offset = 53FA_8824h



IOMUXC_WEIMV2_RD14_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[14] 0 Selecting Pad: UART4_TXD for Mode: ALT6. 1 Selecting Pad: EPDC_D14 for Mode: ALT2.

35.4.523 WEIMV2_IPP_IND_READ_DATA_15_SELECT_INPUT (IOMUXC_WEIMV2_RD15_SI)

Address: IOMUXC_WEIMV2_RD15_SI is 53FA_8000h base + 828h offset = 53FA_8828h

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

IOMUXC_WEIMV2_RD15_SI field descriptions

Field	Description
31–1 -	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: weimv2, In Pin: ipp_ind_read_data[15] 0 Selecting Pad: UART4_RXD for Mode: ALT6. 1 Selecting Pad: EPDC_D15 for Mode: ALT2.

Chapter 36
Keypad Port (KPP)

36.1 Overview

The Keypad Port (KPP) is a 16-bit peripheral that can be used as a keypad matrix interface or as general purpose input/output (I/O). The figure below shows the KPP block diagram. The KPP provides interface for the keypad matrix with 2-point contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously on the keypad.

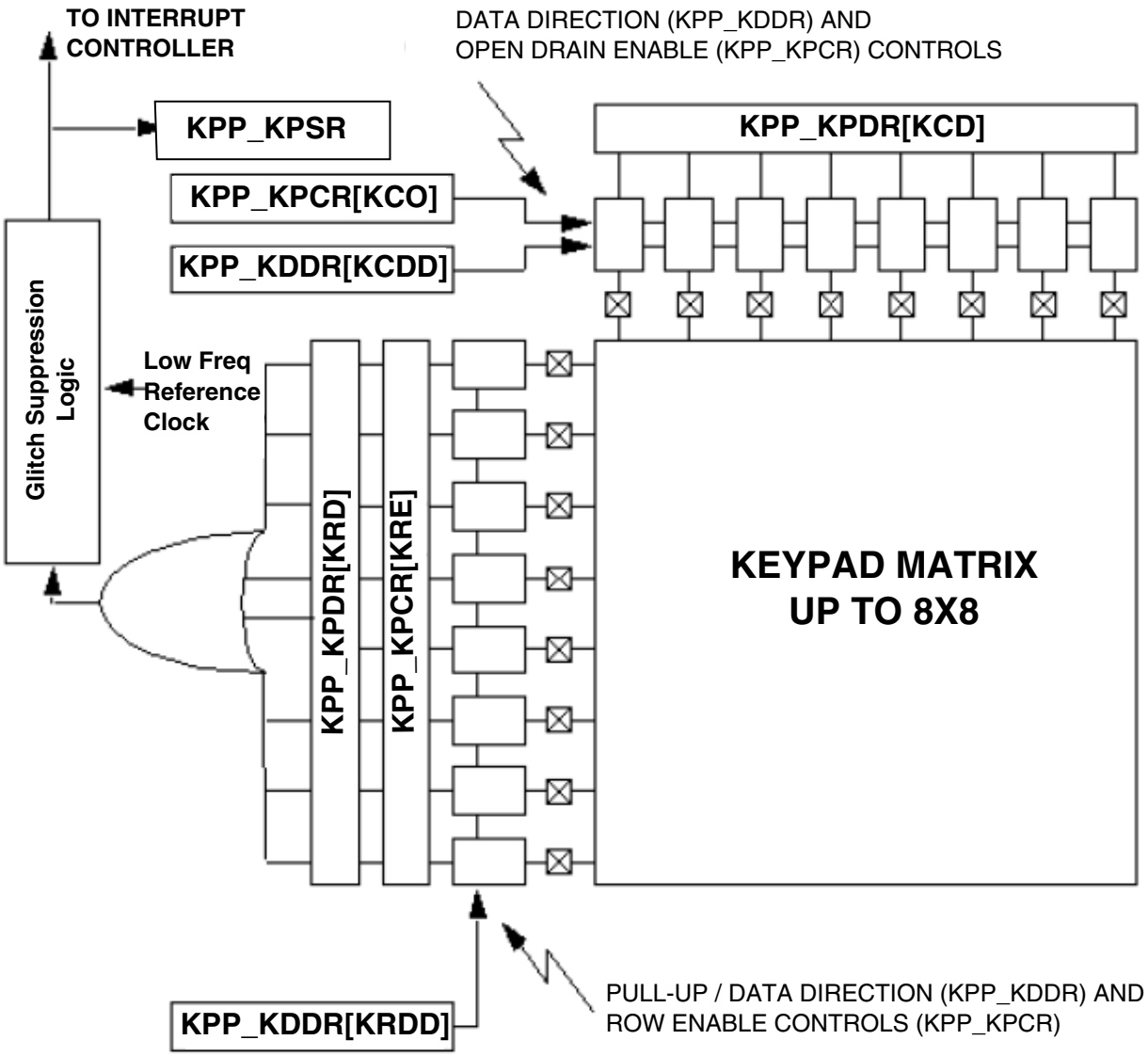


Figure 36-1. KPP Peripheral Block Diagram

36.1.1 Features

The KPP includes these distinctive features:

- Supports up to an 8 x 8 external key pad matrix
- Port pins can be used as general purpose I/O
- Open drain design
- Glitch suppression circuit design
- Multiple-key detection
- Long key-press detection

- Standby key-press detection
- Synchronizer chain clear
- Supports a 2-point and 3-point contact key matrix

36.1.2 Modes and Operations

This block supports the following modes:

- Run Mode-This is the normal functional mode in which the KPP can detect any key press event.
- Low Power Mode-The keypad can detect any key press even in low power modes (when there is no MCU clock).

36.2 External Signals

36.2.1 External Signals Overview

There are 16 pins dedicated to the KPP. Keypads of any configuration up to eight rows and eight columns are supported through the software configuration of the peripheral pins. Any pins not used for the keypad are available as general purpose I/O. The registers are configured such that the pins can be treated as an I/O port up to 16 bits wide.

See the table below for the list of external signals.

Table 36-1. Block Signals

Name	Function	Reset State	Pull up	
KPCOL[7:0]	Column input or output pin, from chip	0	Active[1]	
KPROW[7:0]	Row input or output pin, from chip	1	Active[2]	

1. The corresponding pads are required to be pull-up enabled.
2. The corresponding pads are required to be pull-up enabled.

36.2.1.1 Input Pins

Any of the 16 pins associated with the KPP can be configured as inputs by writing a "0" to the appropriate bits in the KPP_KDDR. Additionally, the least significant 8 bits (ROW inputs) corresponding to KPP_KDDR[KRDD] have internal pull-ups, which are enabled when the pin is used as an input.

36.2.1.2 Output Pins

Any of the 16 pins associated with the KPP can be configured as outputs by writing the appropriate bits in the KPP_KDDR to a "1". Additionally, the 8 most significant bits (15-8) can be designated as open drain outputs by writing a "1" to the appropriate bits in the KPP_KPCR. The lower 8 bits (7-0) are always in "totem pole" style, driven when configured as outputs.

See the table below.

Table 36-2. Keypad Port Column Modes

KPP_KDDR (15:8)	KPP_KPCR (15:8)	Pin Function
0	x	Input
1	0	Totem-Pole Output
1	1	Open-Drain Output

NOTE

Totem pole capability should be provided for column pins. Totem pole configuration helps for a faster discharge of keypad capacitance when all columns need to be quickly brought to a "1" during the scan routine. With this configuration, delay between the scanning of two subsequent columns is reduced.

36.2.1.3 Generation of Transfer Error Signal on Peripheral Bus

If there is an access to an address which is not implemented, then the KPP asserts a transfer error signal on Peripheral Bus.

36.3 Functional Description

The Keypad Port (KPP) is designed to simplify the software task of scanning a keypad matrix. With appropriate software support and matrix organization, the KPP is capable of detecting, debouncing, and decoding one or more keys pressed simultaneously on the keypad.

Logic in the KPP is capable of detecting a key press even while the processor is in one of the low power standby modes provided that a low frequency reference clock is on. The KPP may generate an ARM platform interrupt any time a key press or key release is detected. This interrupt is capable of forcing the processor out of a low power mode.

36.3.1 Keypad Matrix Construction

The KPP is designed to interface to a keypad matrix, which shorts the intersecting row and column lines together whenever a key is depressed. The interface is not optimized for any other switch configuration.

36.3.2 Keypad Port Configuration

The software must initialize the KPP for the size of the keypad matrix. Pins connected to the keypad columns should be configured as open-drain outputs. Pins connected to the keypad rows should be configured as inputs. On-chip, pull-up resistors should be implemented for active keypad rows.

In addition to enabled row inputs in the Keypad Control register, corresponding interrupt (depress or/and release) must also be enabled to generate an interrupt.

Discrete switches that are not part of the matrix may be connected to any unused row inputs. The second terminal of the discrete switch is connected to ground. The hardware detects closures of these switches without the need for software polling.

36.3.3 Keypad Matrix Scanning

Keypad scanning is performed by a software loop that walks a zero across each of the keypad columns, reading the value on the rows at each step. The process is repeated several times in succession, with the results of each pass optionally compared to those

from the previous pass. When several (3 or 4) consecutive scans yield the same key closures, a valid key press has been detected. Software then can decode exactly which switch was depressed and pass the value up to the next higher software layer.

The basic debouncing period, which must be defined in the software routine, may be controlled with an internal timer. The basic period is the period between the scan of two consecutive columns, so the debouncing time between two consecutive scans of the whole matrix shall be the number of columns multiplied by the basic period.

36.3.4 Keypad Standby

There is no need for the ARM platform to continually scan the keypad. Between key presses, the keypad can be left in a state that requires no software intervention until the next key press is detected. To place the keypad in a standby state, software should write all column outputs low. Row inputs are left enabled. At this point, the ARM platform can attend to other tasks or revert to a low power standby mode. The KPP will interrupt the ARM platform if any key is pressed.

Upon receiving a keypad interrupt, the ARM platform should set all the column strobes high, and begin a normal keypad scanning routine to determine which key was pressed. It is important that open-drain drivers be used when scanning to prevent a possible DC path between power and ground through two or more switches.

36.3.5 Glitch Suppression on Keypad Inputs

A glitch suppression circuit qualifies the keypad inputs to prevent noise from inadvertently interrupting the ARM platform. The circuit is a 4-state synchronizer clocked from a low frequency reference clock source.

This clock must continue to run in any low power mode where the keypad is a wake-up source, as the ARM platform interrupt is generated from the synchronized input. An interrupt is not generated until all four synchronizer stages have latched a valid key assertion. This guarantees the filtering out of any noise less than three clock periods in duration of a low frequency reference clock. Noise filtering of the duration between three to four clock periods cannot be guaranteed. The interrupt output is latched in an S-R latch and remains asserted until cleared by the software. The Set input of the latch is rising-edge clocked. See the figure below.

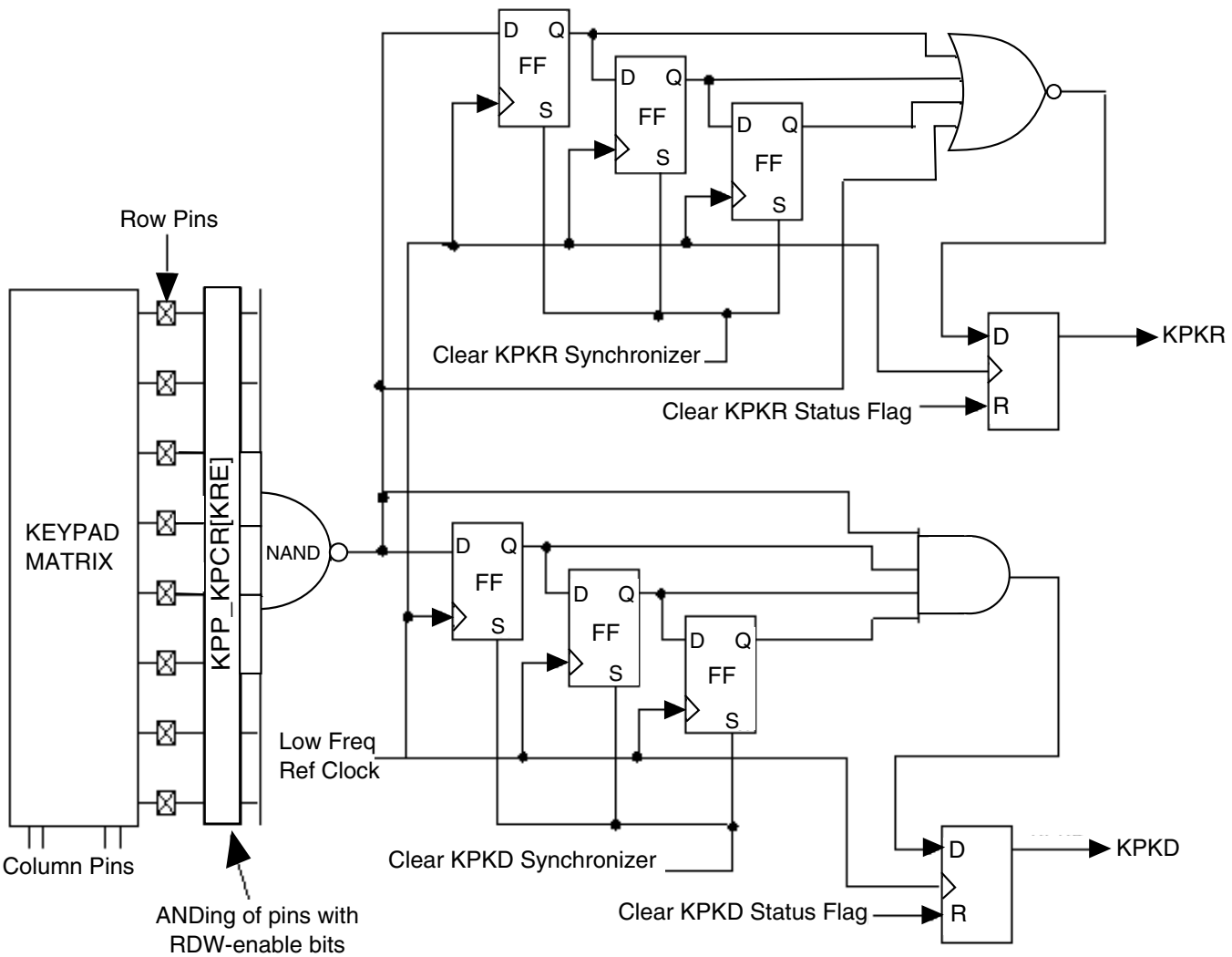


Figure 36-2. Keypad Synchronizer Functional Diagram

36.3.6 Multiple Key Closures

Using the key press and Key release interrupts, the software can detect multiple keys or achieve n key rollover. The key scanning routine can be programmed accordingly.

See the following figures for illustrations of the interfacing of a 2-contact keypad matrix with the KPP controller. With proper enabling of row lines and the performing scan-routine, multiple key presses can be detected. When keys present on the same row are pressed, corresponding row lines (multiple lines) become low when the column is driven low during a scan-routine. By reading the data-register, pressed keys can be detected. Similarly, when keys present on same row line are pressed, the corresponding row line (only one line) becomes low when logic "0" is driven on the column line during a scan-routine.

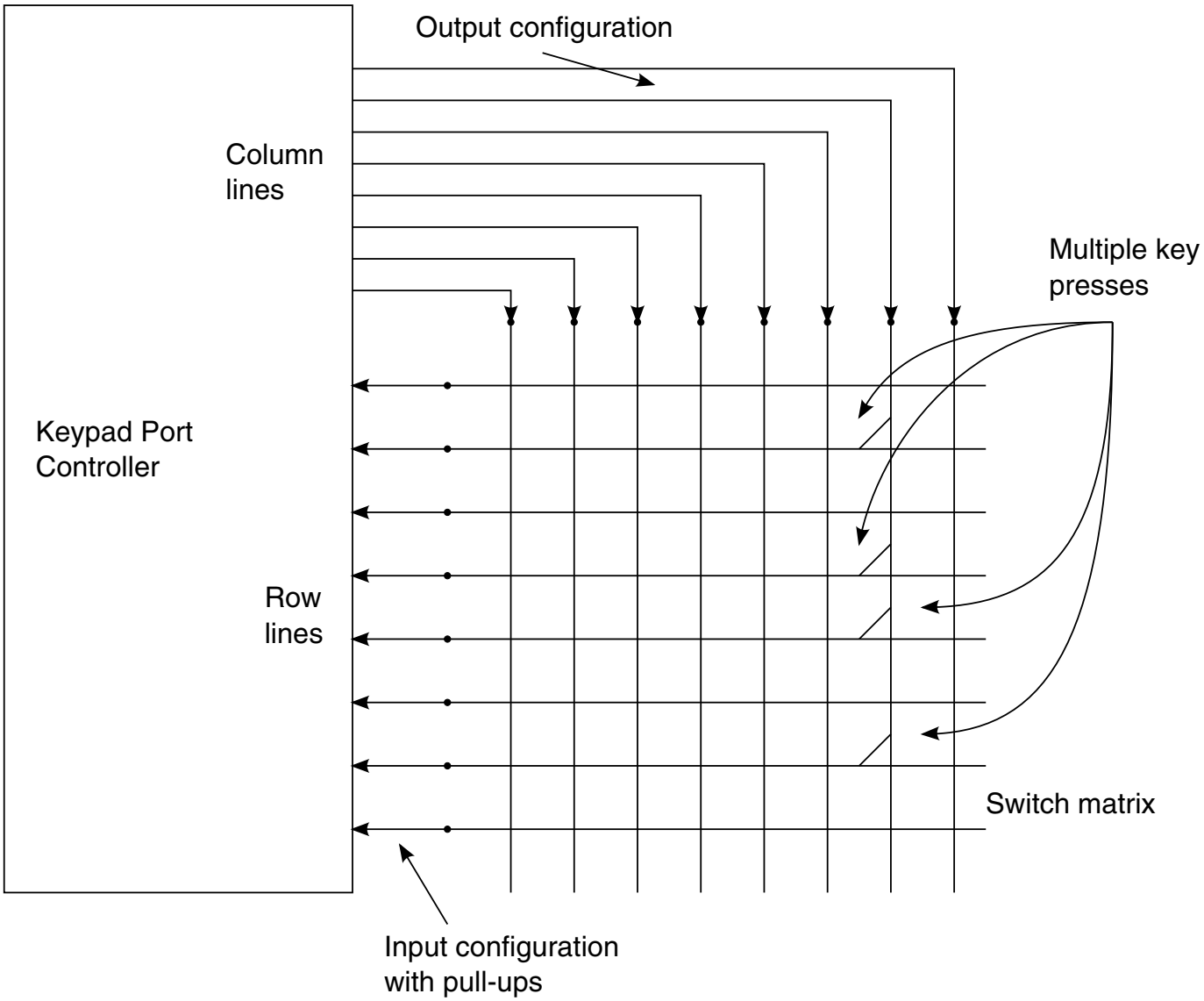


Figure 36-3. Multiple Key Presses on Same Column Line (Simplified View)

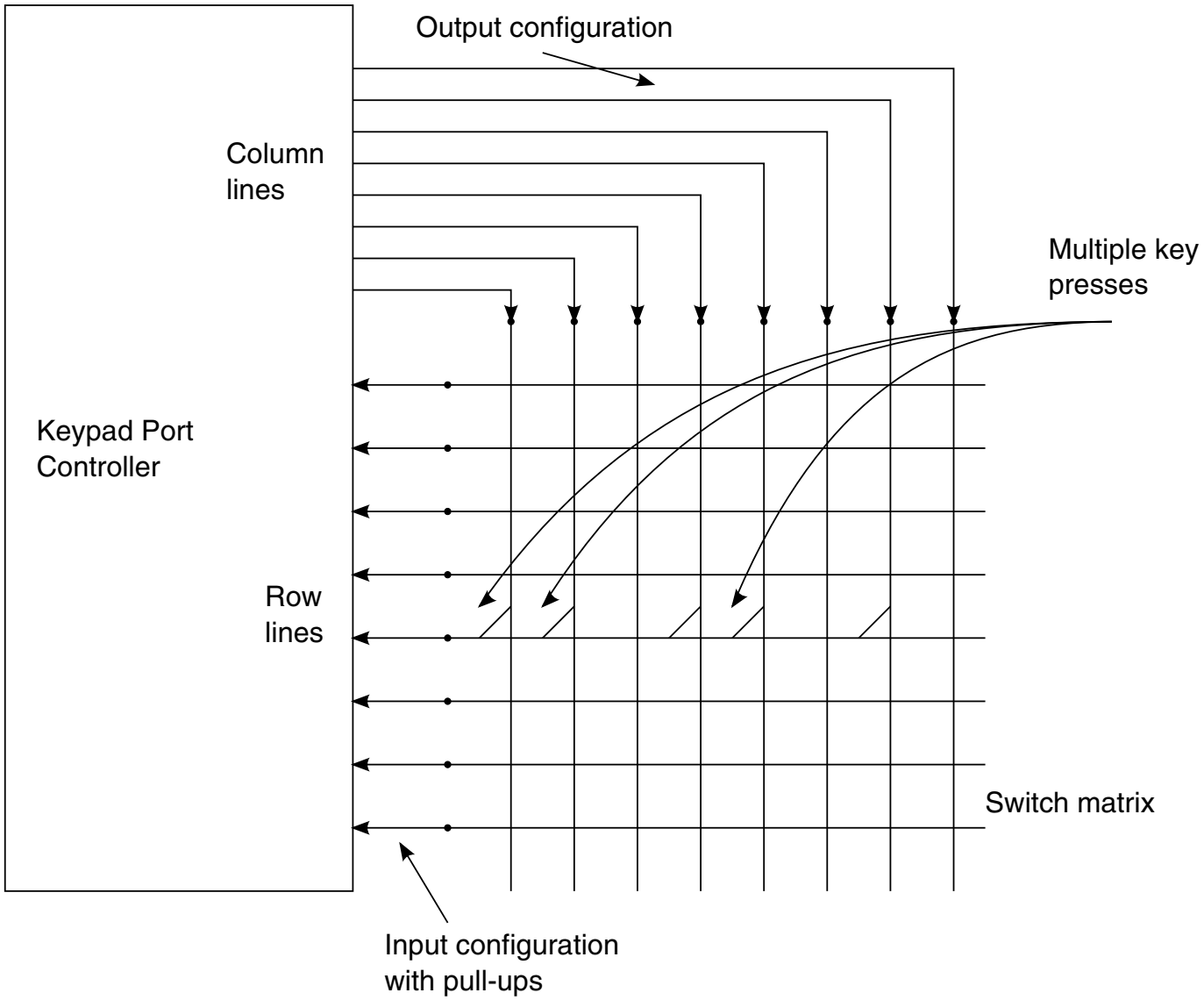


Figure 36-4. Multiple Key Presses on Same Row Line (Simplified View)

NOTE

An n key rollover is a technique with which the system can recognize the order in which keys are pressed.

36.3.6.1 Ghost Key Problem and Correction

The KPP detects if one or multiple keys are pressed or released. In the case where a simple keypad matrix with two-contact switches is used, there is a chance of "ghost" key detection when three or more keys are pressed. This is a limitation imposed by such a keypad matrix.

As can be seen in Figure 36-5, three keys pressed simultaneously can cause a short between the column currently "scanned" by the software and another column. Depending on the location of the third key pressed, a "ghost" key press may be detected.

However, this can be corrected by using a keypad matrix that provides "ghost" key protection. Such a matrix implements a one-way "diode" at all keypad points between rows and columns. This way, the multiple pressing of three keys will not cause a short at a fourth key (see Figure 36-6).

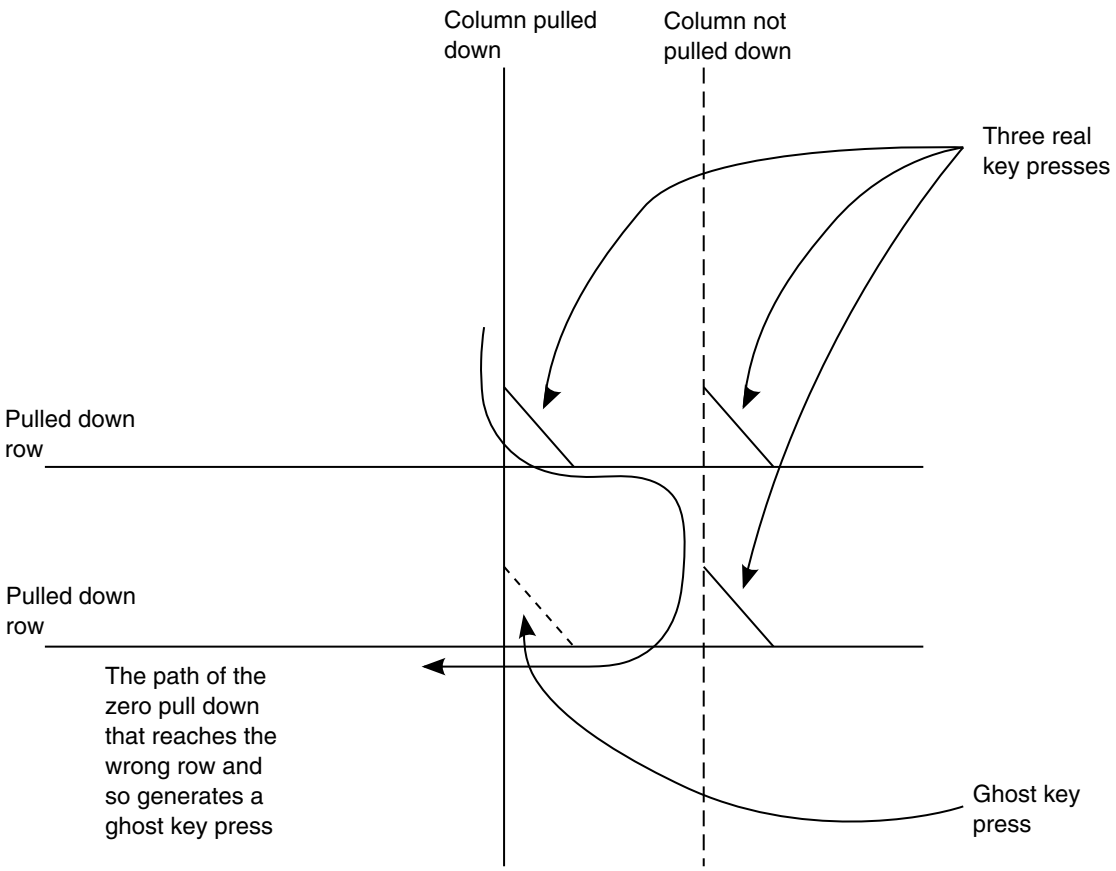


Figure 36-5. Decoding Wrong Three- Key-Presses

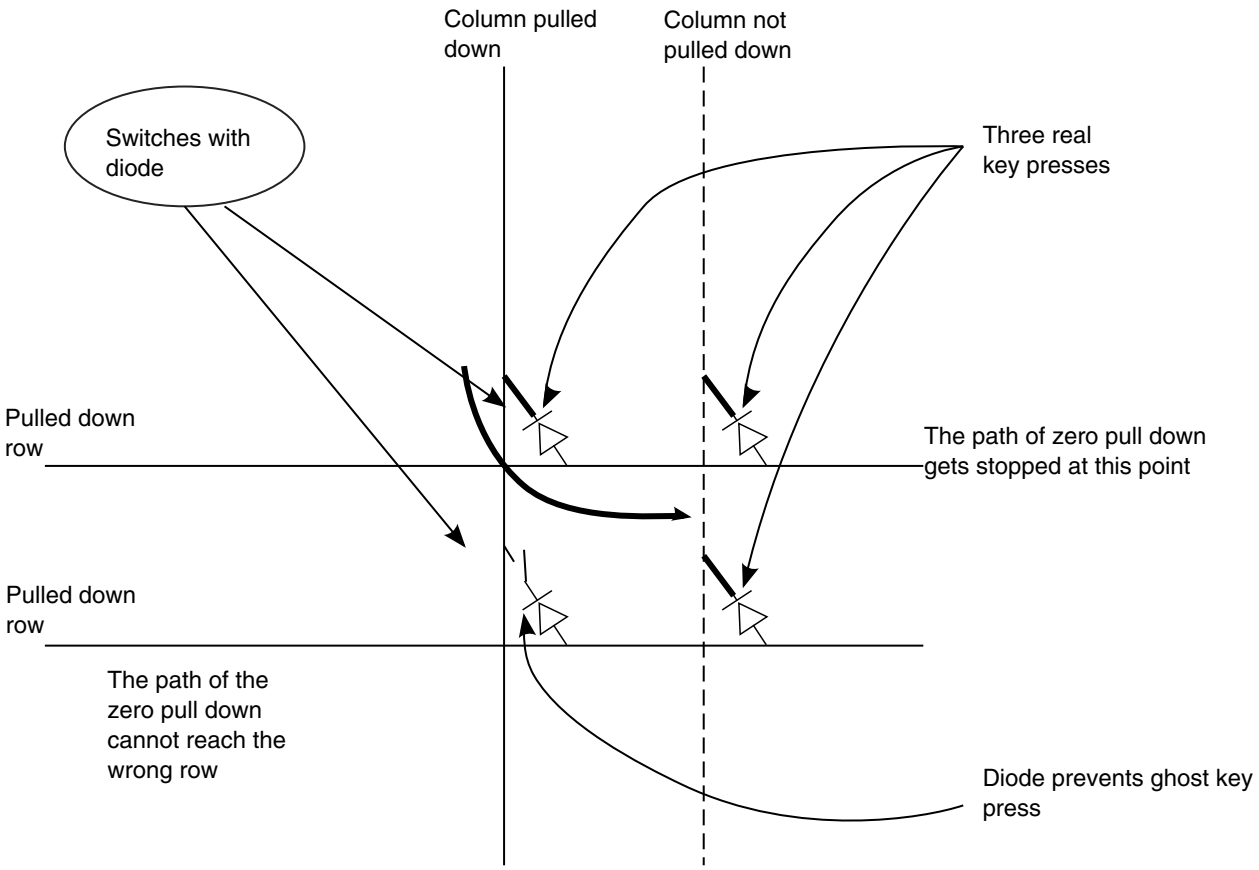


Figure 36-6. Matrix with "Ghost" Key Protections

36.3.7 3-Point Contact Keys Support

The KPP supports interfacing to a matrix consisting of 3-point contact keys. As shown in Figure 36-7, two points of such a key are connected to keypad lines, while a third point is connected to ground (low logic).

The keypad lines should be configured as input and a pull-up should to be present on these lines. When such a key is pressed, corresponding keypad lines go low and an interrupt is generated. There is no need to perform a scanning routine for identification of pressed key as it can be done by reading the keypad data-register. A limitation with such a matrix is that for every key at least one keypad row line should be used.

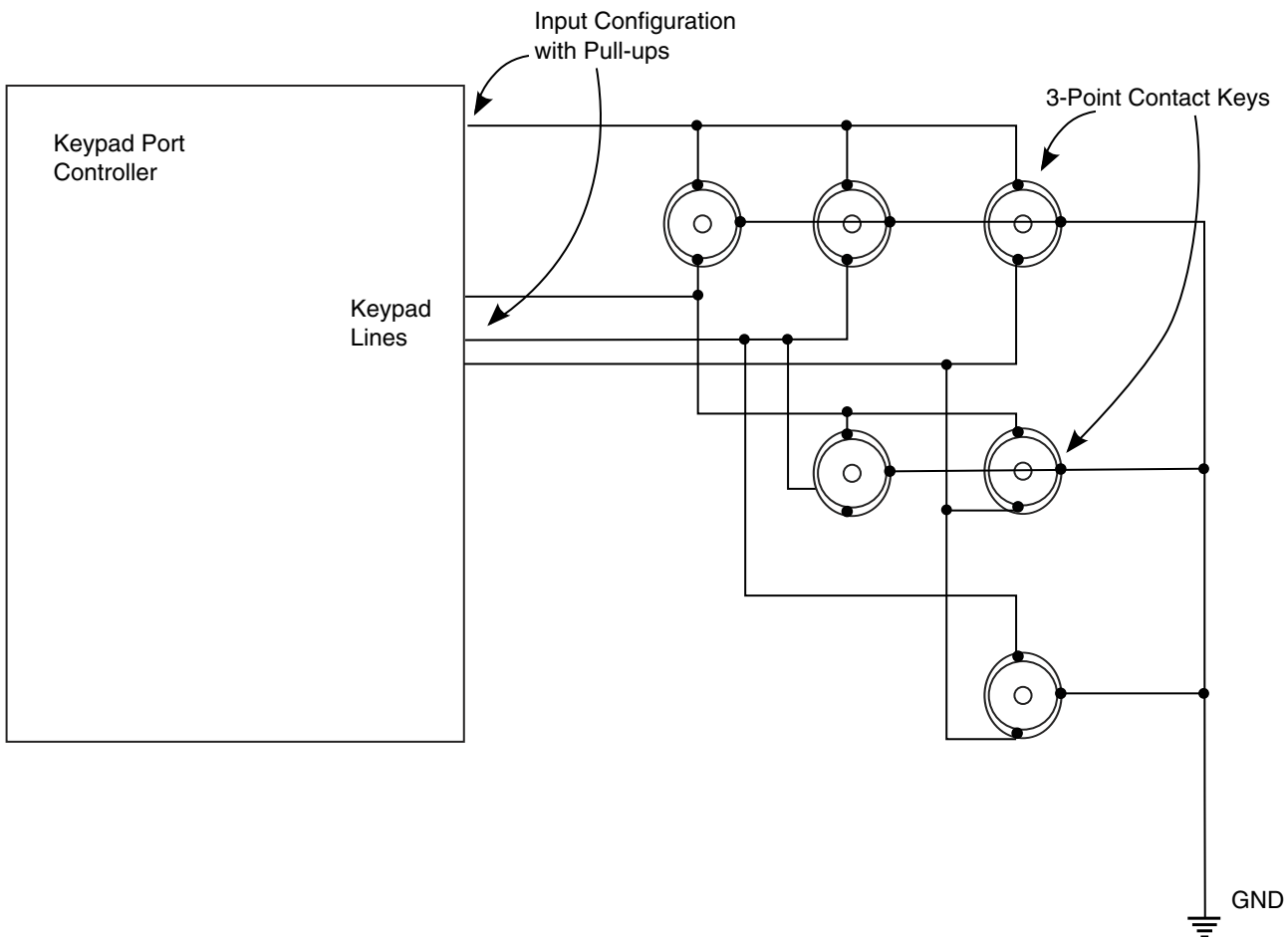


Figure 36-7. KPP Interface with 3-point Contact Key Matrix (Simplified View)

36.4 Initialization/Application Information

36.4.1 Typical Keypad Configuration and Scanning Sequence

Perform the following steps to configure the keypad:

1. Enable the number of rows in the keypad (KPP_KPCR[KRE]).
2. Write 0s to KPP_KPDR[KCD].
3. Configure the keypad columns as open-drain (KPP_KPCR[KCO]).
4. Configure columns as output (KPP_KDDR[KCDD]) and rows as input (KPP_KDDR[KRDD]).
5. Clear the KPKD Status Flag and Synchronizer chain.
6. Set the KDIE control bit, and clear the KRIE control bit (avoid false release events).
7. (The system is now in standby mode, and awaiting a key press.)

36.4.2 Key Press Interrupt Scanning Sequence

Perform the following steps to perform a keypad scanning routine:

1. Disable both (depress and release) keypad interrupts.
2. Write 1s to KPP_KPDR[KCD], setting column data to 1s.
3. Configure columns as totem pole outputs (for quick discharging of keypad capacitance).
4. Configure columns as open-drain.
5. Write a single column to 0, and other columns to 1.
6. Sample row inputs and save data. Multiple key presses can be detected on a single column.
7. Repeat Steps 2-6 for remaining columns.
8. Return all columns to 0 in preparation for standby mode.
9. Clear KPKD and KPKR status bit(s) by writing to a "1"; set the KPKR synchronizer chain by writing a "1" to the KPP_KRSS register; and clear the KPKD synchronizer chain by writing a "1" to the KDSC register.
10. Re-enable the appropriate keypad interrupt(s) so that the KDIE detects a key hold condition, or the KRIE detects a key-release event.

36.4.3 Additional Comments

The order of key press detection can be done in software only. Therefore, the software may need to run the scan routines at very short intervals of time per the application's demands. The reason that such functionality cannot be put in the KPP is that the block is limited by the number of external pins.

For the keys that require a very precise order (such as game keys), individual GPIO pins may be more useful.

36.5 Programmable Registers

The KPP contains four registers.

KPP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F9_4000	Keypad Control Register (KPP_KPCR)	16	R/W	0000h	36.5.1/ 2243
53F9_4002	Keypad Status Register (KPP_KPSR)	16	R/W	0400h	36.5.2/ 2244
53F9_4004	Keypad Data Direction Register (KPP_KDDR)	16	R/W	0000h	36.5.3/ 2246
53F9_4006	Keypad Data Register (KPP_KPDR)	16	R/W	0000h	36.5.4/ 2246

36.5.1 Keypad Control Register (KPP_KPCR)

The Keypad Control Register determines which of the eight possible column strobes are to be open drain when configured as outputs, and which of the eight row sense lines are considered in generating an interrupt to the core.

It is up to the programmer to ensure that pins being used for functions other than the keypad are properly disabled. The KPP_KPCR register is byte- or half-word-addressable.

Address: KPP_KPCR is 53F9_4000h base + 0h offset = 53F9_4000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	KCO								KRE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KPCR field descriptions

Field	Description
15–8 KCO	Keypad Column Strobe Open-Drain Enable. Setting a column open-drain enable bit (KCO7-KCO0) disables the pull-up driver on that pin. Clearing the bit allows the pin to drive to the high state. This bit has no effect when the pin is configured as an input.

Table continues on the next page...

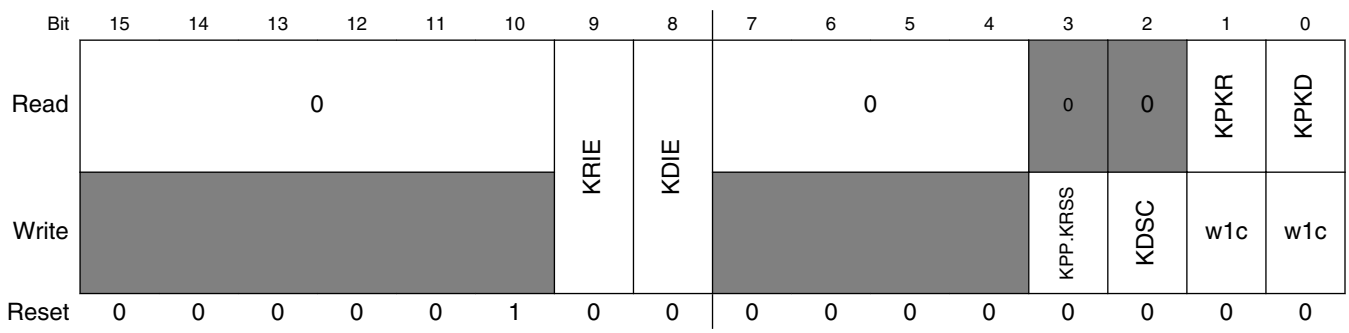
KPP_KPCR field descriptions (continued)

Field	Description
	NOTE: Configuration of external port control logic (for example, IOMUX) should be done properly so that the KPP controls an open-drain enable of the pin. 0 Column strobe output is totem pole drive. 1 Column strobe output is open drain.
7–0 KRE	Keypad Row Enable. Setting a row enable control bit in this register enables the corresponding row line to participate in interrupt generation. Likewise, clearing a bit disables that row from being used to generate an interrupt. This register is cleared by a reset, disabling all rows. The row-enable logic is independent of the programmed direction of the pin. Writing a "0" to the data register of the pins configured as outputs will cause a keypad interrupt to be generated if the row enable associated with that bit is set. 0 Row is not included in the keypad key press detect. 1 Row is included in the keypad key press detect.

36.5.2 Keypad Status Register (KPP_KPSR)

The Keypad Status Register reflects the state of the key press detect circuit. The KPP_KPSR register is byte- or half-word-addressable.

Address: KPP_KPSR is 53F9_4000h base + 2h offset = 53F9_4002h



KPP_KPSR field descriptions

Field	Description
15–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9 KRIE	Keypad Release Interrupt Enable. The software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKR is set. 1 An interrupt request is generated when KPKR is set.

Table continues on the next page...

KPP_KPSR field descriptions (continued)

Field	Description
8 KDIE	Keypad Key Depress Interrupt Enable. Software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice-versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKD is set. 1 An interrupt request is generated when KPKD is set.
7–4 Reserved	This read-only field is reserved and always has the value zero. Reserved, should be cleared
3 KPP.KRSS	Key Release Synchronizer Set. Self-clear bit. The Key release synchronizer is set by writing a logic one into this bit. Reads return a value of "0". 0 No effect 1 Set bits which sets keypad release synchronizer chain
2 KDSC	Key Depress Synchronizer Clear. Self-clear bit. The Key depress synchronizer is cleared by writing a logic "1" into this bit. Reads return a value of "0". 0 No effect 1 Set bits that clear the keypad depress synchronizer chain
1 KPKR	Keypad Key Release. The keypad key release (KPKR) status bit is set when all enabled rows are detected high after synchronization (the KPKR status bit will be set when cleared by a reset). The KPKR bit may be used to generate a maskable key release interrupt. The key release synchronizer may be set high by software after scanning the keypad to ensure a known state. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by software prior to the system exiting the state it represents. Reset value of register is "0" as long as reset is asserted. However when reset is de-asserted, the value of the register depends upon the external row pins and can become "1". 0 No key release detected 1 All keys have been released
0 KPKD	Keypad Key Depress. The keypad key depress (KPKD) status bit is set when one or more enabled rows are detected low after synchronization. The KPKD status bit remains set until cleared by the software. The KPKD bit may be used to generate a maskable key depress interrupt. If desired, the software may clear the key press synchronizer chain to allow a repeated interrupt to be generated while a key remains pressed. In this case, a new interrupt will be generated after the synchronizer delay (4 cycles of the low frequency reference clock elapses if a key remains pressed. This functionality can be used to detect a long key press. This allows detection of additional key presses of the same key or other keys. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by the software prior to the system exiting the state it represents. 0 No key presses detected 1 A key has been depressed

36.5.3 Keypad Data Direction Register (KPP_KDDR)

The bits in the KPP_KDDR control the direction of the keypad port pins. The upper eight bits in the register affect the pins designated as column strobes, while the lower eight bits affect the row sense pins. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding port pin as an input. For the Keypad Row DDR, an internal pull-up is enabled if the corresponding bit is clear. This register is cleared by a reset, configuring all pins as inputs. The KPP_KDDR register is byte- or half-word addressable.

NOTE

When a pin is used as row pin for keypad purposes, all corresponding pull-ups should be enabled at the upper level (for example, IOMUX) when the bit in row DDR is cleared.

Address: KPP_KDDR is 53F9_4000h base + 4h offset = 53F9_4004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read / Write	KCDD								KRDD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KDDR field descriptions

Field	Description
15–8 KCDD	Keypad Column Data Direction Register. Setting a bit configures the corresponding COL*n* pin as an output (where *n* = 7 through 0). 0 COLn pin is configured as an input. 1 COLn pin is configured as an output.
7–0 KRDD	Keypad Row Data Direction. Setting a bit configures the corresponding ROW*n* pin as an output (where *n* = 7 through 0). 0 ROWn pin configured as an input. 1 ROWn pin configured as an output.

36.5.4 Keypad Data Register (KPP_KPDR)

This 16-bit register is used to access the column and row data. Data written to this register is stored in an internal latch, and for each pin configured as an output, the stored data is driven onto the pin. A read of this register returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register.

The KPP_KPDR register is byte- or half-word addressable. This register is not initialized by a reset. Valid data should be written to this register before any bits are configured as outputs.

Address: KPP_KPDR is 53F9_4000h base + 6h offset = 53F9_4006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCD								KRD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KPDR field descriptions

Field	Description
15–8 KCD	Keypad Column Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write "0" from/to column ports 1 Read/Write "1" from/to column ports
7–0 KRD	Keypad Row Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write "0" from/to row ports 1 Read/Write "1" from/to row ports

Chapter 37
Enhanced LCD Interface (eLCDIF)

37.1 Overview

The eLCDIF is a general purpose display controller used to drive a wide range of display devices varrying in size and capability. Many of these displays have had an asynchronous parallel MPU interface for command and data transfer to an integrated frame buffer. There are other popular displays that support moving pictures and require the RGB interface mode (called DOTCLK interface in this document) or the VSYNC mode for high-speed data transfers. In addition to these displays, it is also common to provide support for digital video encoders that accept ITU-R BT.656 format 4:2:2 YCbCr digital component video and convert it to analog TV signals. The eLCDIF block supports these interfaces by providing fully programmable functionality.

The block has several major features:

- Bus master interface to source frame buffer data for display refresh and a DMA interface to manage input data transfers from the LCD requiring minimal ARM platform overhead.
- 8/16/18/24/32 bit LCD data bus support available depending on I/O mux options.
- Programmable timing and parameters for MPU, VSYNC and DOTCLK LCD interfaces to support a wide variety of displays.
- ITU-R BT.656 mode (called Digital Video Interface or DVI mode here) including progressive-to-interlace feature and RGB to YCbCr 4:2:2 color space conversion to support 525/60 and 625/50 operation.

37.2 Operation

Bus Interface Mechanisms through Initializing the eLCDIF, describe the internal pipeline for the MPU write/read interface, VSYNC, DOTCLK, and DVI interfaces. Differences for each mode are then described in separate sections, as follows:

- MPU Interface
- VSYNC Interface
- DOTCLK Interface
- ITU-R BT.656 Digital Video Interface (DVI)

eLCDIF pin usage by interface mode is described in eLCDIF Pin Usage by Interface Mode.

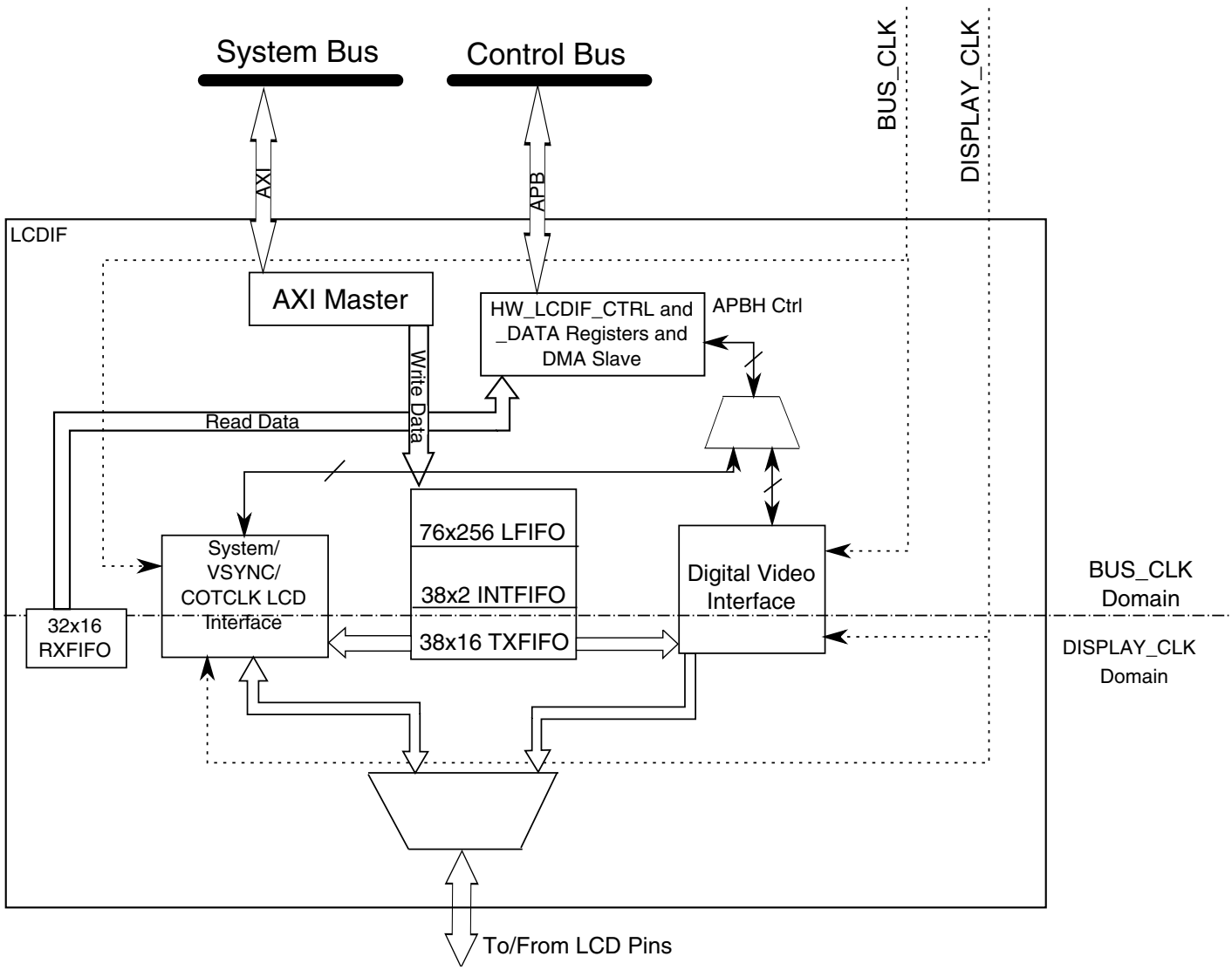


Figure 37-1. Top-Level Block Diagram of eLCDIF subsystem

37.2.1 Bus Interface Mechanisms

The eLCDIF block has memory-mapped control, data and status registers. It provides two efficient methods of transferring data to/from an external display or digital video encoder, the APB DMA slave for reads and AXI bus master for writes. The bus interface runs off

the BUS_CLK domain, while the display interface runs on the DISPLAY_CLK domain. There are three main FIFOs in the block datapath. The LFIFO in the write datapath is a 2K byte synchronous FIFO to manage external DRAM access latency. The TXFIFO is a 16 word deep asynchronous FIFO that provides data crossing between the two clock domains. In the read datapath, there is a 16-deep asynchronous RXFIFO that is read by DMA operation. The following sub-sections describe the two system bus interface mechanisms.

37.2.1.1 Bus Master Operation in Write/Display Modes

The eLCDIF block has a bus master interface that initiates requests for data to drive the display. The LCDIF_MASTER bit must be set to 1 to enable the bus master interface. Software should program all control registers required to transfer the frame sequence.

In the MPU and VSYNC modes, single frames are transferred. When a complete frame is transferred, eLCDIF enters idle and clears the RUN bit in the CTRL register. For subsequent frame transmission, the eLCDIF setup sequence should be repeated.

The DOTCLK and DVI modes are used to refresh the display at the desired refresh rate and resolution. These modes are used to drive displays that do not integrate a display buffer memory. When the display is refreshed, the eLCDIF will automatically update the LCDIF_CUR_BUF_ADDR register with the value in LCDIF_NEXT_BUF_ADDR at the end of current frame and start fetching the next frame from the new address. If the LCDIF_NEXT_BUF_ADDR register was not updated within a frame refresh cycle, eLCDIF will keep transmitting the last frame until a new value is programmed into that register.

eLCDIF also provides the capability of interlacing a progressive frame by fetching odd lines in the first field and then fetching even lines in the second field. This feature can be used in the DVI mode and can be turned on by setting the INTERLACE_FIELDS bit in the LCDIF_CTRL1 register.

37.2.1.2 System Bus Master Performance

The performance of the eLCDIF block can be controlled by changing the burst length and the outstanding cycle issuing capability depending on the memory bandwidth requirements. Two fields in the LCDIF_CTRL2 register will throttle system memory requests. The LCDIF_CTRL2_OUTSTANDING_REQS field will control how many requests the eLCDIF can have in flight on any given clock cycle. This should be programmed based on the expected system bus latency for returned read data. Also, the LCDIF_CTRL2_BURST_LEN_8 bit will set the number of 64 bit words requested for

each eLCDIF system bus request to either 8 or 16 QWORDS. Generally, 4 outstanding requests of length 16 will provide enough performance to drive any standard display resolution. These configuration bits are intended to change the access pattern of the eLCDIF to optimize system bus throughput when other system masters will contend for system memory resources.

The LCDIF_THRES register can also be used to optimize bus throughput and power consumption. The LCDIF_THRES_FASTCLOCK value can be used to change the BUS_CLK frequency when the number of pixels in the LFIFO is below this programmed threshold. The BUS_CLK should be set to a frequency that will exceed the bandwidth requirements of the desired display. In systems supporting the dynamic frequency modulation of the bus clock frequency, the BUS_CLK will be driven at the desired frequency when the number of pixels is below the FASTCLOCK threshold. When the number of pixels is above this threshold, the BUS_CLK will be reduced by a divide factor selected in the centralized clock control module. This will provide an average clock frequency that will exactly meet the pixel bandwidth requirements over time, and minimize power consumption to meet the display bandwidth requirements. During horizontal or vertical blanking intervals, when the LFIFO is full with data for the next active display interval, the clock can be reduced to a slow frequency for extended periods to reduce power consumption.

The LCDIF_THRES_PANIC value can be used to raise the priority of requests initiated by the eLCDIF to alter how the eLCDIF requests are arbitrated by the system bus infrastructure. The panic output control signal is raised when the number of 32bpp pixel equivalents in the LFIFO is less than this programmed value. Since the LFIFO is arranged as a 256x64bit quadword FIFO, it contains two 32bpp pixels per quadword, or 512 32bpp pixels total. To set the panic output when 3/4s of the LFIFO is empty, set the LCDIF_THRES_PANIC value to 3/4 * 512, or 128. The panic signal output is used to assess higher priority to eLCDIF system requests to avoid eLCDIF under run errors during periods of high system bandwidth utilization.

The features available with the LCDIF_THRES register require support from system clocking and dynamic priority control. Refer to the appropriate block documentation to assess the system support for these features.

37.2.1.3 DMA Operation in MPU Read Mode

Data can be read from an external display using the APBDMA interface. A bus master mode is not available to read data via the display interface. The read data interface is typically used with MPU type interfaces.

The eLCDIF block has a DMA slave interface that uses the APBDMA to manage read data transfers. The data transfers are managed by a linked data descriptor chain executed by the APBDMA engine and the DMA master/slave interface between the eLCDIF and the APBDMA engine. This mode is enabled by setting the LCDIF_MASTER bit to 0.

37.2.2 Write Data Path

eLCDIF supports raster based frame buffers and there is no support for tiled buffers. There are several options to accommodate endianness of display buffers in memory before the data is processed for the external display. The INPUT_DATA_SWIZZLE field in the LCDIF_CTRL register provides the following options for data word multiplexing:

```
00 (0): No swizzle (little-endian)
01 (1): Swap bytes 0 and 3, swap bytes 1 and 2 (big-endian)
10 (2): Swap half-words
11 (3): Swap bytes within each half-word
```

The WORD_LENGTH field of LCDIF_CTRL register indicates the input data/pixel format. LCDIF_TRANSFER_COUNT register denotes how much data is contained in each frame. The H_COUNT field of this register indicates the number of pixels per line and V_COUNT indicates the total number of lines per frame. A special bit field in the LCDIF_CTRL1 register, called the BYTE_PACKING_FORMAT, can be used to specify which bytes within the 32-bit word are going to be valid. For example, if the entire 32-bit word is valid, BYTE_PACKING_FORMAT should be set to 0xF, if only lower 3 bytes of each word in the frame buffer are valid, then BYTE_PACKING_FORMAT should be set to 0x7.

The LCD_DATABUS_WIDTH field in LCDIF_CTRL register suggests the width of the bus going to the external display controller. There is an option to source all 32 bits of the input word and transfer it to the output I/O display interface. Refer to the system I/O muxing options for support of this feature. If the LCD_DATABUS_WIDTH is not the same as WORD_LENGTH, eLCDIF will perform RGB to RGB color space conversion. For example, if the input frame has fewer bits per pixel than the display, as in a 16 bpp input frame going to 24 bpp LCD, eLCDIF will pad the MSBs of each color to the LSBs of the same color for each pixel. If the input frame has more bits per pixel than the display, for example, 24 bpp input frame going to 16 bpp LCD, eLCDIF will drop the LSBs of each color channel to convert to the lower color depth. eLCDIF also has the capability to support delta pixel displays by swizzling the R, G and B colors of each pixel in the odd and even lines of the frame separately by programming the ODD_LINE_PATTERN and the EVEN_LINE_PATTERN bit fields. This operation occurs after the RGB-to-RGB color space conversion operation.

eLCDIF also supports RGB to YCbCr 4:2:2 color space conversion. This is useful in the DVI mode since the TV encoder requires input in YCbCr 4:2:2 format. The LCDIF_CSC* registers have complete programmability over the CSC coefficients and offsets. The values must be written into these registers in the signed two's complement format.

The following list shows how the different input/output combinations can be obtained:

- WORD_LENGTH=1 indicates that the input is 8-bit data. This is most likely going to be used for sending commands in MPU interface, or maybe a gray scale image. Any combination of BYTE_PACKING_FORMAT [3:0] is permissible.

  Limitation: H_COUNT must be a multiple of the sum of BYTE_PACKING_FORMAT [3], BYTE_PACKING_FORMAT [2], BYTE_PACKING_FORMAT [1] and BYTE_PACKING_FORMAT [0]. LCD_DATABUS_WIDTH must be 1, indicating an 8-bit data bus.

- WORD_LENGTH=0 implies the input frame buffer is RGB 16 bits per pixel. DATA_FORMAT_16_BIT field determines the pixels are RGB 555 or RGB 565.

  Limitation: BYTE_PACKING_FORMAT [3:0] should be 0x3 or 0xC if there is only one pixel per word. If there are two pixels per word, it should be 0xF and H_COUNT will be restricted to be a multiple of 2 pixels.

- WORD_LENGTH=2 indicates that input frame buffer is RGB 18 bits per pixel, that is, RGB 666. The valid RGB values can be left-aligned or right-aligned within a 32-bit word. The alignment of the valid 18 bits within a word is indicated by the DATA_FORMAT_18_BIT bit.

  Limitation: BYTE_PACKING_FORMAT can be 0x7, 0xE or 0xF. Packed pixels are not supported in this case. H_COUNT can be any number.

- WORD_LENGTH=3 indicates that the input frame-buffer is RGB 24 bits per pixel (RGB 888). If BYTE_PACKING_FORMAT [3:0] is 0x7, it indicates that there is only one pixel per 32-bit word and there is no restriction on H_COUNT. This is also the option that provides 32 bit output depending on the I/O muxing options available. The fourth byte, or bits [31:24], and connected to the I/Os if this muxing is available in the chip package.

  Limitation: If BYTE_PACKING_FORMAT [3:0] is 0xF, it indicates that the pixels are packed, that is, there are 4 pixels in 3 words or 12 bytes and H_COUNT must be a multiple of 4 pixels.

- YCBCR422_INPUT=1 implies that the input frame is in YCbCr 4:2:2 format. BYTE_PACKING_FORMAT must be 0xF.

Limitation: LCD_DATABUS_WIDTH must be 8-bit and H_COUNT must be a multiple of 2 pixels.

ODD_LINE_PATTERN and EVEN_LINE_PATTERN must be 0 when any of RGB_TO_YCBCR422_CSC or INTERLACE_FIELDS or YCBCR422_INPUT bits is 1.

After the RGB to RGB or RGB to YCbCr 4:2:2 color space conversions, there is one more opportunity to swizzle the data before sending it out to the display or the encoder. This can be done with the CSC_DATA_SWIZZLE field in the LCDIF_CTRL register, and it provides the same options as the INPUT_DATA_SWIZZLE register.

Finally, there is an option to shift the output data before sending it out to the display. This is done based on the SHIFT_DIR and SHIFT_NUM_BITS fields in LCDIF_CTRL register.

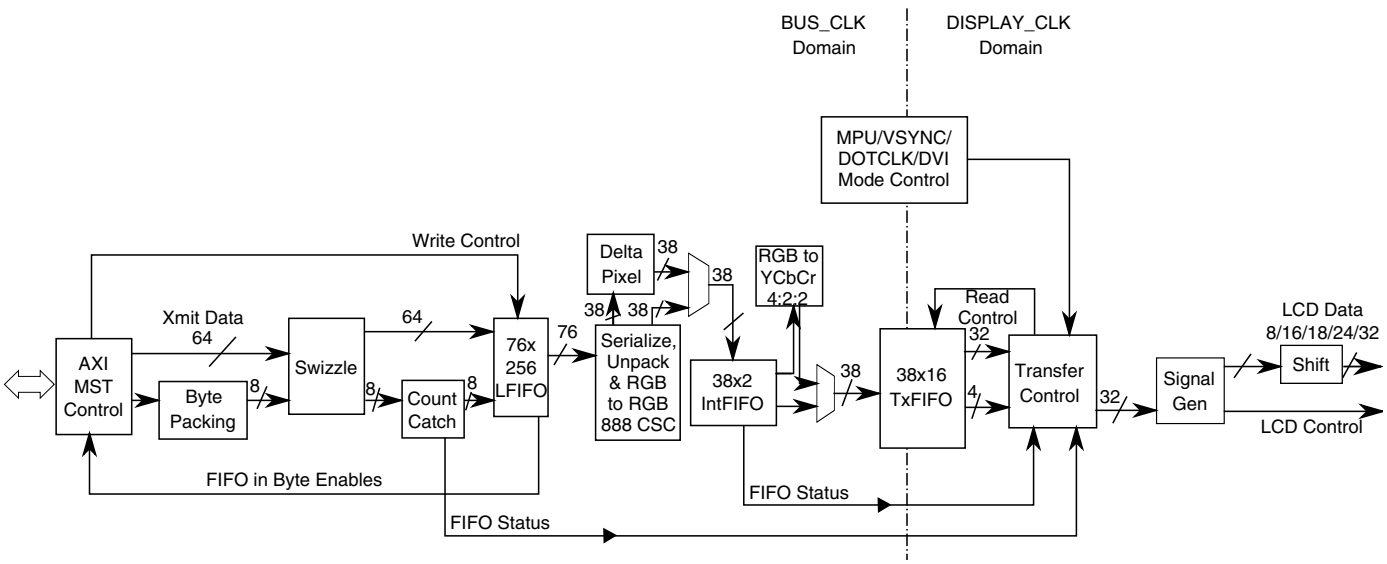


Figure 37-2. General Operations in Write Data Path

The examples in the following figures illustrate some different combinations of register programming for write mode. Assume that the data transferred over the system bus within a 32 bit word is organized as {A7-A0, B7-B0, C7-C0, D7-D0} in 8-bit mode and {A15-A0, B15-B0} in 16-bit mode.

In this example, all 32 bits of the input word are transferred out over an 8 bit display bus. Each byte within the 32 bit word is shifted to the right with zeros appended to I/O bits D[7:6]. The input data bits [7:2] are shifted to the right by 2 bits and presented on the D[5:0].

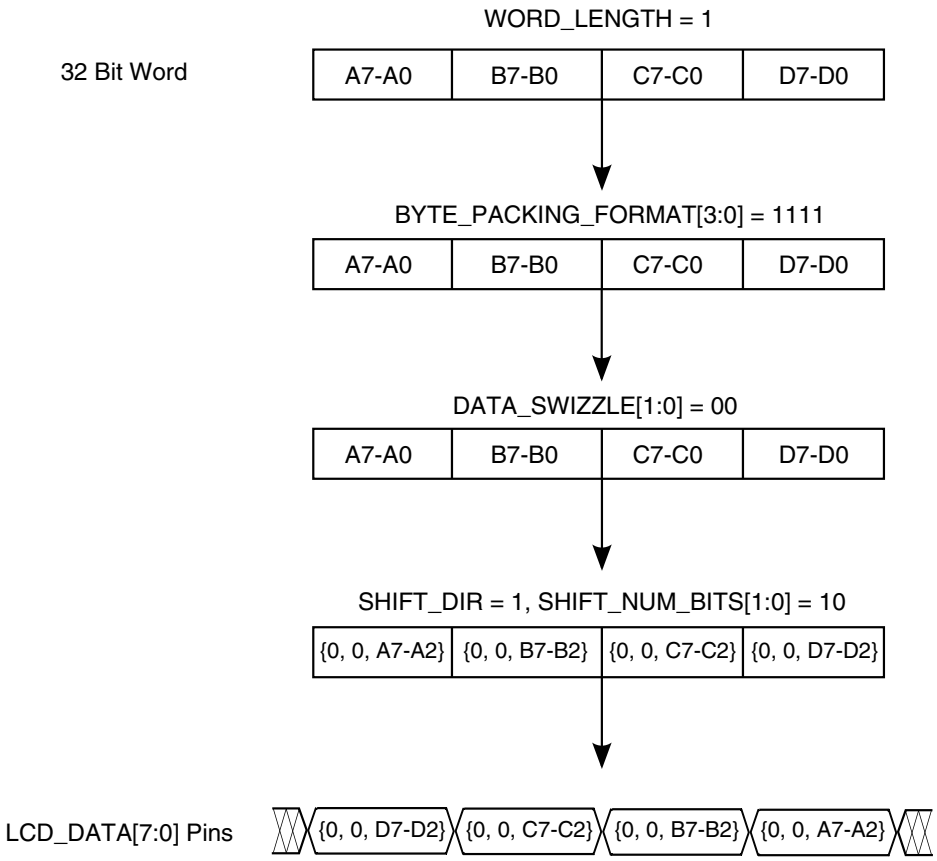


Figure 37-3. Register programming for write mode

In this 8 bit display interface example, one byte of the input word is deleted and not transferred over the external 8 bit display interface. This mode could be used to transfer 24bpp pixels over the 8 bit interface. In this case, the 4th unused byte is not transferred.

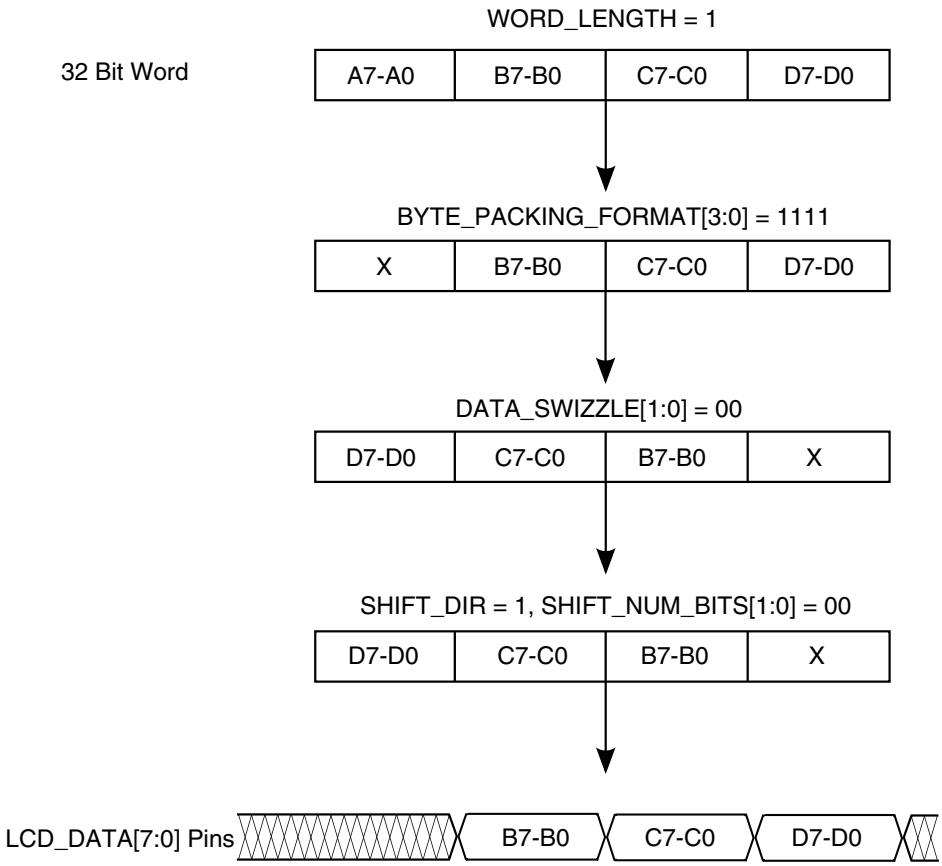


Figure 37-4. Register programming for write mode

The following example uses a 16 bit display interface. Each 16 bit half word is shifted to the right by two bits with zeros appended to the most significant two bits.

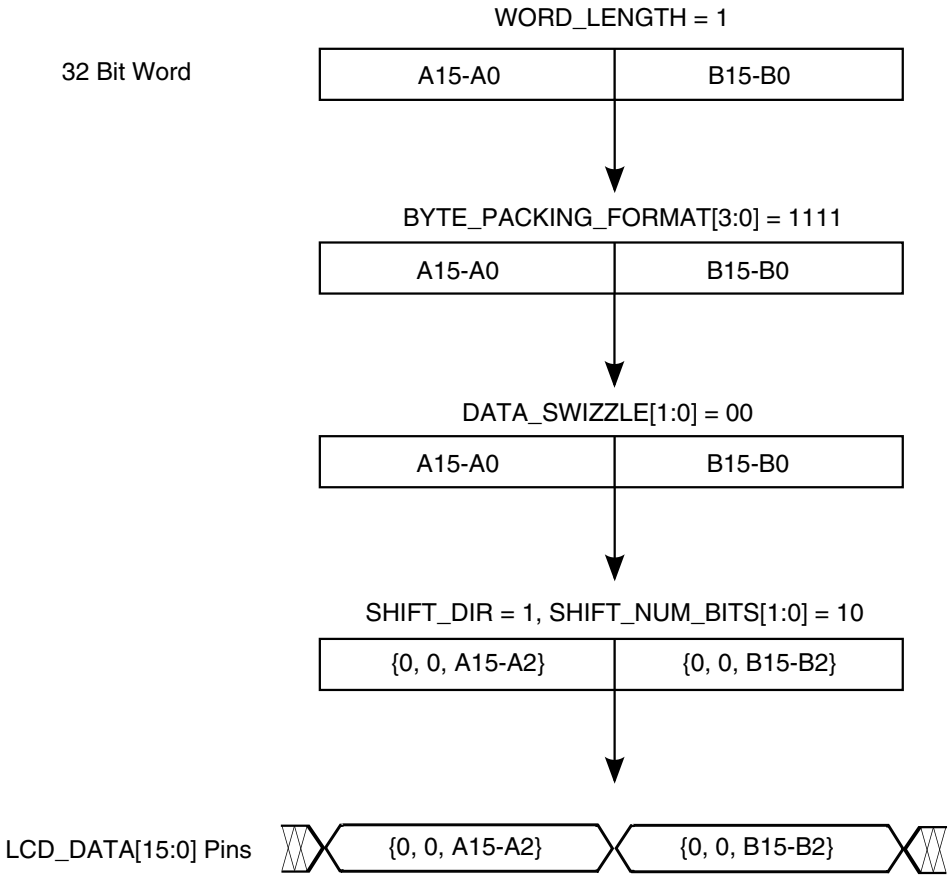


Figure 37-5. Register programming for write mode

This example indicates how an unpacked frame buffer can sourced for display. Only a single 16 bit half word within the 32 bit word is tranferred out via the 16 display bus.

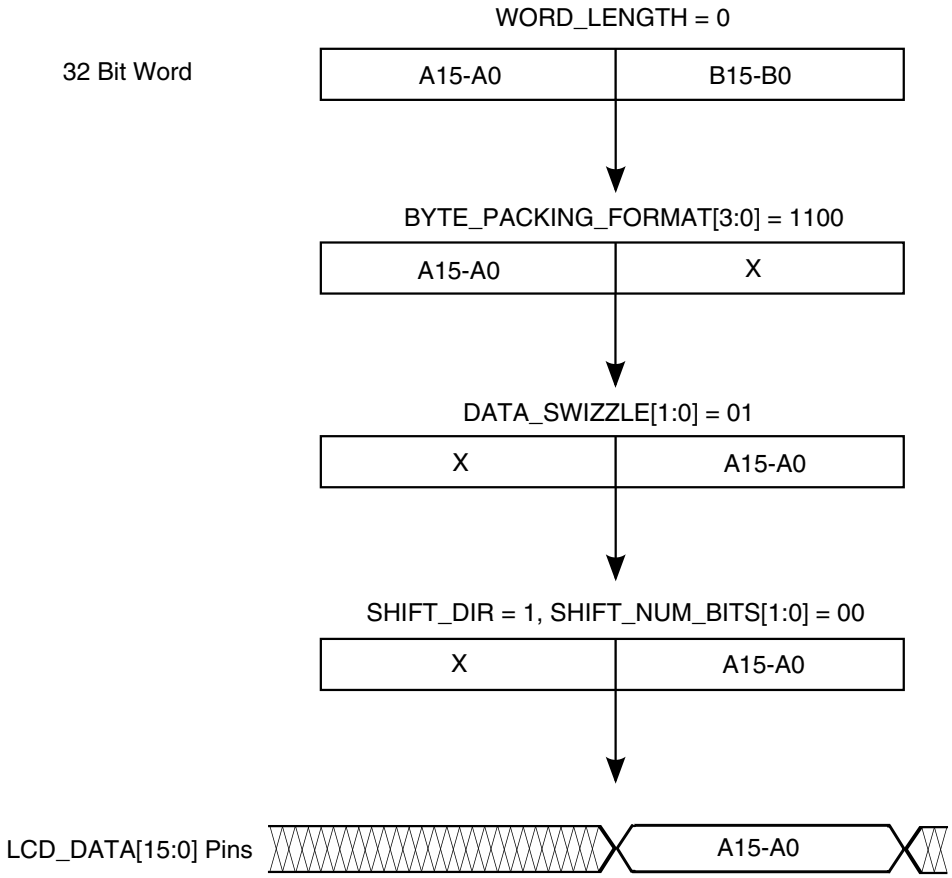


Figure 37-6. Register programming for write mode

37.2.3 Read Data Path

Figure 37-7 shows the MPU read data path in detail. eLCDIF can read from an external display that follows the 6800/8080 MPU protocol. The display bus width is determined by the LCD_DATABUS_WIDTH bit field. The data sampled at every read strobe is called a subword and the number of subwords that can be packed in a 32-bit word is given by the READ_MODE_NUM_PACKED_SUBWORDS bit field. The INITIAL_DUMMY_READ bit field directs the eLCDIF to skip the number of programmed subwords before starting to process read data. This feature is useful in the case of an LCD controller that returns the last written data the first time a read is issued, and then sends the correct data thereafter. SHIFT_DIR and SHIFT_NUM_BITS bit fields indicate whether the data needs to be shifted before getting stored in the internal registers. For example, a value of 2 in READ_MODE_NUM_PACKED_SUBWORDS if lcd databus width is 8 bits indicates two bytes should be packed in a 32-bit word, while if the lcd databus width is 16 bits, it indicates that two half words (or 4 bytes) should be packed.

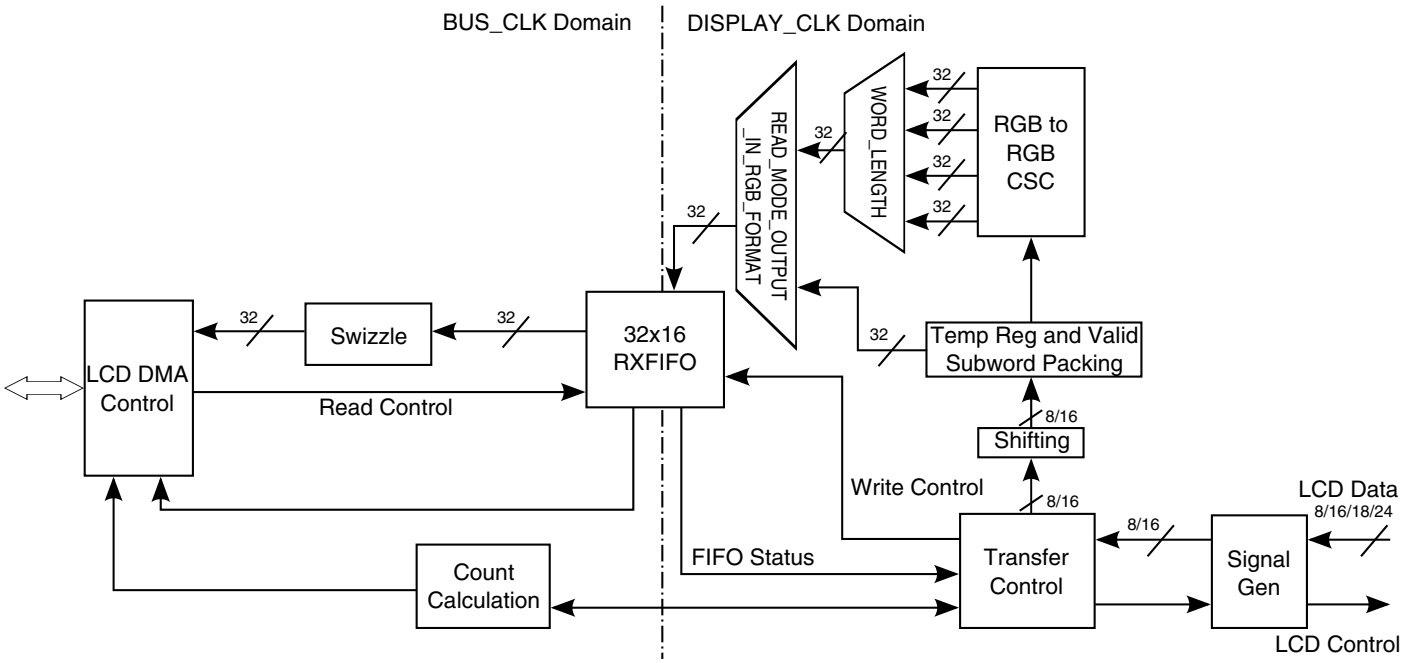


Figure 37-7. MPU Read Data Path

After the last subword within a word is reached, the block looks at the READ_PACK_DIR in the HW_LCDIF_CTRL2 register. If this bit is set, the block will swizzle the data, but only within the valid bytes, unlike in the write mode, where swizzle occurs across all 4 bytes. If the READ_MODE_OUTPUT_IN_RGB_FORMAT bit is set, eLCDIF will convert the data obtained from the READ_PACK_DIR operation into 24-bit unpacked RGB and then re-convert it into 16/18/24 bpp RGB depending on the WORD_LENGTH field. The DATA_FORMAT_16/18/24_BIT bit fields are also considered while converting to 24-bit unpacked RGB format. For example, if DATA_FORMAT_18_BIT is 1, the RGB666 data will be packed in the upper bits [31:4]of a 32-bit word, and that bit is 0, the data will be packed in the lower bits [17:0]. After all these operations, the data gets written into the RXFIFO.

The following figures show some examples of how data is handled in different MPU read modes.

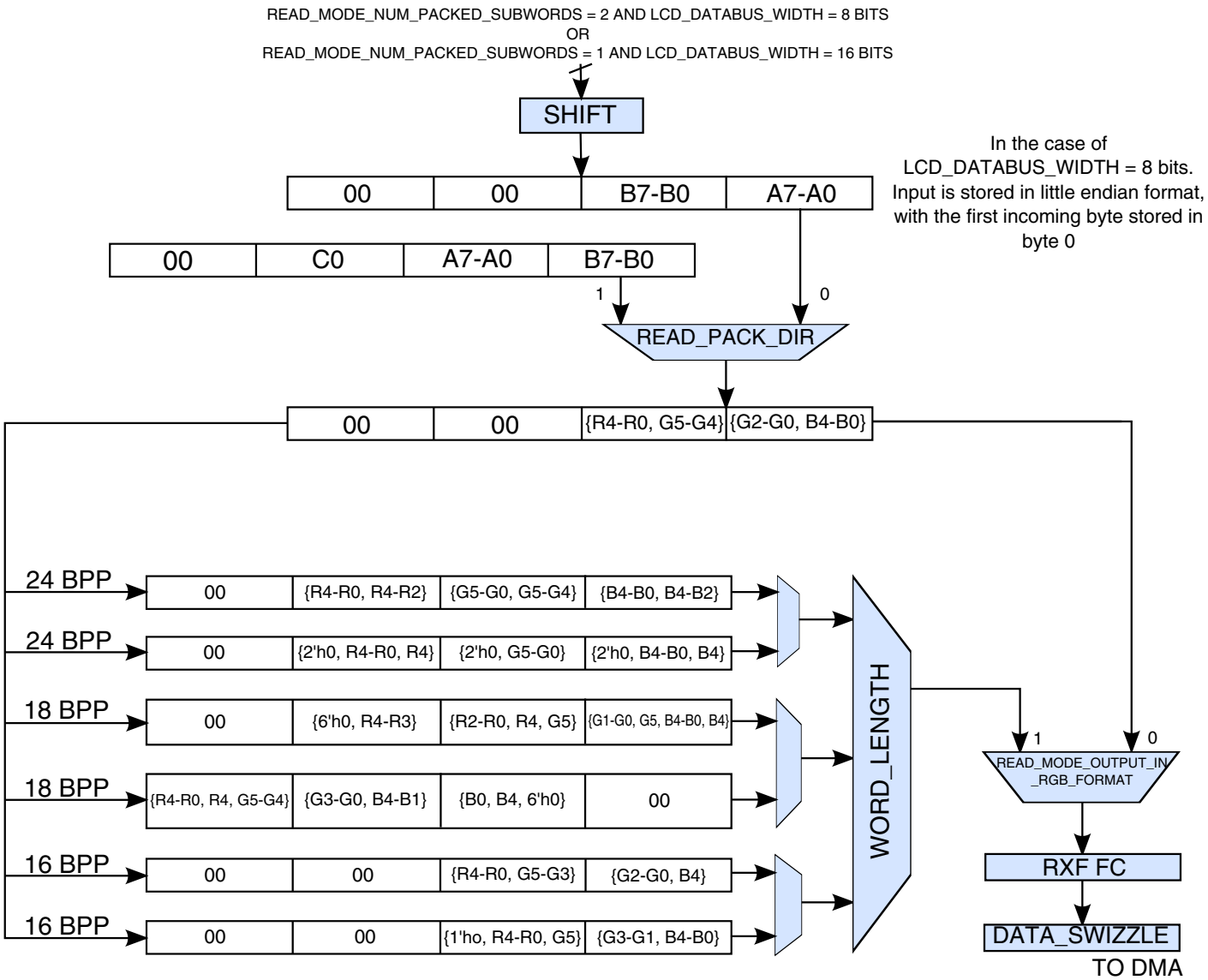


Figure 37-8. Data in MPU read mode

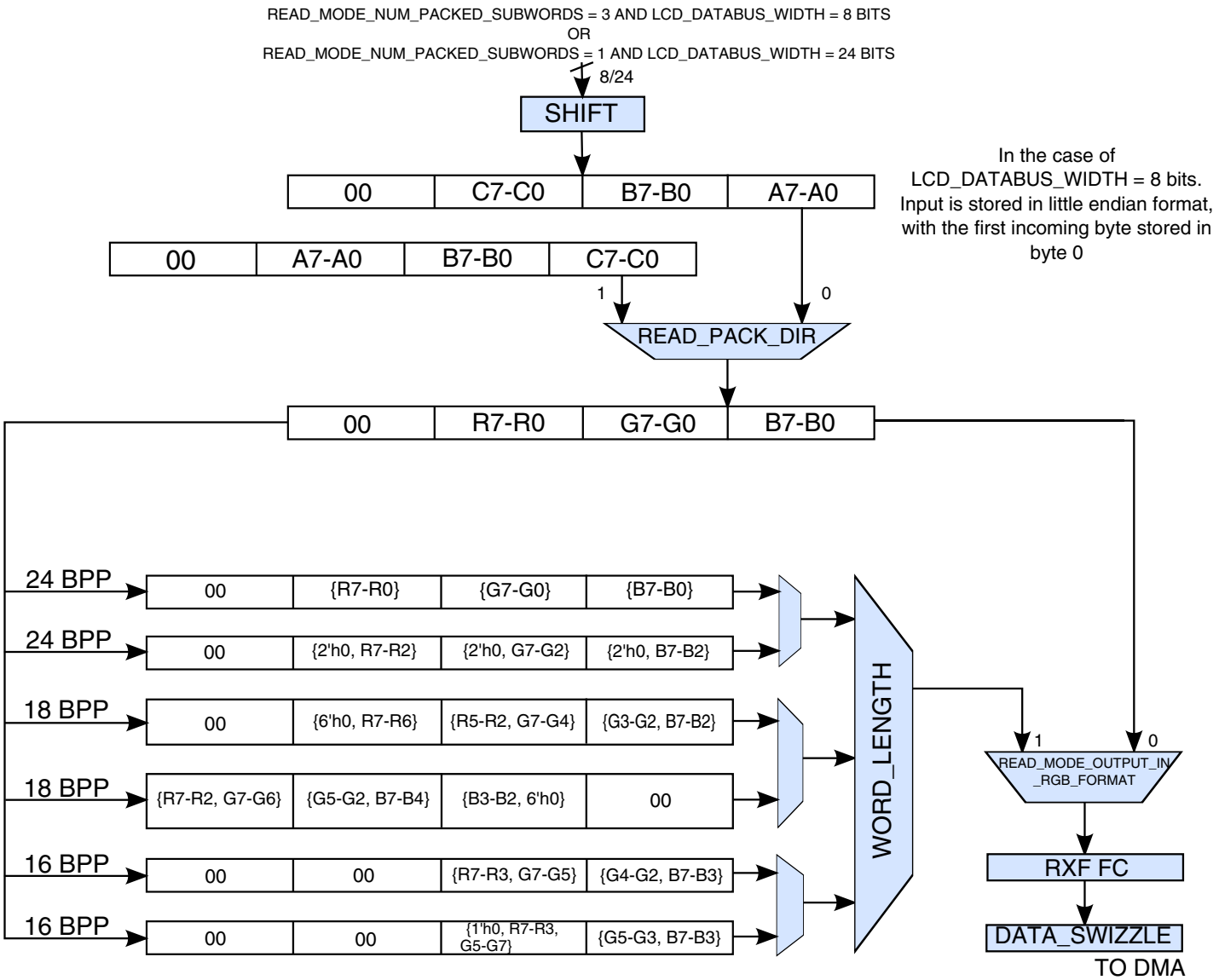


Figure 37-9. Data in MPU read mode

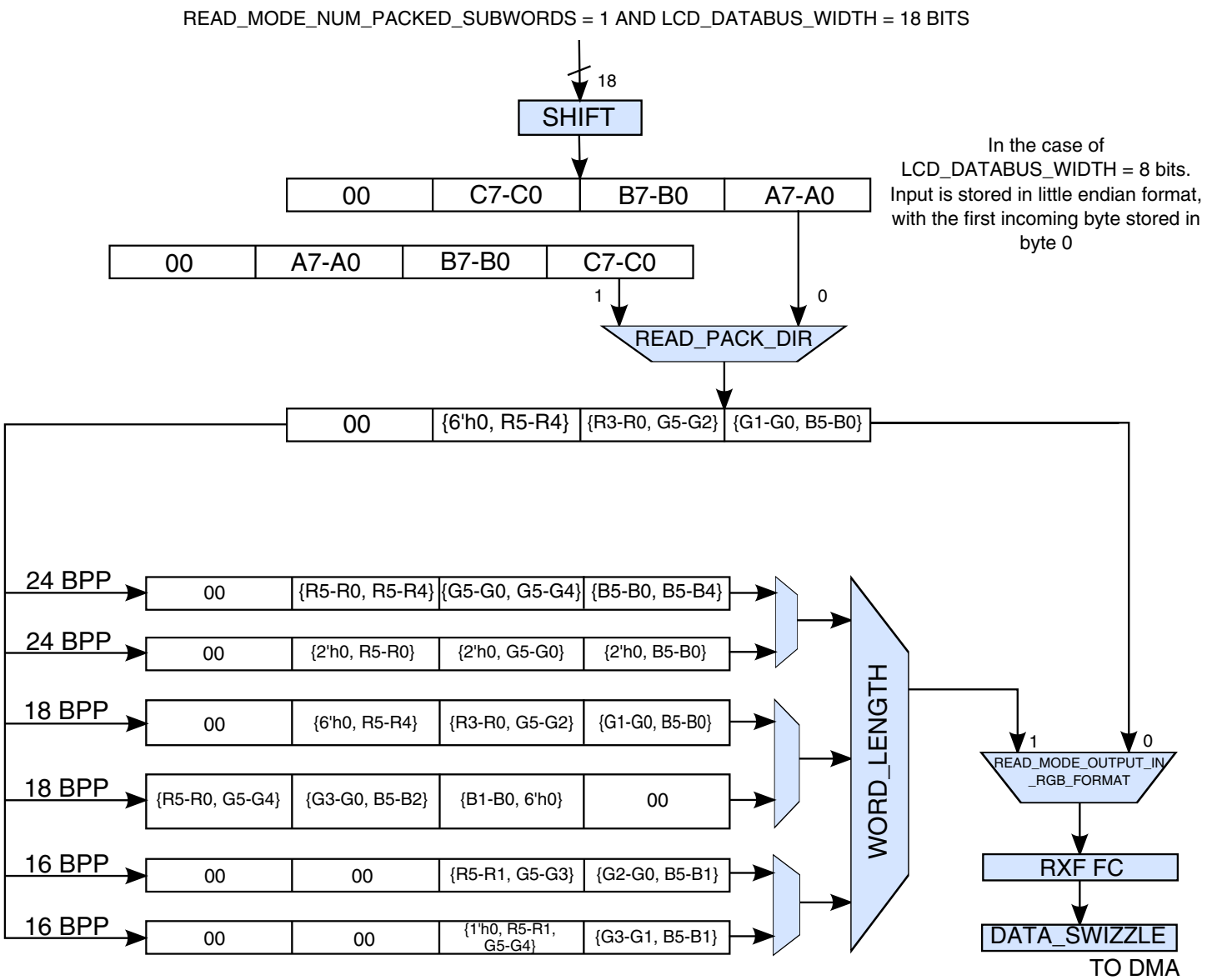


Figure 37-10. Data in MPU read mode

Restrictions:

READ_PACK_DIR should only be used if it is required to swizzle the subwords before doing RGB to RGB CSC, otherwise the DATA_SWIZZLE field should be used to swizzle across bytes.

READ_PACK_DIR must be 0 if LCD_DATABUS_WIDTH is 8 bits and READ_MODE_NUM_PACKED_SUBWORDS =1

If READ_MODE_OUTPUT_IN_RGB_FORMAT bit is set, the following restrictions should be followed:

- If LCD_DATABUS_WIDTH = 8 bits, then READ_MODE_NUM_PACKED_SUBWORDS <= 3.
- If LCD_DATABUS_WIDTH = 16/18/24 bits, then READ_MODE_NUM_PACKED_SUBWORDS = 1.

37.2.4 eLCDIF Interrupts

eLCDIF supports a number of interrupts to aid controlling and status reporting of the block. All the interrupts have individual mask bits for enabling or disabling each of them. They all get funneled through a single interrupt line connected to the interrupt collector (ICOLL).

The following list describes the different interrupts supported by eLCDIF:

- Underflow interrupt is asserted when the clock domain crossing FIFO (TXFIFO) becomes empty but the block is in active display portion during that time. Software should take corrective action to make sure that this does not happen.
- In the bus master mode, the overflow interrupt will be asserted if the block has requested more data than it's FIFOs could hold. In the read mode, it will be asserted if the RxFIFO becomes full and the block reads more data.
- VSYNC edge interrupt will be asserted every time a leading VSYNC edge occurs.
- Cur_frame_done interrupt occurs at the end of every frame in all modes except DVI. In DVI mode, if IRQ_ON_ALTERNATE_FIELDS bit is set, it will occur at the end of every frame, otherwise it will occur at the end of every field.

37.2.5 Initializing the eLCDIF

This section describes write modes and MPU read mode.

37.2.5.1 Write Modes

The following initialization steps are common to all eLCDIF write modes of operation before entering any particular mode.

Initialization steps:

1. Configure the external I/Os to correctly interface the external display.
2. Start the DISPLAY_CLK clock and set the appropriate frequency by programming the registers in CCM.

3. Start the BUS_CLK and set the appropriate frequency by programming the registers in CCM.
4. Bring the eLCDIF out of soft reset and disable the clock gate bit.
5. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller. See Behavior During Reset for more information on Reset requirements.
6. Make sure READ_WRITEB bit in HW_LCDIF_CTRL register is 0.
7. Select bus master mode by setting the LCDIF_MASTER bit in HW_LCDIF_CTRL register to 1. This is required when writing to the display.
8. Set the INPUT_DATA_SWIZZLE according to the endianness of the LCD controller. Also, set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to shift the data left or right before it is output.
9. Set the WORD_LENGTH field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24/32-bit input. Also, select the correct 16/18/24 bit data format with the corresponding fields in HW_LCDIF_CTRL register.
10. Set the BYTE_PACKING_FORMAT field in HW_LCDIF_CTRL1 according to the input frame.
11. Set the LCD_DATABUS_WIDTH appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24/32-bit output.
12. Enable the necessary IRQs.

37.2.5.2 MPU Read Mode

The following initialization steps should be done to enter the MPU read mode of operation:

Initialization steps:

1. Configure the external I/Os to correctly interface the external display.
2. Start the DISPLAY_CLK and set the appropriate frequency by programming the registers in CCM.
3. Start the BUS_CLK and set the appropriate frequency by programming the registers in CCM.
4. Bring the eLCDIF out of soft reset and clock gate.
5. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller.
6. Set the READ_WRITEB bit in LCDIF_CTRL register to 1.
7. Select the DMA mode by making the LCDIF_MASTER bit in LCDIF_CTRL register 0.
8. Also, set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to shift the data left or right before it is output.

9. Indicate if the read data needs to color-space-converted and stored in a different RGB format by setting the READ_MODE_OUTPUT_IN_RGB_FORMAT field accordingly.
10. Set the WORD_LENGTH field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24-bit input if READ_MODE_OUTPUT_IN_RGB_FORMAT is required. Also, select the correct 16/18/24 bit data format with the corresponding fields in LCDIF_CTRL register.
11. Set the READ_MODE_NUM_PACKED_SUBWORDS field in LCDIF_CTRL2 according to the number of subwords per word required to be packed.
12. Set the READ_PACK_DIR to 1 if it is required to store the data in big-endian format.
13. Set the LCD_DATABUS_WIDTH appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24-bit output.
14. Enable the necessary IRQs.

37.2.6 MPU Interface

The MPU interface is used to transfer data and commands between the internal buffer of LCD controller/display and the MPU or vice versa at relatively lower speeds. eLCDIF can support the 6800 as well as the 8080 MPU protocol. If DOTCLK_MODE, DVI_MODE and VSYNC_MODE bits in LCDIF_CTRL registers are 0, it implies that the block is in MPU interface mode of operation. The LCDIF MPU mode has four basic timing parameters: Setup and Hold for the Command/Data register selection (TCS, TCH) and Setup and Hold for the Data bus (TDS, TDH). These parameters are expressed in DISPLAY_CLK cycles. The LCD_WR signal is used as the write strobe while LCD_RS signal is typically used to switch between command and data modes.

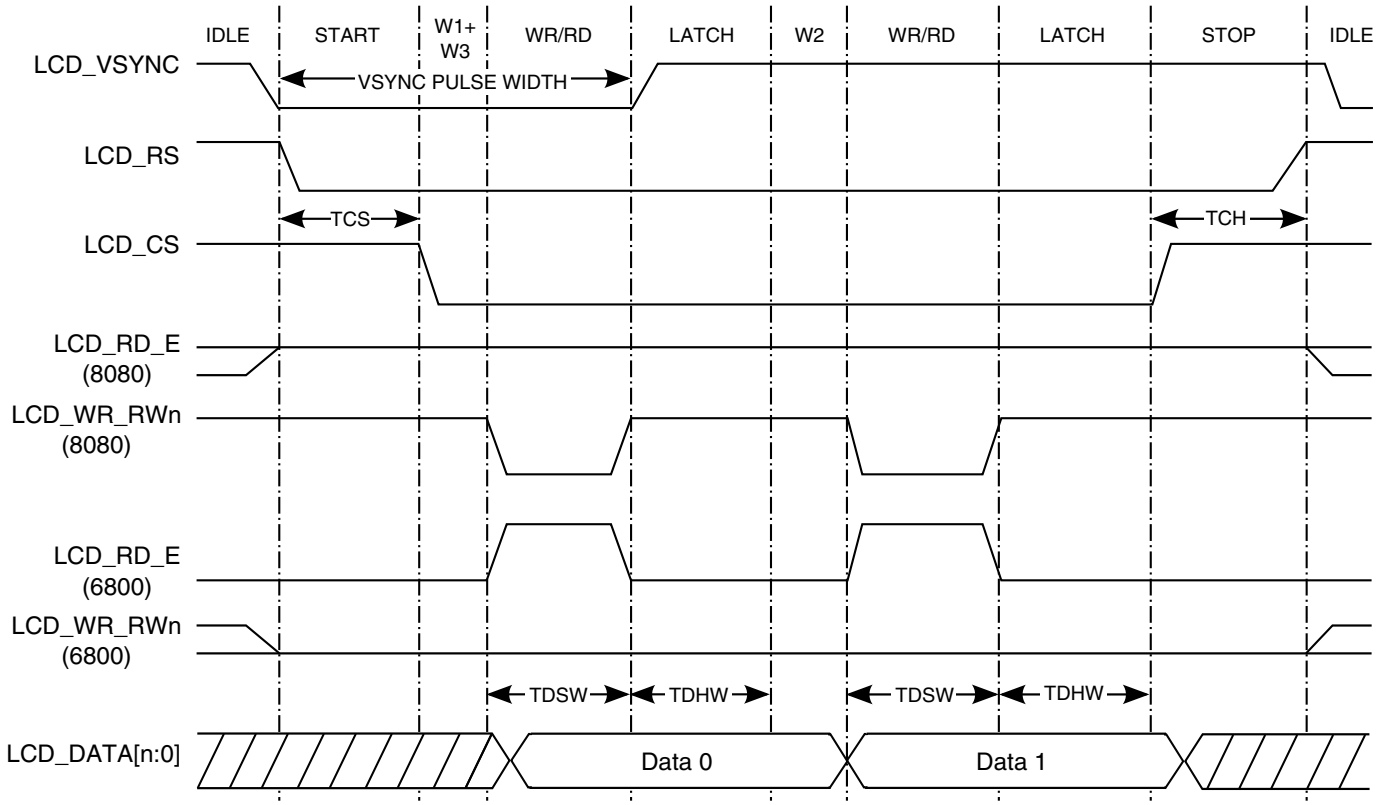


Figure 37-11. Timing in write mode of 6800 and 8080 protocols

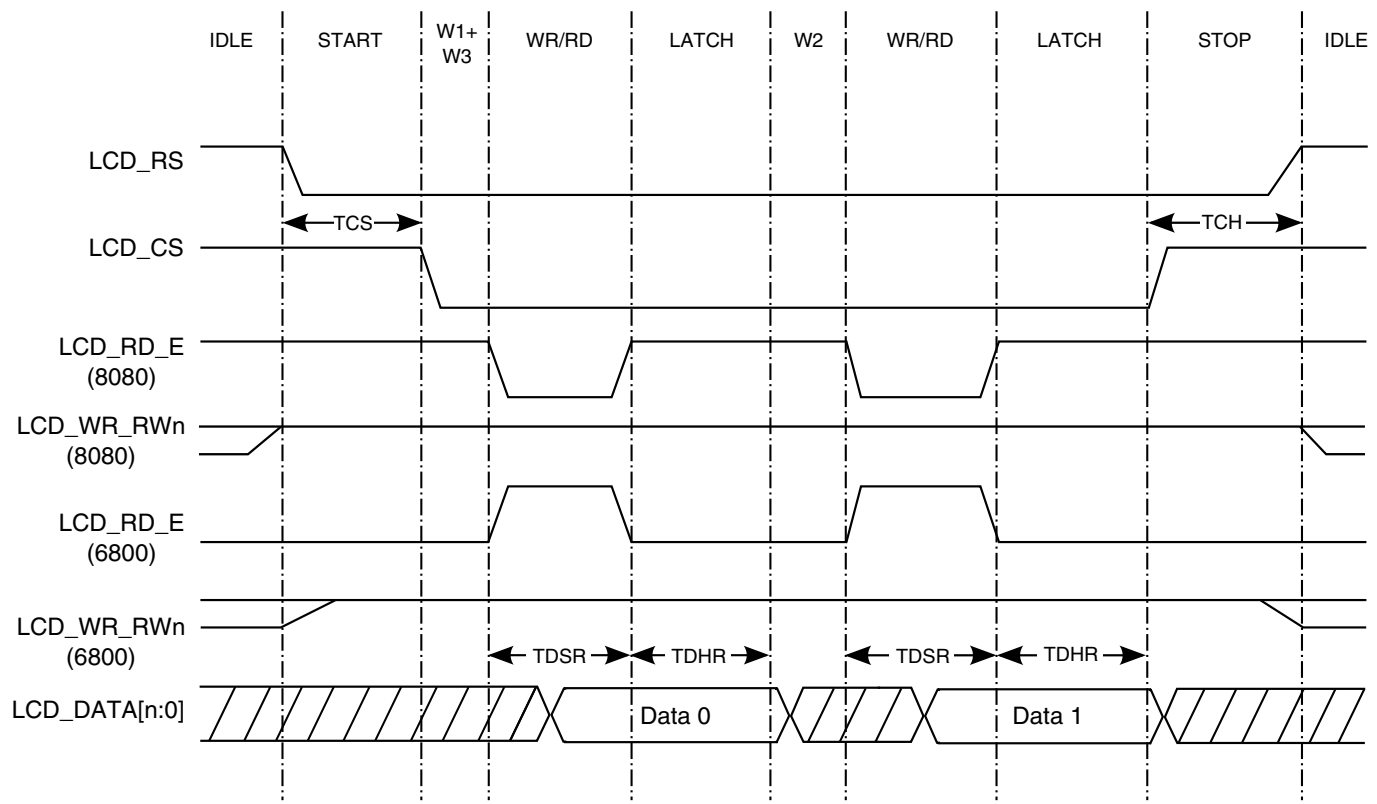


Figure 37-12. Read timing interface in 6800 and 8080 protocols

The eLCDIF has flexible pin and strobe timings which enable it to optimally support a wide range of LCDs. The minimum cycle time is two DISPLAY_CLK cycles (TDS=TDH=1). For example, this results in a maximum LCD data rate of 12 MB/s when DISPLAY_CLK is 24 MHz. TDS and TDH are 8-bit values, so the minimum eLCDIF period is 510 DISPLAY_CLK cycles (47 KHz with a 24 MHz DISPLAY_CLK). The timings are not automatically adjusted if the DISPLAY_CLK frequency changes, so it may be necessary to adjust the timings if DISPLAY_CLK changes.

In the MPU interface mode, the LCDIF_CTRL_ BYPASS_COUNT bit must be 0. The RUN bit is cleared automatically once the eLCDIF has received/transmitted all the data as per the LCDIF_TRANSFER_COUNT register and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.

37.2.6.1 Code Example to Initialize the eLCDIF in MPU Write Mode

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1(LCDIF_CTRL, DATA_SELECT, 1); // 0 if sending command, 1 if sending data. Note that the
                                    // idle state for LCD_RS signal is high, regardless of the
                                     // programming of the DATA_SELECT register.
BF_CS1 (LCDIF_CTRL, MODE86, 8080_MODE);
BF_CS1 (LCDIF_CTRL, READ_WRITEB, 0);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 0); //Must be 0 in MPU mode
BF_CS1 (LCDIF_CTRL1, BUSY_ENABLE, 1);//Only if LCD controller implements a busy line
BF_CS4 (LCDIF_TIMING, CMD_HOLD, 2, CMD_SETUP, 2, DATA_HOLD, 2, DATA_SETUP, 2);  //Values
based
                               // on DISPLAY_CLK frequency and timing requirements of
controller.
                              // Note that these register must be non-zero for correct
operation.
BF_CS2 (LCDIF_TRANSFER_COUNT, H_COUNT, 320, V_COUNT, 240); //For a 320 RGB x 240 display
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

The eLCDIF is now ready to receive data through DMA writes to the LCDIF_DATA register or fetch data directly from memory as a bus master. Also, note that, while in soft DMA mode, the software will need to poll the FIFO STATUS bits to ensure that it does not overflow the eLCDIF data buffers. When eLCDIF is done transmitting H_COUNT x V_COUNT pixels, it will stop, turn off the RUN bit and assert the cur_frame_done interrupt.

37.2.7 VSYNC Interface

The VSYNC interface uses the same protocol as the MPU interface, with an additional signal VSYNC at the frame rate of the display, as shown in the figure given in MPU Interface section. It is used in the moving picture display mode where data has to be written to the internal LCD buffer at a speed higher than the display rate and displayed in synchronization with the VSYNC signal. This mode is selected by setting the

VSYNC_MODE bit in LCDIF_CTRL register. The VSYNC signal is programmable for period, polarity and direction. Many other programmable parameters are shared with the MPU interface. The VSYNC_OEB bit in LCDIF_VDCTRL0 register indicates whether the display controller will send the VSYNC signal, or whether it should be generated by eLCDIF. The timing of the VSYNC signal is based on the DISPLAY_CLK (make sure VSYNC_PULSE_WIDTH_UNIT = VSYNC_PERIOD_UNIT = 0 and VSYNC_ONLY = 1) and it is determined by the VSYNC_PERIOD, VSYNC_PULSE_WIDTH and VSYNC_POL fields in LCDIF_VDCTRL0-4 registers. The SYNC_SIGNALS_ON bit in LCDIF_VDCTRL4 register must be set if the target requires the VSYNC signal to be generated by eLCDIF. If the WAIT_FOR_VSYNC_EDGE bit in LCDIF_CTRL register is set, it indicates that the hardware should wait until it sees the leading VSYNC edge before starting the data transfer. The VERITCAL_WAIT_CNT indicates the number of DISPLAY_CLK cycles from the leading VSYNC edge after which data transfer will be started on the interface.

In the VSYNC interface mode, the LCDIF_CTRL_ BYPASS_COUNT bit must be 0. The RUN bit is cleared automatically once the eLCDIF has received/transmitted all the data as per the LCDIF_TRANSFER_COUNT register and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.

37.2.7.1 Code Example to Initialize eLCDIF in VSYNC Mode

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DATA_SELECT, 1); // 0 if sending command, 1 if sending data. Note that
//the idle state for LCD_RS signal is high, regardless of the programming of the DATA_SELECT
//register.

BF_CS1 (LCDIF_CTRL, MODE86, 8080_MODE);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 0); //Must be 0 in MPU mode
BF_CS1 (LCDIF_CTRL1, BUSY_ENABLE, 0);
BF_CS4 (LCDIF_TIMING, CMD_HOLD, 2, CMD_SETUP, 2, DATA_HOLD, 2, DATA_SETUP, 2);  //Values
//based on DISPLAY_CLK frequency and timing requirements of controller. Note that these
//register must be non-zero for the MPU and VSYNC modes.
BF_CS2 (LCDIF_TRANSFER_COUNT, H_COUNT, 320, V_COUNT, 240);//For a 320 RGB x 240 display
//The following section indicates setting up the VSYNC signal timing when VSYNC is an output
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Making VSYNC signal an output
BF_CS1 (LCDIF_VDCTRL4, VSYNC_ONLY, 1); //Only need to generate VSYNC signal
BF_CS1 (VDCTRL0, VSYNC_POL, 0); //Setting the polarity of VSYNC signal to be low during
//VSYNC_PULSE_WIDTH time
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 0, VSYNC_PULSE_WIDTH_UNIT, 0);
BF_CS2 (LCDIF_VDCTRL1, VSYNC_PERIOD, 400000, VSYNC_PULSE_WIDTH, 100);//Frame display rate in
//terms of number of DISPLAY_CLKs.
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 0, HSYNC_PERIOD, 0);
BF_CS1 (LCDIF_VDCTRL3, VERTICAL_WAIT_CNT, 50);
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
BF_CS2 (LCDIF_CTRL, VSYNC_MODE, 1, WAIT_FOR_VSYNC_EDGE, 1); //set WAIT_FOR_VSYNC_EDGE if
//software wishes to transfer the next frame after the VSYNC edge occurs.
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

The eLCDIF is now ready to receive data through DMA writes to the LCDIF_DATA register or fetch data directly from memory as a bus master. When eLCDIF is done transmitting H_COUNT x V_COUNT pixels, it will stop, turn off the RUN bit and assert the cur_frame_done interrupt.

37.2.8 DOTCLK Interface

The DOTCLK interface is another mode used in moving picture displays. It includes the VSYNC, HSYNC, DOTCLK and (optional) ENABLE signals. The interface is popularly called the RGB interface if the ENABLE signal is present.

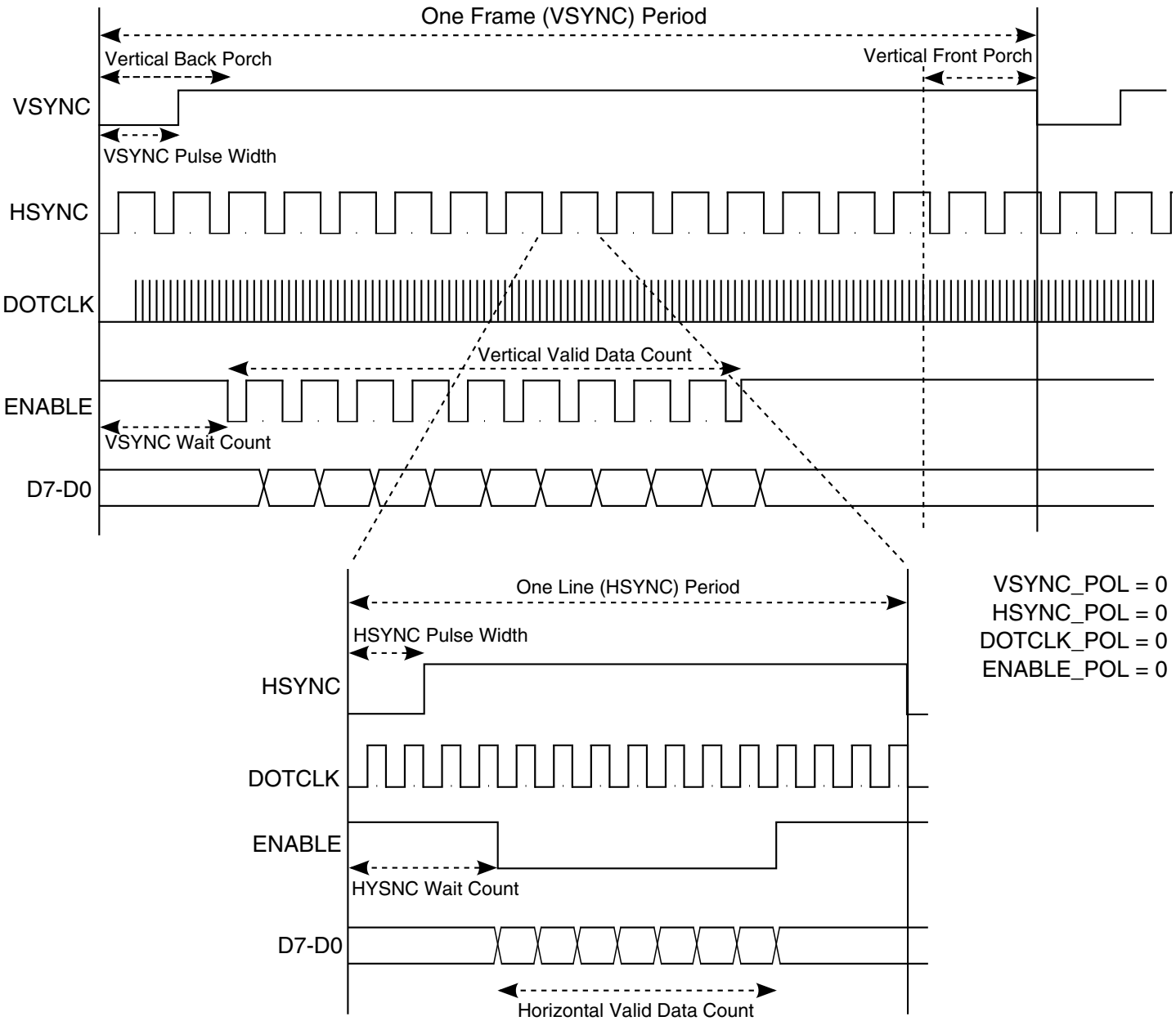


Figure 37-13. DOTCLK protocol with programmable parameters

The DOTCLK mode writes data at high speed to the LCD, and the display operation is synchronized with the VSYNC, HSYNC, ENABLE and DOTCLK signals. The polarities, periods and pulse-widths of the sync signals are programmable using the LCDIF_VDCTRL0-4 registers. The units for the VSYNC signal must be number of horizontal lines and can be selected using the VSYNC_PULSE_WIDTH_UNIT and VSYNC_PERIOD_UNIT bit fields. The VERTICAL_WAIT_CNT is by default given the same unit as the VSYNC_PERIOD. The DISPLAY_CLK frequency is managed by the CCM.

In DOTCLK mode, LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DOTCLK_MODE bit 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and issue the cur_frame_done interrupt.

37.2.8.1 Code Example

The following code shows an example for programming a 320x240 display. Note that setting up the display must be done through the MPU mode or via SPI.

```
// Note: Common initialization steps in Initializing the eLCDIF must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DOTCLK_MODE, 1);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 1); //Always for DOTCLK mode
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Vsync is always an output in the DOTCLK mode
BF_CS4 (LCDIF_VDCTRL0, VSYNC_POL, 0, HSYNC_POL, 0, DOTCLK_POL, 0, ENABLE_POL, 0);
BF_CS1 (LCDIF_VDCTRL0, ENABLE_PRESENT, 1);
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 1, VSYNC_PULSE_WIDTH_UNIT, 1);
BF_CS1 (LCDIF_VDCTRL0, VSYNC_PULSE_WIDTH, 2);
BF_CS1 (LCDIF_VDCTRL1, VSYNC_PERIOD, 280);
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 10, HSYNC_PERIOD, 360); //Assuming
                                                    //  LCD_DATABUS_WIDTH is 24bit
BF_CS2 (LCDIF_VDCTRL3, VSYNC_ONLY, 0);
BF_CS2 (LCDIF_VDCTRL3, HORIZONTAL_WAIT_CNT, 20, VERTICAL_WAIT_CNT, 20);
BF_CS1 (LCDIF_VDCTRL4, DOTCLK_H_VALID_DATA_CNT, 320);//Note that DOTCLK_V_VALID_DATA_CNT is
                              //implicitly assumed to be HW_LCDIF_TRANSFER_COUNT_V_COUNT
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

To stop the transfer completely, the ideal way is to make DOTCLK_MODE = 0. In that case, the block will transmit whatever it had in its FIFO, turn off the RUN bit and toggle the dma_end_cmd signal indicating to the DMA that it is done with the transfer.

37.2.9 ITU-R BT.656 Digital Video Interface (DVI)

ITU-R BT.656 Digital Video Interface shown below transmits 4:2:2 YCbCr digital component video to a digital video encoder that can translate it into 525/60 or 625/50 analog TV signal. Unique timing codes (timing reference signals) are embedded within the video stream to indicate the different timing events that would have been otherwise indicated by VSYNC, HSYNC and BLANK signals. The hardware supports 8-bit data transfers; the pins are shared with the lower 8 bits of LCD data bus. The LCD_RS pin is shared with the clock signal of the interface (called CCIRCLK here for uniqueness). CCIRCLK also can be obtained on the LCD_DOTCK pin. The mode shares the write FIFO with the LCD interface and the associated pipeline. The programmable parameters in registers LCDIF_DVICTRL0-3 allow setting the total number of horizontal lines per frame, vertical and horizontal blanking interval, odd and even field start and end

positions, and so on. In short, these parameters are provided to ensure that the hardware has enough flexibility to generate the right 525/60 or 625/50 data streams. Most of the initialization steps in Initializing the eLCDIF such as data shifting, swizzle, and so on, are applicable to DVI mode also. The register descriptions in the programmable registers section at the end of this chapter include example code for programming the DVICTRL0-3 registers.

In DVI mode, LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DVI_MODE bit the value 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and assert the cur_frame_done interrupt.

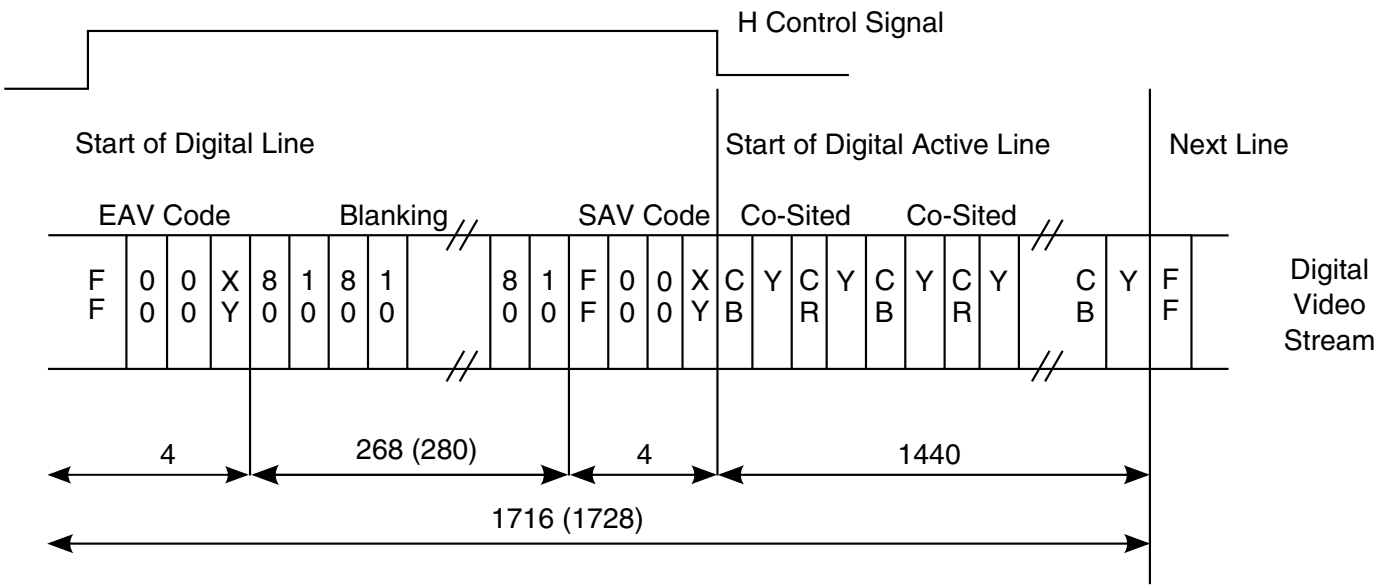


Figure 37-14. Digital Video Interface

37.2.10 eLCDIF Pin Usage by Interface Mode

Table 37-1 and Table 37-2 indicates how the eLCDIF level interface pins are used based on the desired mode of operation. The chip level I/Os should also be configured to be consistent with the desired eLCDIF operating mode.

The VSYNC signal has been mapped onto two pins, LCD_BUSY and LCD_VSYNC. The pin multiplexing can be programmed to select either of those pins to function as VSYN.

NOTE

There is an option to internally mux the HSYNC, DOTCLK and ENABLE signals in the DOTCLK mode by setting the

MUX_SYNC_SIGNALS bit in the VDCTRL0 register. There is also an option to internally mux the LCD_WR_RWn and LCD_RD_E pins in the CTRL1 register for backward compatibility.

Table 37-1. Pin use in MPU Mode and VSYNC Mode

PIN NAME	8-bit MPU LCD IF	16-bit MPU LCD IF	18-bit MPU LCD IF	24-bit MPU LCD IF	8-bit VSYNC LCD IF	16-bit VSYNC LCD IF	18-bit VSYNC LCD IF	24-bit VSYNC LCD IF
LCD_RS	LCD_ RS	LCD_ RS	LCD_ RS	LCD_RS	LCD_RS	LCD_ RS	LCD_RS	LCD_RS
LCD_CS	LCD_CS	LCD_CS	LCD_ CS	LCD_CS	LCD_CS	LCD_CS	LCD_ CS	LCD_CS
LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn
LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E
LCD_VSYNC* (Two options)	X	X	X	X	LCD_ VSYNC	LCD_ VSYNC	LCD_ VSYNC	LCD_ VSYNC
LCD_HSYNC	X	X	X	X	X	X	X	X
LCD_DOTCLK	X	X	X	X	X	X	X	X
LCD_ENABLE	X	X	X	X	X	X	X	X
LCD_D23	X	X	X	LCD_D23	X	X	X	LCD_D23
LCD_D22	X	X	X	LCD_D22	X	X	X	LCD_D22
LCD_D21	X	X	X	LCD_D21	X	X	X	LCD_D21
LCD_D20	X	X	X	LCD_D20	X	X	X	LCD_D20
LCD_D19	X	X	X	LCD_D19	X	X	X	LCD_D19
LCD_D18	X	X	X	LCD_D18	X	X	X	LCD_D18
LCD_D17	X	X	LCD_D17	LCD_D17	X	X	LCD_D17	LCD_D17
LCD_D16	X	X	LCD_D16	LCD_D16	X	X	LCD_D16	LCD_D16
LCD_D15 / VSYNC*	X	LCD_D15	LCD_D15	LCD_D15	VSYNC (optional)	LCD_D15	VSYNC (optional)	LCD_D15
LCD_D14 / HSYNC**	X	LCD_D14	LCD_D14	LCD_D14	X	LCD_D14	X	LCD_D14
LCD_D13 / LCD_DOTCLK**	X	LCD_D13	LCD_D13	LCD_D13	X	LCD_D13	X	LCD_D13

Table continues on the next page...

Table 37-1. Pin use in MPU Mode and VSYNC Mode (continued)

PIN NAME	8-bit MPU LCD IF	16-bit MPU LCD IF	18-bit MPU LCD IF	24-bit MPU LCD IF	8-bit VSYNC LCD IF	16-bit VSYNC LCD IF	18-bit VSYNC LCD IF	24-bit VSYNC LCD IF
LCD_D12 / ENABLE**	X	LCD_D12	LCD_D12	LCD_D12	X	LCD_D12	X	LCD_D12
LCD_D11	X	LCD_D11	LCD_D11	LCD_D11	X	LCD_D11	X	LCD_D11
LCD_D10	X	LCD_D10	LCD_D10	LCD_D10	X	LCD_D10	X	LCD_D10
LCD_D9	X	LCD_D9	LCD_D9	LCD_D9	X	LCD_D9	X	LCD_D9
LCD_D8	X	LCD_D8	LCD_D8	LCD_D8	X	LCD_D8	X	LCD_D8
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET
LCD_BUSY / LCD_VSYNC	LCD_BUSY	LCD_BUSY	LCD_BUSY	LCD_BUSY	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)

Table 37-2. Pin use in DOTCLK Mode and DVI Mode

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
LCD_RS	X	X	X	X	CCIR_CLK
LCD_CS	X	X	X	X	X
LCD_WR_RWn	X	X	X	X	X
LCD_RD_E	X	X	X	X	X
LCD_VSYNC* (Two options)	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	X
LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	X
LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	X
LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	X

Table continues on the next page...

Table 37-2. Pin use in DOTCLK Mode and DVI Mode (continued)

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
LCD_D23	X	X	X	LCD_D23	X
LCD_D22	X	X	X	LCD_D22	X
LCD_D21	X	X	X	LCD_D21	X
LCD_D20	X	X	X	LCD_D20	X
LCD_D19	X	X	X	LCD_D19	X
LCD_D18	X	X	X	LCD_D18	X
LCD_D17	X	X	LCD_D17	LCD_D17	X
LCD_D16	X	X	LCD_D16	LCD_D16	X
LCD_D15/ VSYNC*	X	LCD_D15	LCD_D15	LCD_D15	X
LCD_D14 / HSYNC**	X	LCD_D14	LCD_D14	LCD_D14	X
LCD_D13 / LCD_DOTCLK**	X	LCD_D13	LCD_D13	LCD_D13	X
LCD_D12 / ENABLE**	X	LCD_D12	LCD_D12	LCD_D12	X
LCD_D11	X	LCD_D11	LCD_D11	LCD_D11	X
LCD_D10	X	LCD_D10	LCD_D10	LCD_D10	X
LCD_D9	X	LCD_D9	LCD_D9	LCD_D9	X
LCD_D8	X	LCD_D8	LCD_D8	LCD_D8	X
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	X
LCD_BUSY / LCD_VSYNC	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	X

37.3 Behavior During Reset

BUS_CLK and DISPLAY_CLK must be running before making any changes to SFTRST or CLKGATE bits. A soft reset (SFTRST) can take multiple clock periods to complete, so do not set CLKGATE when setting SFTRST. The reset process gates the clocks automatically.

37.4 Programmable Registers

eLCDIF Hardware Register Format Summary

LCDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_A000	eLCDIF General Control Register (LCDIF_CTRL)	32	R/W	C000_0000h	37.4.1/ 2280
4100_A004	eLCDIF General Control Register (LCDIF_CTRL_SET)	32	R/W	C000_0000h	37.4.1/ 2280
4100_A008	eLCDIF General Control Register (LCDIF_CTRL_CLR)	32	R/W	C000_0000h	37.4.1/ 2280
4100_A00C	eLCDIF General Control Register (LCDIF_CTRL_TOG)	32	R/W	C000_0000h	37.4.1/ 2280
4100_A010	eLCDIF General Control1 Register (LCDIF_CTRL1)	32	R/W	000F_0000h	37.4.2/ 2283
4100_A014	eLCDIF General Control1 Register (LCDIF_CTRL1_SET)	32	R/W	000F_0000h	37.4.2/ 2283
4100_A018	eLCDIF General Control1 Register (LCDIF_CTRL1_CLR)	32	R/W	000F_0000h	37.4.2/ 2283
4100_A01C	eLCDIF General Control1 Register (LCDIF_CTRL1_TOG)	32	R/W	000F_0000h	37.4.2/ 2283
4100_A020	eLCDIF General Control2 Register (LCDIF_CTRL2)	32	R/W	0020_0000h	37.4.3/ 2286
4100_A024	eLCDIF General Control2 Register (LCDIF_CTRL2_SET)	32	R/W	0020_0000h	37.4.3/ 2286
4100_A028	eLCDIF General Control2 Register (LCDIF_CTRL2_CLR)	32	R/W	0020_0000h	37.4.3/ 2286
4100_A02C	eLCDIF General Control2 Register (LCDIF_CTRL2_TOG)	32	R/W	0020_0000h	37.4.3/ 2286

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_A030	eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIF_TRANSFER_COUNT)	32	R/W	0001_0000h	37.4.4/ 2288
4100_A040	LCD Interface Current Buffer Address Register (LCDIF_CUR_BUF)	32	R/W	0000_0000h	37.4.5/ 2288
4100_A050	LCD Interface Next Buffer Address Register (LCDIF_NEXT_BUF)	32	R/W	0000_0000h	37.4.6/ 2289
4100_A060	LCD Interface Timing Register (LCDIF_TIMING)	32	R/W	0000_0000h	37.4.7/ 2290
4100_A070	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0)	32	R/W	0000_0000h	37.4.8/ 2290
4100_A074	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_SET)	32	R/W	0000_0000h	37.4.8/ 2290
4100_A078	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_CLR)	32	R/W	0000_0000h	37.4.8/ 2290
4100_A07C	eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_TOG)	32	R/W	0000_0000h	37.4.8/ 2290
4100_A080	eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF_VDCTRL1)	32	R/W	0000_0000h	37.4.9/ 2292
4100_A090	LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF_VDCTRL2)	32	R/W	0000_0000h	37.4.10/ 2292
4100_A0A0	eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF_VDCTRL3)	32	R/W	0000_0000h	37.4.11/ 2293
4100_A0B0	eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF_VDCTRL4)	32	R/W	0000_0000h	37.4.12/ 2294
4100_A0C0	Digital Video Interface Control0 Register (LCDIF_DVICTRL0)	32	R/W	0000_0000h	37.4.13/ 2295
4100_A0D0	Digital Video Interface Control1 Register (LCDIF_DVICTRL1)	32	R/W	0000_0000h	37.4.14/ 2296
4100_A0E0	Digital Video Interface Control2 Register (LCDIF_DVICTRL2)	32	R/W	0000_0000h	37.4.15/ 2297
4100_A0F0	Digital Video Interface Control3 Register (LCDIF_DVICTRL3)	32	R/W	0000_0000h	37.4.16/ 2298
4100_A100	Digital Video Interface Control4 Register (LCDIF_DVICTRL4)	32	R/W	0000_0000h	37.4.17/ 2299
4100_A110	RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIF_CSC_COEFF0)	32	R/W	0000_0000h	37.4.18/ 2300
4100_A120	RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIF_CSC_COEFF1)	32	R/W	0000_0000h	37.4.19/ 2301
4100_A130	RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIF_CSC_COEFF2)	32	R/W	0000_0000h	37.4.20/ 2302

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_A140	RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIF_CSC_COEFF3)	32	R/W	0000_0000h	37.4.21/ 2303
4100_A150	RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIF_CSC_COEFF4)	32	R/W	0000_0000h	37.4.22/ 2303
4100_A160	RGB to YCbCr 4:2:2 CSC Offset Register (LCDIF_CSC_OFFSET)	32	R/W	0080_0010h	37.4.23/ 2304
4100_A170	RGB to YCbCr 4:2:2 CSC Limit Register (LCDIF_CSC_LIMIT)	32	R/W	00FF_00FFh	37.4.24/ 2305
4100_A180	LCD Interface Data Register (LCDIF_DATA)	32	R/W	0000_0000h	37.4.25/ 2306
4100_A190	Bus Master Error Status Register (LCDIF_BM_ERROR_STAT)	32	R/W	0000_0000h	37.4.26/ 2306
4100_A1A0	CRC Status Register (LCDIF_CRC_STAT)	32	R/W	0000_0000h	37.4.27/ 2307
4100_A1B0	LCD Interface Status Register (LCDIF_STAT)	32	R	9500_0000h	37.4.28/ 2307
4100_A1C0	LCD Interface Version Register (LCDIF_VERSION)	32	R	0400_0000h	37.4.29/ 2309
4100_A1D0	LCD Interface Debug0 Register (LCDIF_DEBUG0)	32	R	0E81_0000h	37.4.30/ 2309
4100_A1E0	LCD Interface Debug1 Register (LCDIF_DEBUG1)	32	R	0000_0000h	37.4.31/ 2311
4100_A1F0	LCD Interface Debug2 Register (LCDIF_DEBUG2)	32	R	0000_0000h	37.4.32/ 2312
4100_A200	eLCDIF Threshold Register (LCDIF_THRES)	32	R/W	0100_000Fh	37.4.33/ 2312

37.4.1 eLCDIF General Control Register (LCDIF_CTRL*n*)

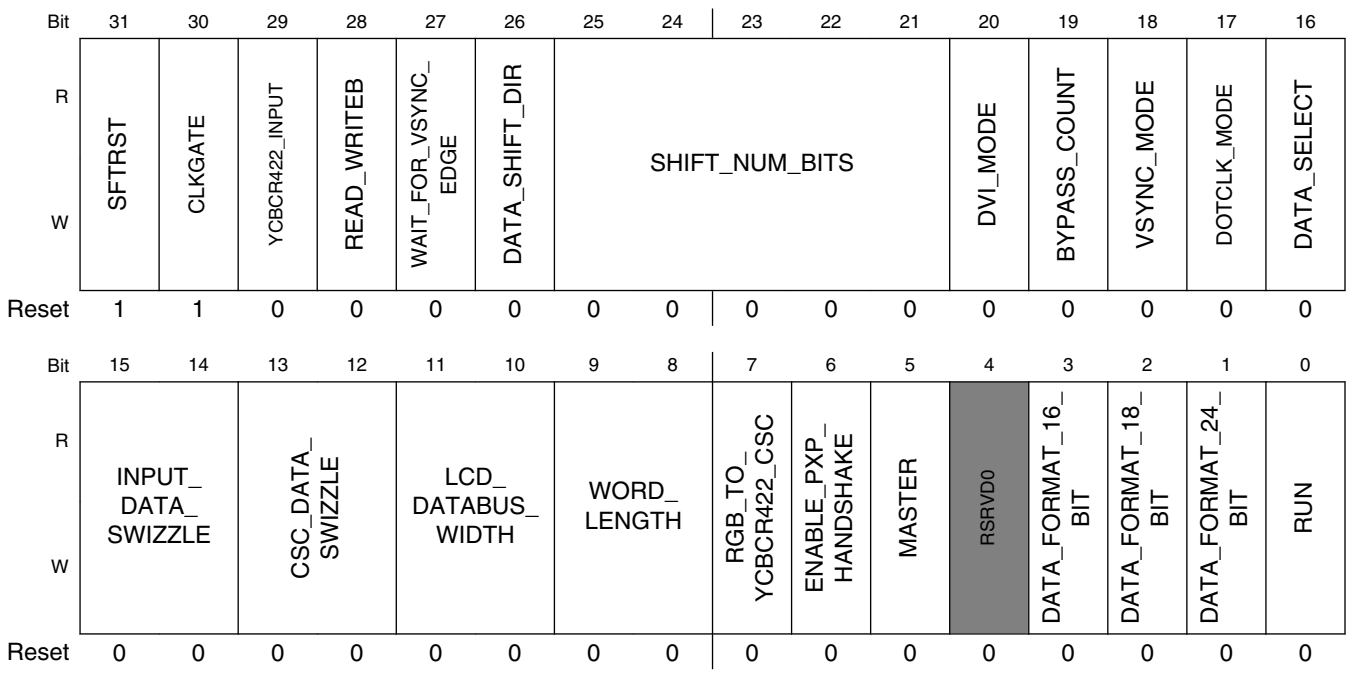
The LCD Interface Control Register provides overall control of the eLCDIF block. The eLCDIF Control Register provides a variety of control functions to the programmer. These functions allow the interface to be very flexible to work with a variety of LCD controllers, and to minimize overhead and increase performance of LCD programming. The register has been organized such that switching between the different LCD modes can be done with minimum PIO writes.

Addresses: LCDIF_CTRL is 4100_A000h base + 0h offset = 4100_A000h

LCDIF_CTRL_SET is 4100_A000h base + 4h offset = 4100_A004h

LCDIF_CTRL_CLR is 4100_A000h base + 8h offset = 4100_A008h

LCDIF_CTRL_TOG is 4100_A000h base + Ch offset = 4100_A00Ch



LCDIF_CTRL*n* field descriptions

Field	Description
31 SFTRST	This bit must be set to zero to enable normal operation of the eLCDIF. When set to one, it forces a block level reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 YCBCR422_INPUT	Zero implies input data is in RGB color space. One implies input data is in YCbCr 4:2:2 format, such that YCbYCr are packed in a 32-bit word. It also means that there are 2 pixels in 4 bytes. If this bit is set, software should program the H_COUNT field in the TRANSFER_COUNT register to the total number of pixels that will have to be fetched by the eLCDIF block per line and the BYTE_PACKING_FORMAT should be 0xF. The WORD_LENGTH does not matter in this case.

Table continues on the next page...

LCDIF_CTRL*n* field descriptions (continued)

Field	Description
28 READ_WRITEB	By default, eLCDIF is in the write mode. Setting this bit to 1 will make the hardware go into 6800/8080 MPU read mode. Make sure that DMA operation is selected (LCDIF_MASTER bit is 0) when performing read operations.
27 WAIT_FOR_ VSYNC_EDGE	Setting this bit to 1 will make the hardware wait for the triggering VSYNC edge before starting write transfers to the LCD. Used only in the VSYNC mode of operation.
26 DATA_SHIFT_ DIR	Use this bit to determine the direction of shift of transmit data. In the DVI mode, it works only on the active data, not on the timing codes and ancillary data. 0x0 **TXDATA_SHIFT_LEFT** — Data to be transmitted is shifted LEFT by SHIFT_NUM_BITS bits. 0x1 **TXDATA_SHIFT_RIGHT** — Data to be transmitted is shifted RIGHT by SHIFT_NUM_BITS bits.
25–21 SHIFT_NUM_ BITS	The data to be transmitted is shifted left or right by this number of bits.
20 DVI_MODE	Set this bit to 1 to get into theITU-R BT.656 digital video interface mode. Toggle this bit from 1 to 0 to make the hardware go out of DVI mode after completing all data transfer, deasserting the RUN bit and toggling the dma_end_cmd signal.
19 BYPASS_ COUNT	When this bit is 0, it means that eLCDIF will stop the block operation and turn off the RUN bit after the amount of data indicated by the LCDIF_TRANSFER_COUNT register has been transferred out. When this bit is set to 1, the block will continue normal operation indefinitely until it is told to stop. This bit must be 0 in MPU and VSYNC modes, and must be 1 in DOTCLK and DVI modes of operation.
18 VSYNC_MODE	Setting this bit to 1 will make the eLCDIF hardware go into VSYNC mode. WAIT_FOR_VSYNC_EDGE can be used only if this bit is set. If VSYNC signal is required to be an output from the block, SYNC_SIGNALS_ON bit in LCDIF_VDCTRL4 register must be set.
17 DOTCLK_MODE	Set this bit to 1 to make the hardware go into the DOTCLK mode, i.e. VSYNC/HSYNC/DOTCLK/ENABLE interface mode. ENABLE is optional, selected by the ENABLE_PRESENT bit. Toggle this bit from 1 to 0 to make the hardware go out of DOTCLK mode after completing all data transfer and deasserting the RUN bit.
16 DATA_SELECT	Command Mode polarity bit. This bit should only be changed when RUN is 0. 0x0 **CMD_MODE** — Command Mode. DCn signal is Low. 0x1 **DATA_MODE** — Data Mode. DCn signal is High.
15–14 INPUT_DATA_ SWIZZLE	This field specifies how to swap the bytes fetched by the bus master interface. The swizzle function is independent of the WORD_LENGTH bit. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
13–12 CSC_DATA_ SWIZZLE	This field specifies how to swap the bytes after the data has been converted into an internal representation of 24 bits per pixel and before it is transmitted over the LCD interface bus. The data is always transmitted with the least significant byte/hword (half word) first after the swizzle takes place. So, INPUT_DATA_SWIZZLE takes place first on the incoming data, and then CSC_DATA_SWIZZLE is applied. The swizzle function is independent of the WORD_LENGTH or the LCD_DATABUS_WIDTH fields. If RGB_TO_YCRCB422_CSC bit is set, the swizzle occurs on the Y, Cb, Cr values. The supported swizzle configurations are:

Table continues on the next page...

LCDIF_CTRL*n* field descriptions (continued)

Field	Description
	0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
11–10 LCD_ DATABUS_ WIDTH	LCD Data bus transfer width. 0x0 **16_BIT** — 16-bit data bus mode. 0x1 **8_BIT** — 8-bit data bus mode. 0x2 **18_BIT** — 18-bit data bus mode. 0x3 **24_BIT** — 24-bit data bus mode.
9–8 WORD_ LENGTH	Input data format. 0x0 **16_BIT** — Input data is 16 bits per pixel. 0x1 **8_BIT** — Input data is 8 bits wide. 0x2 **18_BIT** — Input data is 18 bits per pixel. 0x3 **24_BIT** — Input data is 24 bits per pixel.
7 RGB_TO_ YCBCR422_ CSC	Set this bit to 1 to enable conversion from RGB to YCbCr colorspace. See the LCDIF_CSC_ registers for further details.
6 ENABLE_PXP_ HANDSHAKE	If this bit is set and LCDIF_MASTER bit is set, the eLCDIF will act as bus master and the handshake mechanism between eLCDIF and ePXP will be turned on. If LCDIF_MASTER bit is not set, this bit becomes a don't care.
5 MASTER	Set this bit to make the eLCDIF act as a bus master. If this bit is reset, the eLCDIF will act in its traditional DMA slave mode.
4 RSRVD0	Reserved bits. Write as 0.
3 DATA_ FORMAT_16_ BIT	When this bit is 1 and WORD_LENGTH = 0, it implies that the the 16-bit data is in ARGB555 format. When this bit is 0 and WORD_LENGTH = 0, it implies that the 16-bit data is in RGB565 format. When WORD_LENGTH is not 0, this bit is a dont care.
2 DATA_ FORMAT_18_ BIT	Used only when WORD_LENGTH = 2, i.e. 18-bit. 0x0 **LOWER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that lower 18 bits contain RGB 666 and upper 14 bits do not contain any useful data. 0x1 **UPPER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that upper 18 bits contain RGB 666 and lower 14 bits do not contain any useful data.
1 DATA_ FORMAT_24_ BIT	Used only when WORD_LENGTH = 3, i.e. 24-bit. Note that this applies to both packed and unpacked 24-bit data. 0x0 **ALL_24_BITS_VALID** — Data input to the block is in 24 bpp format, such that all RGB 888 data is contained in 24 bits. 0x1 **DROP_UPPER_2_BITS_PER_BYTE** — Data input to the block is actually RGB 18 bpp, but there is 1 color per byte, hence the upper 2 bits in each byte do not contain any useful data, and should be dropped.

Table continues on the next page...

LCDIF_CTRL*n* field descriptions (continued)

Field	Description
0 RUN	When this bit is set by software, the eLCDIF will start fetching data in either the DMA mode or the bus master mode and sending it across the interface. This bit must remain set for all the time the block is in operation.

37.4.2 eLCDIF General Control1 Register (LCDIF_CTRL1*n*)

The eLCDIF Control Register provides overall control of the eLCDIF block.

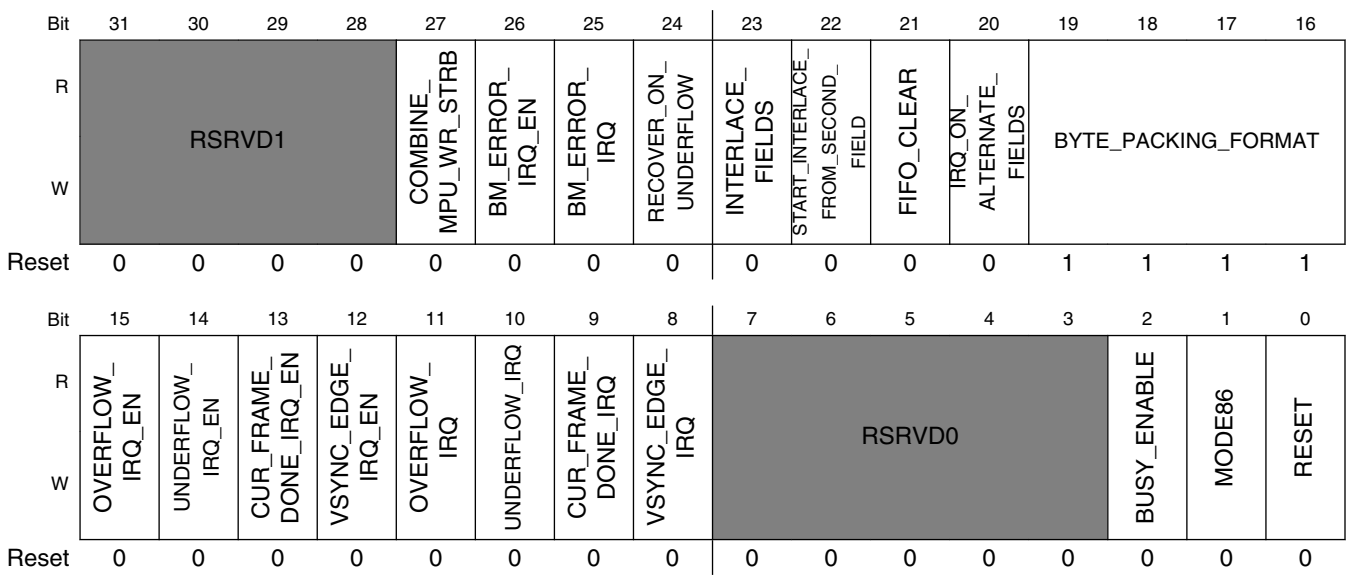
The eLCDIF Control1 Register provides additional programming to the eLCDIF. It implements some bits which are unlikely to change often in a particular application. It also carries interrupt-related bits which are common across more than one mode of operation.

Addresses: LCDIF_CTRL1 is 4100_A000h base + 10h offset = 4100_A010h

LCDIF_CTRL1_SET is 4100_A000h base + 14h offset = 4100_A014h

LCDIF_CTRL1_CLR is 4100_A000h base + 18h offset = 4100_A018h

LCDIF_CTRL1_TOG is 4100_A000h base + 1Ch offset = 4100_A01Ch



LCDIF_CTRL1*n* field descriptions

Field	Description
31–28 RSRVD1	Reserved bits. Write as 0.
27 COMBINE_ MPU_WR_STRB	If this bit is not set, the write strobe will be driven on LCD_WR_RWn pin in the 8080 mode and on the LCD_RD_E pin in the 6800 mode. If it is set, the write strobe of both the 6800 and 8080 modes will be driven only on the LCD_WR_RWn pin. Note that this does not work for read strobe.

Table continues on the next page...

LCDIF_CTRL1*n* field descriptions (continued)

Field	Description
26 BM_ERROR_ IRQ_EN	This bit is set to enable bus master error interrupt in the eLCDIF master mode.
25 BM_ERROR_ IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. This bit will be set when the eLCDIF is in master mode and an error response was returned by the slave. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
24 RECOVER_ON_ UNDERFLOW	Set this bit to enable the eLCDIF block to recover in the next field/frame if there was an underflow in the current field/frame.
23 INTERLACE_ FIELDS	Set this bit if it is required that the eLCDIF block fetches odd lines in one field and even lines in the other field. It will work only in LCDIF_MASTER is set to 1.
22 START_ INTERLACE_ FROM_ SECOND_FIELD	The default is to grab the odd lines first and then the even lines. Set this bit if it is required to grab the even lines first and then the odd lines. (Line numbers start from 1, so odd lines are 1,3,5,etc. and even lines are 2,4,6, etc.)
21 FIFO_CLEAR	Set this bit to clear all the data in the latency FIFO (LFIFO), TXFIFO and the RXFIFO.
20 IRQ_ON_ ALTERNATE_ FIELDS	If this bit is set, the eLCDIF block will assert the cur_frame_done interrupt only on alternate fields, otherwise it will issue the interrupt on both odd and even field. This bit is mostly relevant if INTERLACE_FIELDS is set. This feature is only available in DOTCLK and DVI modes.
19–16 BYTE_ PACKING_ FORMAT	This bitfield is used to show which data bytes in a 32-bit word are valid. Default value 0xf indicates that all bytes are valid. For 8-bit transfers, any combination in this bitfield will mean valid data is present in the corresponding bytes. In the 16-bit mode, a 16-bit half-word is valid only if adjacent bits [1:0] or [3:2] or both are 1. A value of 0x0 will mean that none of the bytes are valid and should not be used. For example, set the bit field value to 0x7 if the display data is arranged in the 24-bit unpacked format (A-R-G-B where A value does not have be transmitted). When input data is in YCbCr 4:2:2 format (YCBCR422_INPUT is 1), H_COUNT should be the number of pixels that should be fetched by the block and the BYTE_PACKING_FORMAT should be 0xF.(Note - YCBCR422_INPUT = 1 implies 2 pixels per 32 bits).
15 OVERFLOW_ IRQ_EN	This bit is set to enable an overflow interrupt in the TXFIFO in the write mode.
14 UNDERFLOW_ IRQ_EN	This bit is set to enable an underflow interrupt in the TXFIFO in the write mode.
13 CUR_FRAME_ DONE_IRQ_EN	This bit is set to 1 enable an interrupt every time the hardware enters in the vertical blanking state.
12 VSYNC_EDGE_ IRQ_EN	This bit is set to enable an interrupt every time the hardware encounters the leading VSYNC edge in the VSYNC and DOTCLK modes, or the beginning of every field in DVI mode.

Table continues on the next page...

LCDIF_CTRL1*n* field descriptions (continued)

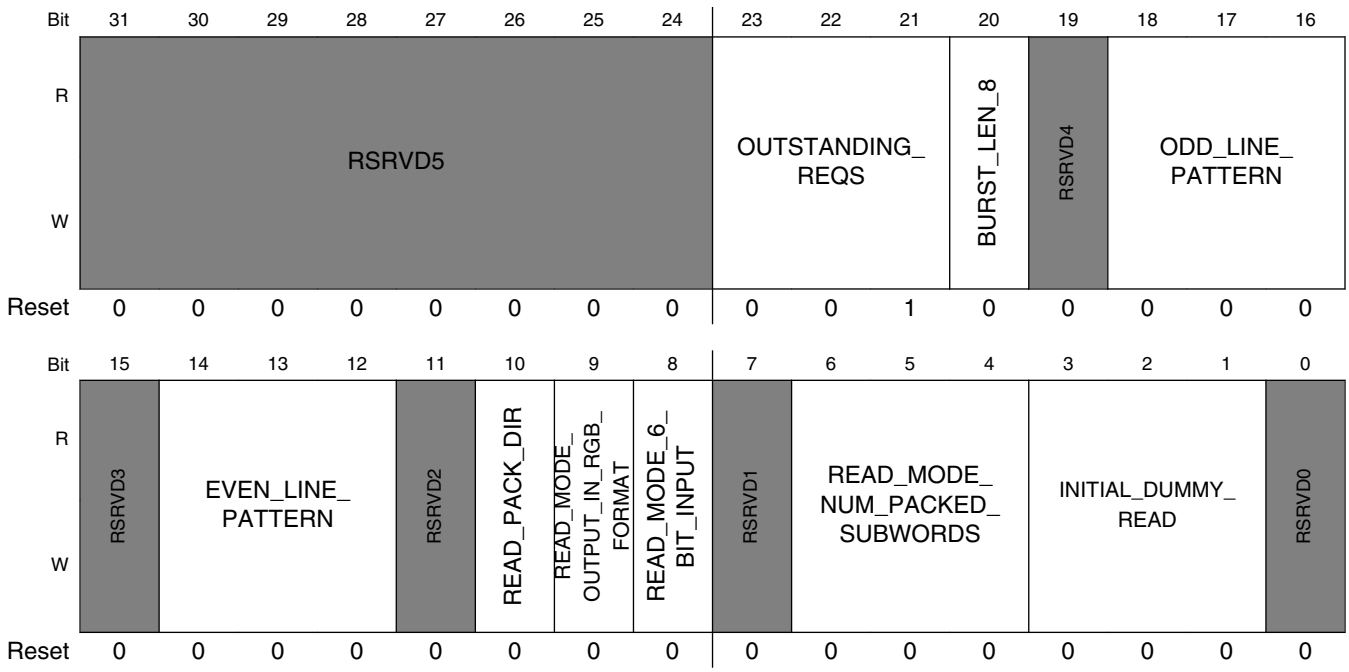
Field	Description
11 OVERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A latency FIFO (LFIFO) overflow in the write mode (MPU/ VSYNC/DOTCLK/DVI mode) was detected, data samples have been lost. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
10 UNDERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A TXFIFO underflow in the write mode (MPU/VSYNC/DOTCLK/ DVI mode) was detected. Could produce an error in the DOTCLK / DVI modes. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
9 CUR_FRAME_ DONE_IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It indicates that the hardware has completed transmitting the current frame and is in the vertical blanking period in the DOTCLK/DVI modes. In the MPU and VSYNC modes, this IRQ is asserted at the end of the data transfer indicated by LCDIF_TRANSFER_COUNT register. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
8 VSYNC_EDGE_ IRQ	This bit is set to indicate that an interrupt is requested by the eLCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It is set whenever the leading VSYNC edge is detected in the VSYNC and DOTCLK modes. In the DVI mode, it is asserted every time the block enters a new field. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
7–3 RSRVD0	Reserved bits. Write as 0.
2 BUSY_ENABLE	This bit enables the use of the interface's busy signal input. This should be enabled for LCD controllers that implement a busy line (to stall the eLCDIF from sending more data until ready). Otherwise this bit should be cleared. 0x0 **BUSY_DISABLED** — The busy signal from the LCD controller will be ignored. 0x1 **BUSY_ENABLED** — Enable the use of the busy signal from the LCD controller.
1 MODE86	This bit is used to select between the 8080 and 6800 series of microprocessor modes. This bit should only be changed when RUN is 0. 0x0 **8080_MODE** — Pins LCD_WR_RWn and LCD_RD_E function as active low WR and active low RD signals respectively. 0x1 **6800_MODE** — Pins LCD_WR_RWn and LCD_RD_E function as Read/Writeb and active high Enable signals respectively.
0 RESET	Reset bit for the external LCD controller. This bit can be changed at any time. It CANNOT be reset by SFTRST. 0x0 **LCDRESET_LOW** — LCD_RESET output signal is low. 0x1 **LCDRESET_HIGH** — LCD_RESET output signal is high.

37.4.3 eLCDIF General Control2 Register (LCDIF_CTRL2*n*)

The eLCDIF Control Register provides overall control of the eLCDIF block.

The eLCDIF Control2 Register provides additional programming to the eLCDIF. It implements some bits which are unlikely to change often in a particular application.

Addresses: LCDIF_CTRL2 is 4100_A000h base + 20h offset = 4100_A020h
LCDIF_CTRL2_SET is 4100_A000h base + 24h offset = 4100_A024h
LCDIF_CTRL2_CLR is 4100_A000h base + 28h offset = 4100_A028h
LCDIF_CTRL2_TOG is 4100_A000h base + 2Ch offset = 4100_A02Ch



LCDIF_CTRL2*n* field descriptions

Field	Description
31–24 RSRVD5	Reserved bits. Write as 0.
23–21 OUTSTANDING_ REQS	This bitfield indicates the maximum number of outstanding transactions that eLCDIF should request when it is acting as a bus master. Default is 2 outstanding transactions. 0x0 **REQ_1** — 0x1 **REQ_2** — 0x2 **REQ_4** — 0x3 **REQ_8** — 0x4 **REQ_16** —
20 BURST_LEN_8	By default, when the eLCDIF is in the bus master mode, it will issue AXI bursts of length 16 (except when in packed 24 bpp mode, it will issue bursts of length 15). When this bit is set to 1, the block will

Table continues on the next page...

LCDIF_CTRL2*n* field descriptions (continued)

Field	Description
	issue bursts of length 8 (except when in packed 24 bpp mode, it will issue bursts of length 9). Note that this bitfield is only applicable when LCDIF_MASTER is set to 1.
19 RSRVD4	Reserved bits. Write as 0.
18–16 ODD_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in ODD lines (line numbers 1,3,5,..). This bitfield must be 0 in DVI mode. 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** — 0x4 **BRG** — 0x5 **BGR** —
15 RSRVD3	Reserved bits. Write as 0.
14–12 EVEN_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in EVEN lines (line numbers 2,4,6,..). This bitfield must be 0 in DVI mode. 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** — 0x4 **BRG** — 0x5 **BGR** —
11 RSRVD2	Reserved bits. Write as 0.
10 READ_PACK_DIR	The default value of 0 indicates data is stored in the little endian format. When LCD_DATABUS_WIDTH is 8-bit, this bit provides the option of rearranging the data byte-wise in the big endian format. For example, if READ_MODE_NUM_PACKED_SUBWORDS = 3 and the order of incoming data is 0x11, 0x22 and 0x33, then setting this bit to 1 will cause the data to be stored as 0x00112233 as opposed to the default 0x00332211. This operation occurs after the shifting operation done by SHIFT_NUM_BITS bitfield.
9 READ_MODE_ OUTPUT_IN_ RGB_FORMAT	Setting this bit will enable the eLCDIF to convert the incoming data to the RGB format given by WORD_LENGTH bitfield. This feature is not available when WORD_LENGTH is set to 8 bits. eLCDIF performs this operation of converting to RGB format after the endianness has been determined by the READ_PACK_DIR bitfield.
8 READ_MODE_6_ BIT_INPUT	Setting this bit to 1 indicates to eLCDIF that even though LCD_DATABUS_WIDTH is set to 8 bits, the input data is actually only 6 bits wide and exists on D5-D0.
7 RSRVD1	Reserved bits. Write as 0.
6–4 READ_MODE_ NUM_PACKED_ SUBWORDS	Indicates the number of valid 8/16/18/24-bit subwords that will be packed into the 32-bit word in read mode. The subword size (8,16, 18 or 24 bits) is determined by the LCD_DATABUS_WIDTH field. The swizzle operation is performed after READ_MODE_NUM_PACKED_SUBWORDS number of subwords has been received and stored in little-endian format. For example, if LCD_DATABUS_WIDTH is set to 8-bit and data to be read back has to be stored in memory in 24-bit unpacked RGB format, set READ_MODE_NUM_PACKED_SUBWORDS to 0x3 so that each 32-bit word will contain only 3 valid

Table continues on the next page...

LCDIF_CTRL2*n* field descriptions (continued)

Field	Description
	bytes (RGB). Maximum value of READ_MODE_NUM_PACKED_SUBWORDS is 4 for 8-bit databus, 2 for 16-bit databus and 1 for 18/24-bit databus.
3–1 INITIAL_DUMMY_READ	The value in this field determines the number of dummy 8/16/18/24-bit subwords that have to be read back from the LCD panel/controller. They will then not be stored in the read FIFO.
0 RSRVD0	Reserved bits. Write as 0.

37.4.4 eLCDIF Horizontal and Vertical Valid Data Count Register (LCDIF_TRANSFER_COUNT)

This register tells the eLCDIF how much data will be sent for this frame, or transaction. The total number of words is a product of the V_COUNT and H_COUNT fields. The word size is specified by the WORD_LENGTH field.

This register gives the dimensions of the input frame. For normal operation, but V_COUNT and H_COUNT should be non-zero.

Address: LCDIF_TRANSFER_COUNT is 4100_A000h base + 30h offset = 4100_A030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V_COUNT																H_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_TRANSFER_COUNT field descriptions

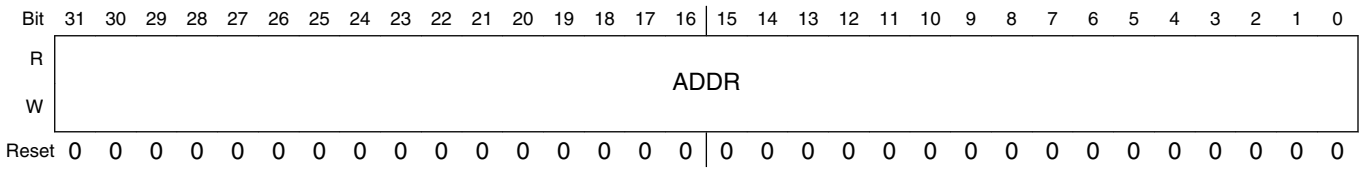
Field	Description
31–16 V_COUNT	Number of horizontal lines per frame which contain valid data. In DOTCLK mode, V_COUNT should be the same as the number of active horizontal lines in a progressive frame. In DVI mode, V_COUNT should be the number of active horizontal lines per frame, and not per field.
15–0 H_COUNT	Total valid data (pixels) in each horizontal line. The data size is given by the WORD_LENGTH. When input data is is in YCbCr 4:2:2 format (YCBCR422_INPUT is 1), H_COUNT should be the number of 32-bit words that should be fetched by the block and the BYTE_PACKING_FORMAT should be 0xF. In 24-bit packed format (WORD_LENGTH=0x3, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 4 pixels. In 16-bit packed format (WORD_LENGTH=0x0, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 2 pixels.

37.4.5 LCD Interface Current Buffer Address Register (LCDIF_CUR_BUF)

This register indicates the address of the current frame being transmitted by eLCDIF.

When the eLCDIF is behaving as a master, this address points to the address of the current frame of data being sent out via the LCDIF. When the current frame is done, the LCDIF block will assert the cur_frame_done interrupt for software to take action. The block will also copy the LCDIF_NEXT_BUF_ADDR into this bitfield so that the software can program the next frame address into the LCDIF_NEXT_BUF_ADDR bitfield. This address must always be double-word aligned.

Address: LCDIF_CUR_BUF is 4100_A000h base + 40h offset = 4100_A040h



LCDIF_CUR_BUF field descriptions

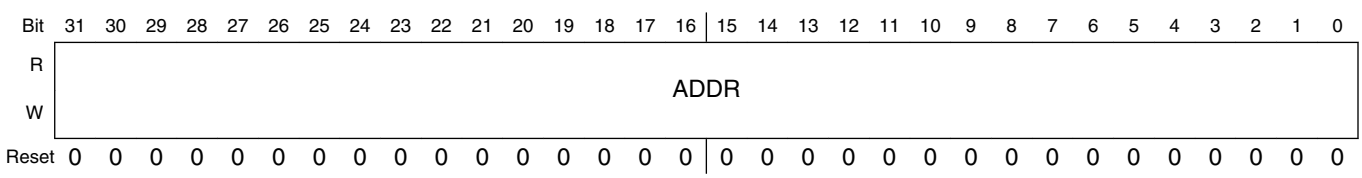
Field	Description
31–0 ADDR	-

37.4.6 LCD Interface Next Buffer Address Register (LCDIF_NEXT_BUF)

This register indicates the address of next frame that will be transmitted by eLCDIF.

When the eLCDIF is behaving as a master, this address points to the address of the next frame of data that will be sent out via the eLCDIF. It is upto the software to make sure that this register is programmed before the end of the current frame, otherwise it might result in old data going out the eLCDIF. This address must always be double-word aligned.

Address: LCDIF_NEXT_BUF is 4100_A000h base + 50h offset = 4100_A050h



LCDIF_NEXT_BUF field descriptions

Field	Description
31–0 ADDR	-

37.4.7 LCD Interface Timing Register (LCDIF_TIMING)

The LCD interface timing register controls the various setup and hold times enforced by the LCD interface in the 6800/8080 MPU and VSYNC modes of operation.

The values used in this register are dependent on the particular LCD controller used, consult the users manual for the particular controller for required timings. Each field of the register must be non-zero, therefore the minimum value is: 0x01010101. NOTE: the timings are not automatically adjusted if the CLK_DIS_LCDIFn frequency changes--it may be necessary to adjust the timings if CLK_DIS_LCDIFn changes. NOTE: Each field in this register must be non-zero for the MPU and VSYNC modes to function. The settings in this register do not affect the DOTCLK and DVI modes.

Address: LCDIF_TIMING is 4100_A000h base + 60h offset = 4100_A060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_HOLD								CMD_SETUP								DATA_HOLD								DATA_SETUP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_TIMING field descriptions

Field	Description
31–24 CMD_HOLD	Number of CLK_DIS_LCDIFn cycles that the DCn signal is active after CEn is deasserted.
23–16 CMD_SETUP	Number of CLK_DIS_LCDIFn cycles that the the DCn signal is active before CEn is asserted.
15–8 DATA_HOLD	Data bus hold time in CLK_DIS_LCDIFn cycles. Also the time that the data strobe is de-asserted in a cycle
7–0 DATA_SETUP	Data bus setup time in CLK_DIS_LCDIFn cycles. Also the time that the data strobe is asserted in a cycle.

37.4.8 eLCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0*n*)

This register is used to control the VSYNC and DOTCLK modes of the LCDIF so as to work with different types of LCDs like moving picture displays and delta pixel displays.

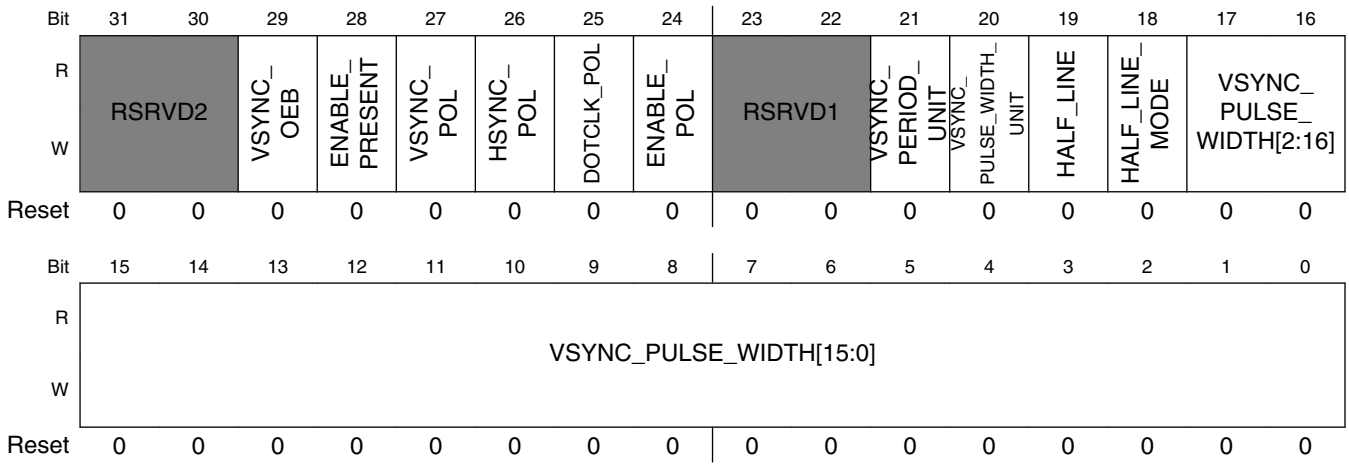
This register gives general programmability to the VSYNC signal including polarity, direction, pulse width, etc.

Addresses: LCDIF_VDCTRL0 is 4100_A000h base + 70h offset = 4100_A070h

LCDIF_VDCTRL0_SET is 4100_A000h base + 74h offset = 4100_A074h

LCDIF_VDCTRL0_CLR is 4100_A000h base + 78h offset = 4100_A078h

LCDIF_VDCTRL0_TOG is 4100_A000h base + 7Ch offset = 4100_A07Ch



LCDIF_VDCTRL0*n* field descriptions

Field	Description
31–30 RSRVD2	Reserved bits. Write as 0.
29 VSYNC_OEB	0 means the VSYNC signal is an output, 1 means it is an input. Should be set to 0 in the DOTCLK mode. 0x0 **VSYNC_OUTPUT** — The VSYNC pin is in the output mode and the VSYNC signal has to be generated by the eLCDIF block. 0x1 **VSYNC_INPUT** — The VSYNC pin is in the input mode and the LCD controller sends the VSYNC signal to the block.
28 ENABLE_ PRESENT	Setting this bit to 1 will make the hardware generate the ENABLE signal in the DOTCLK mode, thereby making it the true RGB interface along with the remaining three signals VSYNC, HSYNC and DOTCLK.
27 VSYNC_POL	Default 0 active low during VSYNC_PULSE_WIDTH time and will be high during the rest of the VSYNC period. Set it to 1 to invert the polarity.
26 HSYNC_POL	Default 0 active low during HSYNC_PULSE_WIDTH time and will be high during the rest of the HSYNC period. Set it to 1 to invert the polarity.
25 DOTCLK_POL	Default is data launched at negative edge of DOTCLK and captured at positive edge. Set it to 1 to invert the polarity. Set it to 0 in DVI mode.
24 ENABLE_POL	Default 0 active low during valid data transfer on each horizontal line.
23–22 RSRVD1	Reserved bits. Write as 0.
21 VSYNC_ PERIOD_UNIT	Default 0 for counting VSYNC_PERIOD in terms of CLK_DIS_LCDIFn cycles. Set it to 1 to count in terms of complete horizontal lines. CLK_DIS_LCDIFn cycles should be used in the VSYNC mode, while horizontal line should be used in the DOTCLK mode.

Table continues on the next page...

LCDIF_VDCTRL0*n* field descriptions (continued)

Field	Description
20 VSYNC_ PULSE_WIDTH_ UNIT	Default 0 for counting VSYNC_PULSE_WIDTH in terms of CLK_DIS_LCDIFn cycles. Set it to 1 to count in terms of complete horizontal lines.
19 HALF_LINE	Setting this bit to 1 will make the total VSYNC period equal to the VSYNC_PERIOD field plus half the HORIZONTAL_PERIOD field (i.e. VSYNC_PERIOD field plus half horizontal line), otherwise it is just VSYNC_PERIOD. Should be only used in the DOTCLK mode, not in the VSYNC interface mode.
18 HALF_LINE_ MODE	When this bit is 0, the first field (VSYNC period) will end in half a horizontal line and the second field will begin with half a horizontal line. When this bit is 1, all fields will end with half a horizontal line, and none will begin with half a horizontal line.
17–0 VSYNC_ PULSE_WIDTH	Number of units for which VSYNC signal is active. For the DOTCLK mode, the unit is determined by the VSYNC_PULSE_WIDTH_UNIT. If the VSYNC_PULSE_WIDTH_UNIT is 0 for DOTCLK mode, VSYNC_PULSE_WIDTH must be less than HSYNC_PERIOD. For the VSYNC interface mode, it should be in terms of number of CLK_DIS_LCDIFn cycles only.

37.4.9 eLCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF_VDCTRL1)

This register is used to control the VSYNC signal in the VSYNC and DOTCLK modes of the block.

This register determines the period and duty cycle of the VSYNC signal when it is generated in the block.

Address: LCDIF_VDCTRL1 is 4100_A000h base + 80h offset = 4100_A080h

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

LCDIF_VDCTRL1 field descriptions

Field	Description
31–0 VSYNC_ PERIOD	Total number of units between two positive or two negative edges of the VSYNC signal. If HALF_LINE is set, it is implicitly calculated to be VSYNC_PERIOD plus half HSYNC_PERIOD.

37.4.10 LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF_VDCTRL2)

This register is used to control the HSYNC signal in the DOTCLK mode of the block.

This register determines the period and duty cycle of the HSYNC signal when it is generated in the block.

Address: LCDIF_VDCTRL2 is 4100_A000h base + 90h offset = 4100_A090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HSYNC_PULSE_WIDTH																HSYNC_PERIOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_VDCTRL2 field descriptions

Field	Description
31–18 HSYNC_ PULSE_WIDTH	Number of CLK_DIS_LCDIFn cycles for which HSYNC signal is active.
17–0 HSYNC_ PERIOD	Total number of CLK_DIS_LCDIFn cycles between two positive or two negative edges of the HSYNC signal.

37.4.11 eLCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF_VDCTRL3)

This register is used to determine the vertical and horizontal wait counts.

This register determines the back porches of HSYNC and VSYNC signals when they are generated by the block.

Address: LCDIF_VDCTRL3 is 4100_A000h base + A0h offset = 4100_A0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSRVD0		MUX_SYNC_SIGNALS	VSYNC_ONLY	HORIZONTAL_WAIT_CNT											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VERTICAL_WAIT_CNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_VDCTRL3 field descriptions

Field	Description
31–30 RSRVD0	Reserved bits, write as 0.
29 MUX_SYNC_ SIGNALS	When this bit is set, the eLCDIF block will internally mux HSYNC with LCD_D14, DOTCLK with LCD_D13 and ENABLE with LCD_D12, otherwise these signals will go out on separate pins. This feature can be used to maintain backward compatability with 37xx.
28 VSYNC_ONLY	This bit must be set to 1 in the VSYNC mode of operation, and 0 in the DOTCLK mode of operation.
27–16 HORIZONTAL_ WAIT_CNT	In the DOTCLK mode, wait for this number of clocks from falling edge (or rising if HSYNC_POL is 1) of HSYNC signal to account for horizontal back porch plus the number of DOTCLKs before the moving picture information begins.
15–0 VERTICAL_ WAIT_CNT	In the VSYNC interface mode, wait for this number of CLK_DIS_LCDIFn cycles from the falling VSYNC edge (or rising if VSYNC_POL is 1) before starting LCD transactions and is applicable only if WAIT_FOR_VSYNC_EDGE is set. Minimum is CMD_SETUP+5. In the DOTCLK mode, it accounts for the veritcal back porch lines plus the number of horizontal lines before the moving picture begins. The unit for this parameter is inherently the same as the VSYNC_PERIOD_UNIT.

37.4.12 eLCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF_VDCTRL4)

This register is used to control the DOTCLK mode of the block.

This register determines the active data in each horizontal line in the DOTCLK mode. Note that the total number of active horizontal lines in the DOTCLK mode is the same as the V_COUNT bitfield in the LCDIF_TRANSFER_COUNT register.

Address: LCDIF_VDCTRL4 is 4100_A000h base + B0h offset = 4100_A0B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DOTCLK_DLY_SEL			RSRVD0										SYNC_ SIGNALS_ON	DOTCLK_H_ VALID_ DATA_CNT [2:16]	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DOTCLK_H_VALID_DATA_CNT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_VDCTRL4 field descriptions

Field	Description
31–29 DOTCLK_DLY_SEL	This bitfield selects the amount of time by which the DOTCLK signal should be delayed before coming out of the LCD_DOTCK pin. 0 = 2ns; 1=4ns;2=6ns;3=8ns. Remaining values are reserved.
28–19 RSRVD0	Reserved bits, write as 0.
18 SYNC_SIGNALS_ON	Set this field to 1 if the LCD controller requires that the VSYNC or VSYNC/HSYNC/DOTCLK control signals should be active atleast one frame before the data transfers actually start and remain active atleast one frame after the data transfers end. The hardware does not count the number of frames automatically. Rather, the VSYNC edge interrupt can be monitored by software to count the number of frames that have occured after this bit is set and then the RUN bit can be set to start the data transactions. This bit must always be set in the DOTCLK mode of operation, and it must be set in the VSYNC mode of operation when VSYNC signal is an output.
17–0 DOTCLK_H_VALID_DATA_CNT	Total number of CLK_DIS_LCDIFn cycles on each horizontal line that carry valid data in DOTCLK mode.

37.4.13 Digital Video Interface Control0 Register (LCDIF_DVICTRL0)

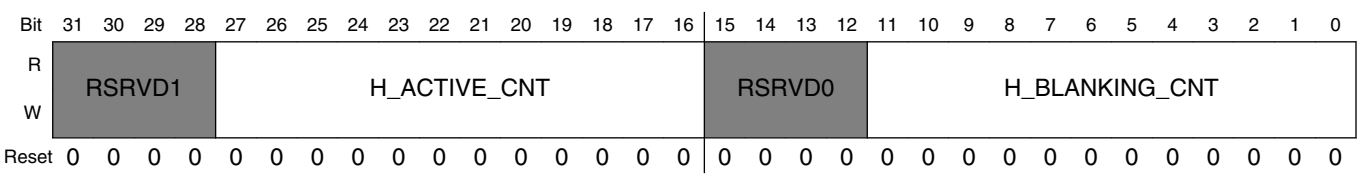
The Digital Video interface Control0 register provides the overall control of the Digital Video interface.

This register gives information about the horizontal active, horizontal blanking and total number of lines in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0);//1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x106);//262
//625/50 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0);//1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x112);//274
```

Address: LCDIF_DVICTRL0 is 4100_A000h base + C0h offset = 4100_A0C0h



LCDIF_DVICTRL0 field descriptions

Field	Description
31–28 RSRVD1	Reserved bits, write as 0.
27–16 H_ACTIVE_CNT	Number of active video samples to be transmitted. (Mostly will be 1440 for both PAL and NTSC). Must always be a multiple of 4.
15–12 RSRVD0	Reserved bits, write as 0.
11–0 H_BLANKING_ CNT	Number of blanking samples to be inserted between EAV and SAV during horizontal blanking interval.

37.4.14 Digital Video Interface Control1 Register (LCDIF_DVICTRL1)

The Digital Video interface Control1 register provides the overall control of the Digital Video interface.

This register contains information about the Field1 start and end, and the Field2 start in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x4);//4
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x109);//265
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x10A);//266
//625/50 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x138);//312
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x139);//313
```

Address: LCDIF_DVICTRL1 is 4100_A000h base + D0h offset = 4100_A0D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSRVD0		F1_START_LINE										F1_END_LINE[-6:6]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F1_END_LINE[5:0]								F2_START_LINE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DVICTRL1 field descriptions

Field	Description
31–30 RSRVD0	Reserved bits, write as 0.
29–20 F1_START_ LINE	Vertical line number from which Field 1 begins.
19–10 F1_END_LINE	Vertical line number at which Field1 ends.
9–0 F2_START_ LINE	Vertical line number from which Field 2 begins.

37.4.15 Digital Video Interface Control2 Register (LCDIF_DVICTRL2)

The Digital Video interface Control2 register provides the overall control of the Digital Video interface.

This register contains information about the Field2 end, and the Vertical Blanking1 interval in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x3);//3
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x108);//264
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x11A);//282
//625/50 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x271);//625
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x137);//311
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x14F);//335
```

Address: LCDIF_DVICTRL2 is 4100_A000h base + E0h offset = 4100_A0E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSRVD0		F2_END_LINE										V1_BLANK_START_LINE [-6:6]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V1_BLANK_START_LINE[5:0]								V1_BLANK_END_LINE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DVICTRL2 field descriptions

Field	Description
31–30 RSRVD0	Reserved bits, write as 0.
29–20 F2_END_LINE	Vertical line number at which Field 2 ends.
19–10 V1_BLANK_ START_LINE	Vertical line number towards the end of Field1 where first Vertical Blanking interval starts.
9–0 V1_BLANK_ END_LINE	Vertical line number in the beginning part of Field2 where first Vertical Blanking interval ends.

37.4.16 Digital Video Interface Control3 Register (LCDIF_DVICTRL3)

The Digital Video interface Control3 register provides the overall control of the Digital Video interface.

This register contains information about the Vertical Blanking2 interval in the ITU-R BT. 656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL3_V2_BLANK_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL3_V2_BLANK_END_LINE_WR(0x13);//19
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x20D);//525
//625/50 video system
HW_LCDIF_DVICTRL3_V2_BLANK_START_LINE_WR(0x270);//624
HW_LCDIF_DVICTRL3_V2_BLANK_END_LINE_WR(0x16);//22
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x271);//625
```

Address: LCDIF_DVICTRL3 is 4100_A000h base + F0h offset = 4100_A0F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSRVD0		V2_BLANK_START_LINE										V2_BLANK_END_LINE [-6:6]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V2_BLANK_END_LINE[5:0]						V_LINES_CNT									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DVICTRL3 field descriptions

Field	Description
31–30 RSRVD0	Reserved bits, write as 0.
29–20 V2_BLANK_ START_LINE	Vertical line number towards the end of Field2 where second Vertical Blanking interval starts.
19–10 V2_BLANK_ END_LINE	Vertical line number in the beginning part of Field1 where second Vertical Blanking interval ends.
9–0 V_LINES_CNT	Total number of vertical lines per frame (generally 525 or 625)

37.4.17 Digital Video Interface Control4 Register (LCDIF_DVICTRL4)

The Digital Video interface Control4 register provides the overall control of the Digital Video interface.

This register is used to add side borders to the output if the input frame width is less than 720 pixels.

EXAMPLE

```
                //If input frame has only 640 pixels per line, but output is supposed to have
720 pixels per line.
                HW_LCDIF_DVICTRL4_H_FILL_CNT_WR(0x50);//80
                HW_LCDIF_DVICTRL4_Y_FILL_VALUE_WR(0x10);//16
                HW_LCDIF_DVICTRL4_CB_FILL_VALUE_WR(0x80);//128
                HW_LCDIF_DVICTRL4_CR_FILL_VALUE_WR(0x80);//128
```

Address: LCDIF_DVICTRL4 is 4100_A000h base + 100h offset = 4100_A100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Y_FILL_VALUE								CB_FILL_VALUE								CR_FILL_VALUE								H_FILL_CNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DVICTRL4 field descriptions

Field	Description
31–24 Y_FILL_VALUE	Value of Y component of filler data
23–16 CB_FILL_ VALUE	Value of CB component of filler data

Table continues on the next page...

LCDIF_DVICTRL4 field descriptions (continued)

Field	Description
15–8 CR_FILL_ VALUE	Value of CR component of filler data.
7–0 H_FILL_CNT	Number of active video samples that have to be filled with the filler data in the front and back portions of the active horizontal interval. Must be a multiple of 4. This field will have to be programmed if the input frame has less than 720 pixels per line.

37.4.18 RGB to YCbCr 4:2:2 CSC Coefficient0 Register (LCDIF_CSC_COEFF0)

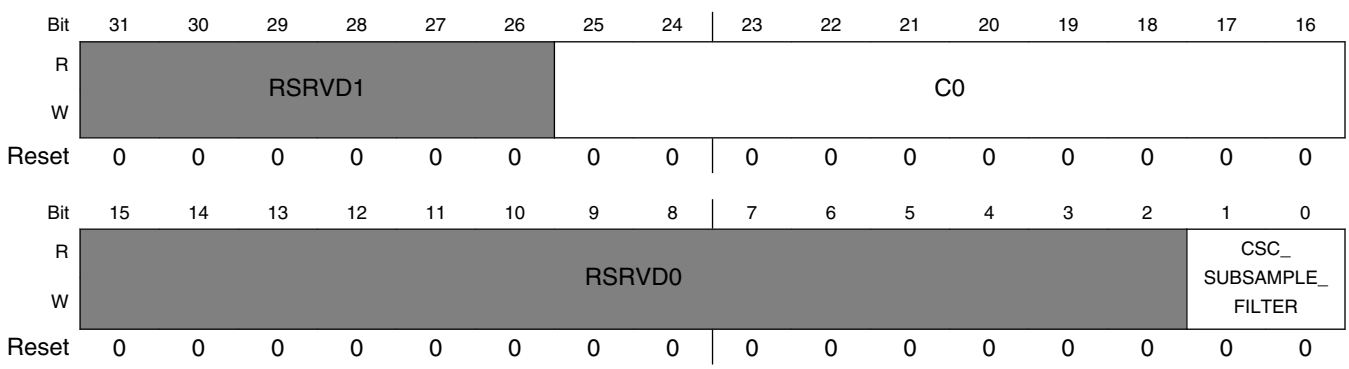
LCDIF_CSC_COEFF0 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF0_C0_WR(0x41);//0.257x256=65
HW_LCDIF_CSC_COEFF0_CSC_SUBSAMPLE_FILTER_WR(0x3);
```

Address: LCDIF_CSC_COEFF0 is 4100_A000h base + 110h offset = 4100_A110h



LCDIF_CSC_COEFF0 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C0	Two's complement red multiplier coefficient for Y

Table continues on the next page...

LCDIF_CSC_COEFF0 field descriptions (continued)

Field	Description
15–2 RSRVD0	Reserved bits, write as 0.
1–0 CSC_ SUBSAMPLE_ FILTER	This register describes the filtering and subsampling scheme to be performed on the chroma components in order to convert from YCbCr 4:4:4 to YCbCr 4:2:2 space. Note that the following descriptions apply individually to Cb and Cr. 0x0 **SAMPLE_AND_HOLD** — No filtering, simply keep every chroma value for samples numbered 2n and discard chroma values associated with all samples numbered 2n+1. 0x1 **RSRVD** — Reserved 0x2 **INTERSTITIAL** — Chroma samples numbered 2n and 2n+1 are averaged (weights 1/2, 1/2) and that chroma value replaces the two chroma values at 2n and 2n+1. This chroma now exists horizontally halfway between the two luma samples. 0x3 **COSITED** — Chroma samples numbered 2n-1, 2n, and 2n+1 are averaged (weights 1/4,1/2,1/4) and that chroma value exists at the same site as the luma sample numbered 2n and the chroma samples at 2n+1 are discarded.

37.4.19 RGB to YCbCr 4:2:2 CSC Coefficient1 Register (LCDIF_CSC_COEFF1)

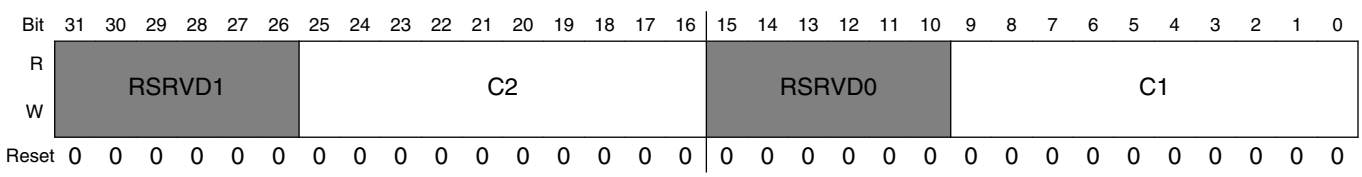
LCDIF_CSC_COEFF1 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF1_C1_WR(0x81);//0.504x256=129
HW_LCDIF_CSC_COEFF1_C2_WR(0x19);//0.098x256=25
```

Address: LCDIF_CSC_COEFF1 is 4100_A000h base + 120h offset = 4100_A120h



LCDIF_CSC_COEFF1 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.

Table continues on the next page...

LCDIF_CSC_COEFF1 field descriptions (continued)

Field	Description
25–16 C2	Two's complement blue multiplier coefficient for Y
15–10 RSRVD0	Reserved bits, write as 0.
9–0 C1	Two's complement green multiplier coefficient for Y

37.4.20 RGB to YCbCr 4:2:2 CSC Coefficent2 Register (LCDIF_CSC_COEFF2)

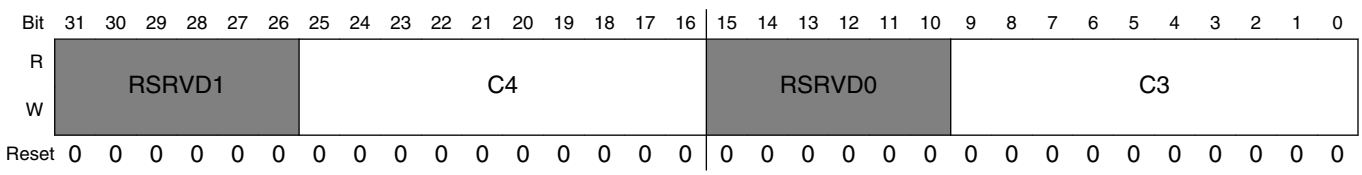
LCDIF_CSC_COEFF2 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF2_C3_WR(0x3DB);//-0.148x256=-37
HW_LCDIF_CSC_COEFF2_C4_WR(0x3B6);//-0.291x256=-74
```

Address: LCDIF_CSC_COEFF2 is 4100_A000h base + 130h offset = 4100_A130h



LCDIF_CSC_COEFF2 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C4	Two's complement green multiplier coefficient for Cb
15–10 RSRVD0	Reserved bits, write as 0.
9–0 C3	Two's complement red multiplier coefficient for Cb

37.4.21 RGB to YCbCr 4:2:2 CSC Coefficient3 Register (LCDIF_CSC_COEFF3)

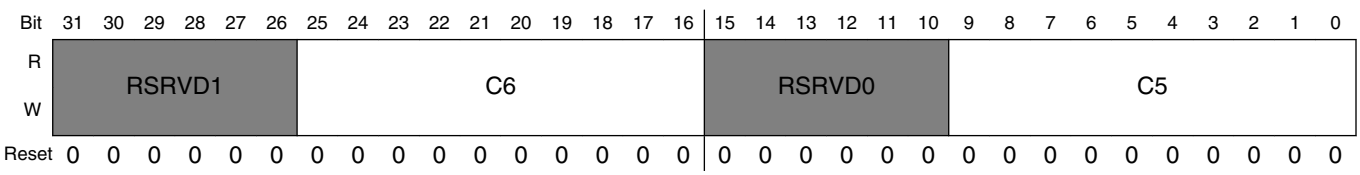
LCDIF_CSC_COEFF3 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF3_C5_WR(0x70);//0.439x256=112
HW_LCDIF_CSC_COEFF3_C6_WR(0x70);//0.439x256=112
```

Address: LCDIF_CSC_COEFF3 is 4100_A000h base + 140h offset = 4100_A140h



LCDIF_CSC_COEFF3 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C6	Two's complement red multiplier coefficient for Cr
15–10 RSRVD0	Reserved bits, write as 0.
9–0 C5	Two's complement blue multiplier coefficient for Cb

37.4.22 RGB to YCbCr 4:2:2 CSC Coefficient4 Register (LCDIF_CSC_COEFF4)

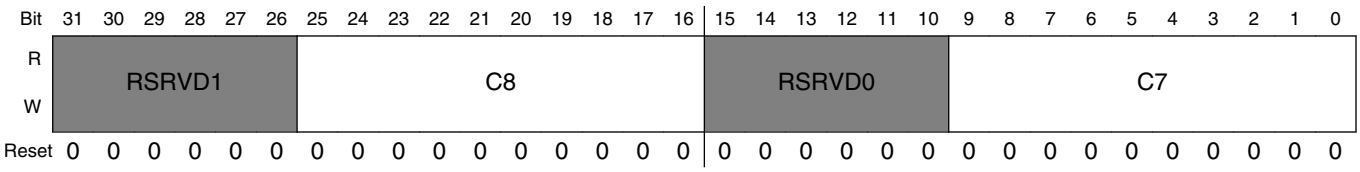
LCDIF_CSC_COEFF4 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF4_C7_WR(0x3A2);//-0.368x256=-94
HW_LCDIF_CSC_COEFF4_C8_WR(0x3EE);//-0.071x256=-18
```

Address: LCDIF_CSC_COEFF4 is 4100_A000h base + 150h offset = 4100_A150h



LCDIF_CSC_COEFF4 field descriptions

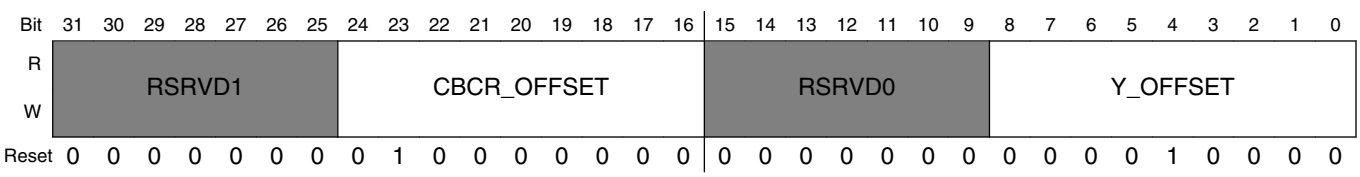
Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C8	Two's complement blue multiplier coefficient for Cr
15–10 RSRVD0	Reserved bits, write as 0.
9–0 C7	Two's complement green multiplier coefficient for Cr

37.4.23 RGB to YCbCr 4:2:2 CSC Offset Register (LCDIF_CSC_OFFSET)

LCDIF_CSC_ register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

Address: LCDIF_CSC_OFFSET is 4100_A000h base + 160h offset = 4100_A160h



LCDIF_CSC_OFFSET field descriptions

Field	Description
31–25 RSRVD1	Reserved bits, write as 0.
24–16 CBCR_OFFSET	Two's complement offset for the Cb and Cr components
15–9 RSRVD0	Reserved bits, write as 0.
8–0 Y_OFFSET	Two's complement offset for the Y component

37.4.24 RGB to YCbCr 4:2:2 CSC Limit Register (LCDIF_CSC_LIMIT)

LCDIF_CSC_CTRL0 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC. Note that the values in this register are unsigned.

EXAMPLE

```
HW_LCDIF_CSC_LIMIT_CBCR_MIN_WR(0x10);//16
HW_LCDIF_CSC_LIMIT_CBCR_MAX_WR(0xF0);//240
HW_LCDIF_CSC_LIMIT_Y_MIN_WR(0x10);//16
HW_LCDIF_CSC_LIMIT_Y_MAX_WR(0xEB);//235
```

Address: LCDIF_CSC_LIMIT is 4100_A000h base + 170h offset = 4100_A170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CBCR_MIN								CBCR_MAX								Y_MIN								Y_MAX							
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

LCDIF_CSC_LIMIT field descriptions

Field	Description
31–24 CBCR_MIN	Lower limit of Cb and Cr after RGB to 4:2:2 YCbCr conversion
23–16 CBCR_MAX	Upper limit of Cb and Cr after RGB to 4:2:2 YCbCr conversion
15–8 Y_MIN	Lower limit of Y after RGB to 4:2:2 YCbCr conversion

Table continues on the next page...

LCDIF_CSC_LIMIT field descriptions (continued)

Field	Description
7–0 Y_MAX	Upper limit of Y after RGB to 4:2:2 YCbCr conversion

37.4.25 LCD Interface Data Register (LCDIF_DATA)

NOTE: this is now an unsupported feature of the eLCDIF and this should be removed from the design. The data sent to an external LCD controller is written to this register. Data can be written to this register (from the processor\'s perspective) as bytes half-words (16 bits) or words (32 bits) as appropriate.

This register holds the 32-bit word written by either the ARM platform or the DMA into LCDIF. This data then gets sent out by the block across the interface. When the block is in bus master mode, this register gets the value of the lower 32 bits of the 64-bit data bus whenever it is received.

Address: LCDIF_DATA is 4100_A000h base + 180h offset = 4100_A180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA_THREE								DATA_TWO								DATA_ONE								DATA_ZERO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DATA field descriptions

Field	Description
31–24 DATA_THREE	Byte 3 (most significant byte) of data written to LCDIF by the DMA or the ARM platform.
23–16 DATA_TWO	Byte 2 of data written to eLCDIF by the DMA or the ARM platform.
15–8 DATA_ONE	Byte 1 of data written to eLCDIF by the DMA or the ARM platform.
7–0 DATA_ZERO	Byte 0 (least significant byte) of data written to eLCDIF by the DMA or the ARM platform.

37.4.26 Bus Master Error Status Register (LCDIF_BM_ERROR_STAT)

This register reflects the virtual address at which the AXI master received an error response from the slave.

When the BM_ERROR_IRQ is asserted, the address of the bus error is updated in the register.

Address: LCDIF_BM_ERROR_STAT is 4100_A000h base + 190h offset = 4100_A190h

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

LCDIF_BM_ERROR_STAT field descriptions

Field	Description
31–0 ADDR	Virtual address at which bus master error occurred.

37.4.27 CRC Status Register (LCDIF_CRC_STAT)

This register reflects the CRC value of each frame sent out by eLCDIF. The CRC is done on the final output bus, so the value will be dependent on the LCD_DATABUS_WIDTH bitfield even if the input data is the same.

This register will be updated when the CUR_FRAME_DONE_IRQ is asserted. In the case of DVI mode, the CRC is calculated for the entire frame, not separately for each field in the frame.

Address: LCDIF_CRC_STAT is 4100_A000h base + 1A0h offset = 4100_A1A0h

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

LCDIF_CRC_STAT field descriptions

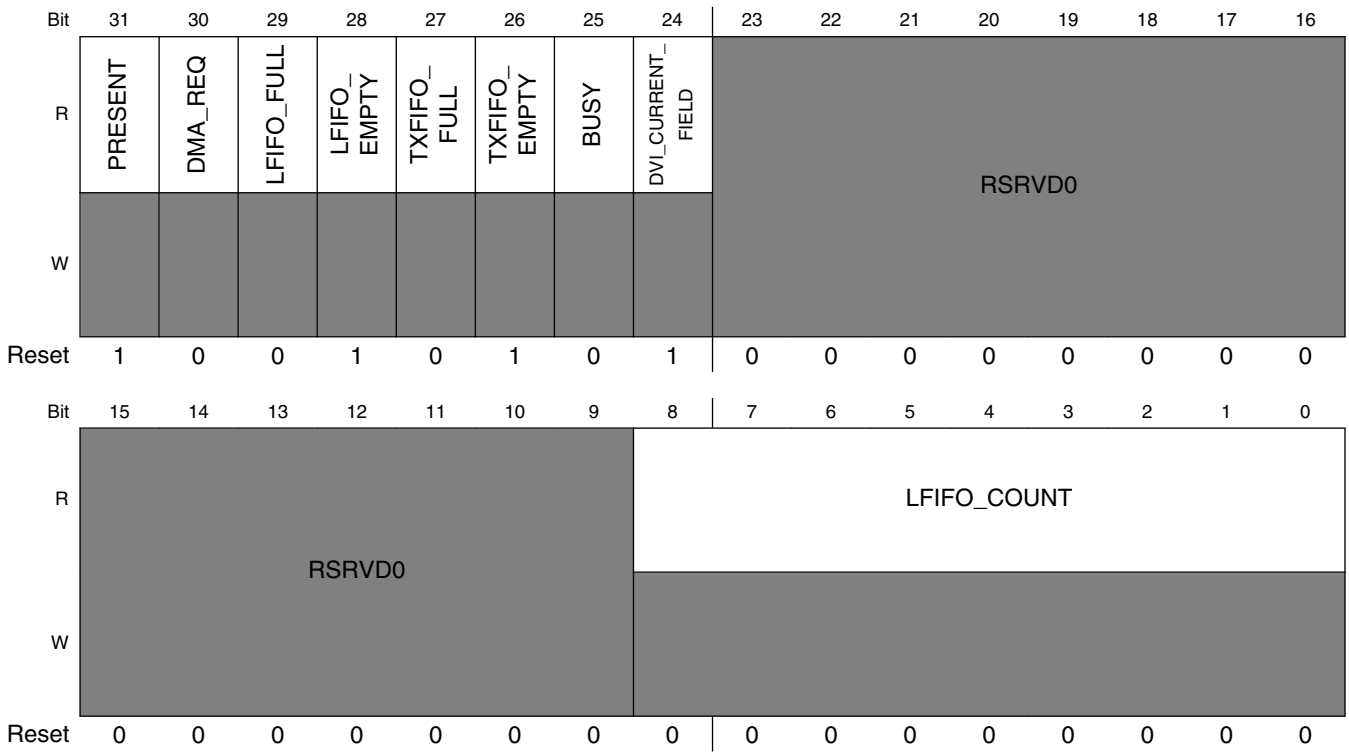
Field	Description
31–0 CRC_VALUE	Calculated CRC value.

37.4.28 LCD Interface Status Register (LCDIF_STAT)

The LCD interface status register can be used to check the current status of the eLCDIF block.

The LCD interface status register that contains read only views of some parameters or current state of the block.

Address: LCDIF_STAT is 4100_A000h base + 1B0h offset = 4100_A1B0h



LCDIF_STAT field descriptions

Field	Description
31 PRESENT	0: eLCDIF not present on this product 1: eLCDIF is present.
30 DMA_REQ	Reflects the current state of the DMA Request line for the eLCDIF. The DMA Request line toggles for each new request.
29 LFIFO_FULL	Read only view of the signal that indicates that LCD read datapath FIFO is full, will be generally used in the write mode of the LCD interface.
28 LFIFO_EMPTY	Read only view of the signal that indicates that LCD read dapatath FIFO is empty, will be generally used in the read mode of the LCD interface.
27 TXFIFO_FULL	Read only view of the signal that indicates that LCD write datapath FIFO is full, will be generally used in the write mode of the LCD interface.
26 TXFIFO_EMPTY	Read only view of the signal that indicates that LCD write dapatath FIFO is empty, will be generally used in the read mode of the LCD interface.
25 BUSY	Read only view of the input busy signal from the external LCD controller.
24 DVI_CURRENT_ FIELD	Read only view of the current field being transmitted. DVI_CURRENT_FIELD = 0 means field 1. DVI_CURRENT_FIELD = 1 means field 2.
23–9 RSRVD0	Reserved bits. Write as 0.
8–0 LFIFO_COUNT	Read only view of the current count in Latency buffer (LFIFO).

37.4.29 LCD Interface Version Register (LCDIF_VERSION)

The LCD interface version register can be used to read the version of the eLCDIF IP being used in this SoC.

The LCD interface debug register is for diagnostic use only.

Address: LCDIF_VERSION is 4100_A000h base + 1C0h offset = 4100_A1C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of RTL version.

37.4.30 LCD Interface Debug0 Register (LCDIF_DEBUG0)

The LCD interface debug0 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: LCDIF_DEBUG0 is 4100_A000h base + 1D0h offset = 4100_A1D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	STREAMING_END_DETECTED	WAIT_FOR_VSYNC_EDGE_OUT	SYNC_SIGNALS_ON_REG	DMACMDKICK	ENABLE	HSYNC	VSYNC	CUR_FRAME_TX	EMPTY_WORD	CUR_STATE						
W																
Reset	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PXP_LCDIF_B0_READY	PXP_B0_DONE	PXP_LCDIF_B1_READY	PXP_B1_DONE	CUR_REQ_STATE		MST_AVALID	MST_OUTSTANDING_REQS					MST_WORDS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_DEBUG0 field descriptions

Field	Description
31 STREAMING_END_DETECTED	Read only view of the DOTCLK_MODE or DVI_MODE bit going from 1 to 0.
30 WAIT_FOR_VSYNC_EDGE_OUT	Read only view of WAIT_FOR_VSYNC_EDGE bit in the VSYNC mode after it comes out of the TXFIFO.
29 SYNC_SIGNALS_ON_REG	Read only view of internal sync_signals_on_reg signal.
28 DMACMDKICK	Read only view of the DMA command kick signal.
27 ENABLE	Read only view of ENABLE signal.
26 HSYNC	Read only view of HSYNC signal.

Table continues on the next page...

LCDIF_DEBUG0 field descriptions (continued)

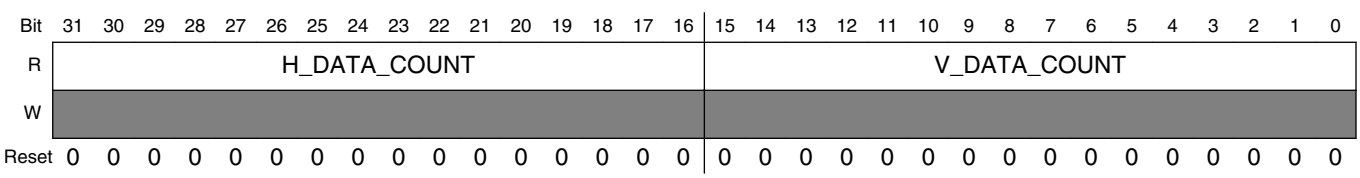
Field	Description
25 VSYNC	Read only view of VSYNC signal.
24 CUR_FRAME_TX	This bit is 1 for the time the current frame is being transmitted in the VSYNC mode. Useful for VSYNC mode debug.
23 EMPTY_WORD	Indicates that the current word is empty.
22–16 CUR_STATE	Read only view of the current state machine state in the current mode of operation.
15 PXP_LCDIF_B0_READY	Buffer0 ready signal issued by ePXP.
14 PXP_B0_DONE	Buffer0 done signal issued by eLCDIF.
13 PXP_LCDIF_B1_READY	Buffer1 ready signal issued by ePXP.
12 PXP_B1_DONE	Buffer1 done signal issued by eLCDIF.
11–10 CUR_REQ_STATE	Read only view of the request state machine.
9 MST_AVALID	Read only view of the mst_avalid signal issued by the AXI bus master.
8–4 MST_OUTSTANDING_REQS	Read only view of the current outstanding requests issued by the AXI bus master.
3–0 MST_WORDS	Read only view of the current bursts issued by the AXI bus master.

37.4.31 LCD Interface Debug1 Register (LCDIF_DEBUG1)

The LCD interface debug1 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: LCDIF_DEBUG1 is 4100_A000h base + 1E0h offset = 4100_A1E0h



LCDIF_DEBUG1 field descriptions

Field	Description
31–16 H_DATA_ COUNT	Read only view of the current state of the horizontal data counter.
15–0 V_DATA_ COUNT	Read only view of the current state of the vertical data counter.

37.4.32 LCD Interface Debug2 Register (LCDIF_DEBUG2)

The LCD interface debug2 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: LCDIF_DEBUG2 is 4100_A000h base + 1F0h offset = 4100_A1F0h

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

LCDIF_DEBUG2 field descriptions

Field	Description
31–0 MST_ADDRESS	Read only view of the current address issued by the AXI bus master.

37.4.33 eLCDIF Threshold Register (LCDIF_THRES)

This register is used to activate control signals when the number of pixels reaches the programmed threshold. These control signals, in turn, can be used to manipulate access priority or dynamically change the input clock frequency to meet the required pixel throughput.

Memory request priority threshold register.

Address: LCDIF_THRES is 4100_A000h base + 200h offset = 4100_A200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2							FASTCLOCK									RSRVD1							PANIC								
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

LCDIF_THRES field descriptions

Field	Description
31–25 RSRVD2	Reserved bits. Write as 0.
24–16 FASTCLOCK	This value should be set to a value of pixels, from 0 to 511. When the number of pixels in the input pixel FIFO is LESS than this value, the fast clock control output will be raised. This signal can be used to reduce the system bus clock frequency to save power during horizontal/vertical blanking intervals. This value should also be programmed to a value that is greater than the "PANIC" threshold value. This will allow a faster clock to recover the number of pixels in the FIFO before a "panic" level is encountered.
15–9 RSRVD1	Reserved bits. Write as 0.
8–0 PANIC	This value should be set to a value of pixels, from 0 to 511. When the number of pixels in the input pixel FIFO is LESS than this value, the panic control output will be raised. This signal can be used to raise the access eLCDIF's access priority.

Chapter 38
On-Chip OTP (OCOTP) Controller

38.1 Introduction

This chapter describes the requirements for the on-chip TSMC eFuse OTP controller. It describes the block functionality and implementation in detail. In this document, the words “eFuse” and “OTP” are interchangeable. OCOTP refers to the hardware block itself. For more information on eFuse, refer to the latest version of TSMC’s 256-bit serial interface electrical fuse specification.

38.2 Overview of On-Chip OTP APB Slave (OCOTP)

The OCOTP provides the following features :

- 32-bit word restricted access to 1.25 Kbyte of TSMC eFuse OTP ROM through APB.
- Memory mapped access for reading eFuse contents.
- Data-register access for programming eFuse contents.
- Generation of hwv_fuse (hardware visible fuse bus) and the hwv_reg bus which is made up of volatile PIO register based “fuses”. The hwv_reg bits come from the SCS (Software Controllable Signals) register.
- Sending of crypto keys and read enable via parallel connections. The 128-bit AES crypto key is sent 32-bits at a time over 4 apb_clk cycles. This key is comprised of OCOTP_SCC0 through OCOTP_SCC3. The second key (the “unique” key used for content key encryption) is 64-bits sent 32-bits at a time over 2 apb_clk cycles. This key is comprised of OCOTP_CFG0 and OCOTP_CFG1.
- Chip-level pin access to unrestricted portions of OTP.
- Loading and housing of fuse and LOCK shadow registers.

38.3 Top-Level Symbol and Functional Overview

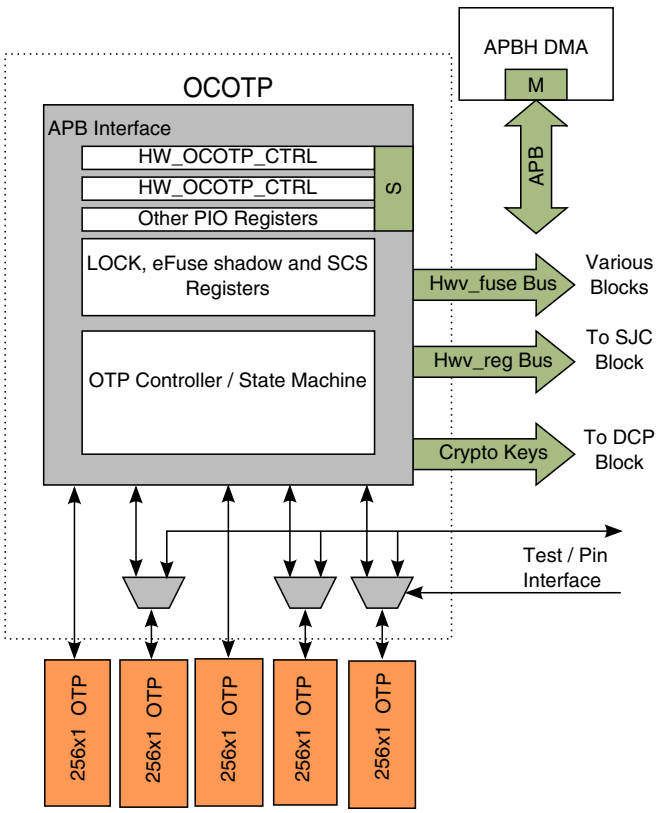


Figure 38-1. OCOTP System Level Diagram

38.3.1 Operation

The APB interface of the OCOTP provides two functions:

- Programmers model access to registers (see OCOTP Memory Map/Register Definition for details).
- Restricted 32-bit word access to the 1.25 Kbyte OTP

OTP reads and writes can only be performed on 32-bit words. For writes, the 32-bit word reflects the “write-mask.” Bit fields with 0 will not be programmed and bit fields with 1 will be programmed.

For OTP random access, the programming interface consists of:

- OCOTP_DATA[31:0] : data register (32- bit) for eFuse programming (writes).

- OCOTP_CTRL[ADDR] : address register (5-bit) for eFuse programming (writes).
- OCOTP_CTRL[BUSY] : Read/Write request/status handshake bit.
- OCOTP_CTRL[RD_BANK_OPEN] : This bit is obsolete in i.MX50 and should not be set.
- OCOTP_CTRL[ERROR] : Read/Write access error status.

38.3.1.1 Fuse and Shadow Register Reads

In past implementations of the OCOTP block the functionality existed to read unshadowed fuse values directly from fuse banks. This was accomplished through the use of the OCOTP_CTRL[RD_BANK_OPEN] bit. This functionality is now obsolete in i.MX50 and should not be used because all fuse bits are shadowed in this implementation.

NOTE

Do not program OCOTP_CTRL[RD_BANK_OPEN] to 1.

Therefore, fuse information is only available through memory mapped shadow registers. If fuses are subsequently programmed, the shadow registers should be reloaded to keep them coherent with the fuse bank arrays.

All shadow registers are always readable through the APB bus except in the SCC and SJC regions. When their corresponding fuse lock bits are set (OCOTP_LOCK[SCC] and OCOTP_LOCK[SJC]), the shadow registers also become read locked. After read locking, reading from these registers will return 0xBADABADA. In addition OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write or reload access can be issued. Subsequent reads to unlocked shadow locations will still work successfully however.

The “reload shadows” feature allows the user to force a reload of the shadow registers (including OCOTP_LOCK) without having to reset the device. To force a reload, complete the following steps:

1. Set the OCOTP_TIMING[RD_BUSY] field value appropriately (as explained in a later section).
2. set the OCOTP_CTRL[RELOAD_SHADOWS] bit.
3. Wait for OCOTP_CTRL[BUSY] and OCOTP_CTRL[RELOAD_SHADOWS] to be cleared by the controller. BUSY will take on the order of 280ns to clear depending on the values programmed into the OCOTP_TIMING register.

The OCOTP_CTRL[RELOAD_SHADOWS] bit can be set at any time. There is no need to wait for OCOTP_CTRL[BUSY] or OCOTP_CTRL[ERROR] to be clear. In the case of OCOTP_CTRL[BUSY] being set due to an active write, the controller will perform

the bank opening and shadow reloading immediately after the completion of the write. The controller will automatically clear the OCOTP_CTRL[RELOAD_SHADOWS] bit after the successful completion of the operation.

38.3.1.2 Fuse and Shadow Register Writes

In order to avoid “rogue” code performing erroneous writes to OTP, a special unlocking sequence is required for writes to the fuse banks. To write to the fuse banks correctly complete the following steps:

1. Program OCOTP_TIMING[SCLK_COUNT] and OCOTP_TIMING[RELAX] fields with timing values to match the current frequency of the apb_clk. OTP writes will work at maximum bus frequencies as long as the OCOTP_TIMING parameters are set correctly.
2. Set the VDDIO voltage to 2.8 V. The VDDIO voltage is used to program OTP. Incorrect voltage and timing parameter settings will result in the OTP being programmed with incorrect values.
3. Check that OCOTP_CTRL[BUSY] and OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write or reload must be completed before a write access can be requested.
4. Write the requested address to OCOTP_CTRL[ADDR] and program the unlock code into OCOTP_CTRL[WR_UNLOCK]. This must be programmed for each write access. The lock code is documented in the register description. Both the unlock code and address can be written in the same operation.
5. Write the data to the OCOTP_DATA register. This will automatically set OCOTP_CTRL[BUSY] and clear OCOTP_CTRL[WR_UNLOCK]. In this case, the data is a programming mask. Bit fields with 1’s will result in that OTP bit being set. Only the controller can clear OCOTP_CTRL[BUSY]. The controller will use the mask to program a 32-bit word in the OTP per the address in OCOTP_CTRL[ADDR]. At the same time that the write is accepted, the controller makes an internal copy of OCOTP_CTRL[ADDR] which cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to OCOTP_CTRL[ADDR] will not affect an active write operation. It should also be noted that during the programming OCOTP_DATA will shift right (with zero fill). This shifting is required to program the OTP serially. During the write operation, OCOTP_DATA cannot be modified.
6. Once complete, the controller will clear BUSY. A write request to a protected or locked region will result in no OTP access and no setting of OCOTP_CTRL[BUSY]. In addition OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new write access can be issued.

It should be noted that write latencies to OTP are in the order of 100's of micro-seconds per word (with max latency up to 1.2ms). Write latencies will vary based on the location of the word within the OTP bank. Each OTP bank has 8 words, so given n-th word location (where 0 n 7), and 32-bits per word the approximate write latency for a given word is:

$5us \times 32 \times (n + 1)$

Shadow register bits can be overridden by software until the corresponding OCOTP_LOCK[*n*_SHADOW] bit for the region is set. When the lock shadow bit is set, the shadow registers for that lock region become write locked. The exceptions to this are registers CFG5, CFG6, MEM5 and LOCK. The first three registers (CFG5, CFG6, MEM5) are always unlocked and they have no corresponding shadow or fuse lock bits. The latter register, LOCK, also has no shadow or fuse lock bits but it is always read only.

OCOTP_CTRL[ERROR] will be set under the following conditions:

- A write is performed to a shadow register during a shadow reload (essentially, while OCOTP_CTRL[RELOAD_SHADOWS] is set. In addition, the contents of the shadow register shall not be updated.
- A write is performed to a shadow register which has been locked.
- A read is performed from a shadow register which has been read locked.

38.3.1.3 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 ms after the clearing of OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes. A recommended software sequence to meet the postamble requirements is as follows:

- Issue the write and poll for BUSY (as per Fuse Shadow Memory Footprint).
- Once BUSY is clear, use EPIT to wait 2 ms.
- Perform the next OTP operation.

38.3.2 Fuse Shadow Memory Footprint

The OTP memory footprint is shown in Figure 38-2. The registers are grouped by lock region. Their names correspond to the PIO register and fusemap names.

NOTE

Only some banks are pin accessible.

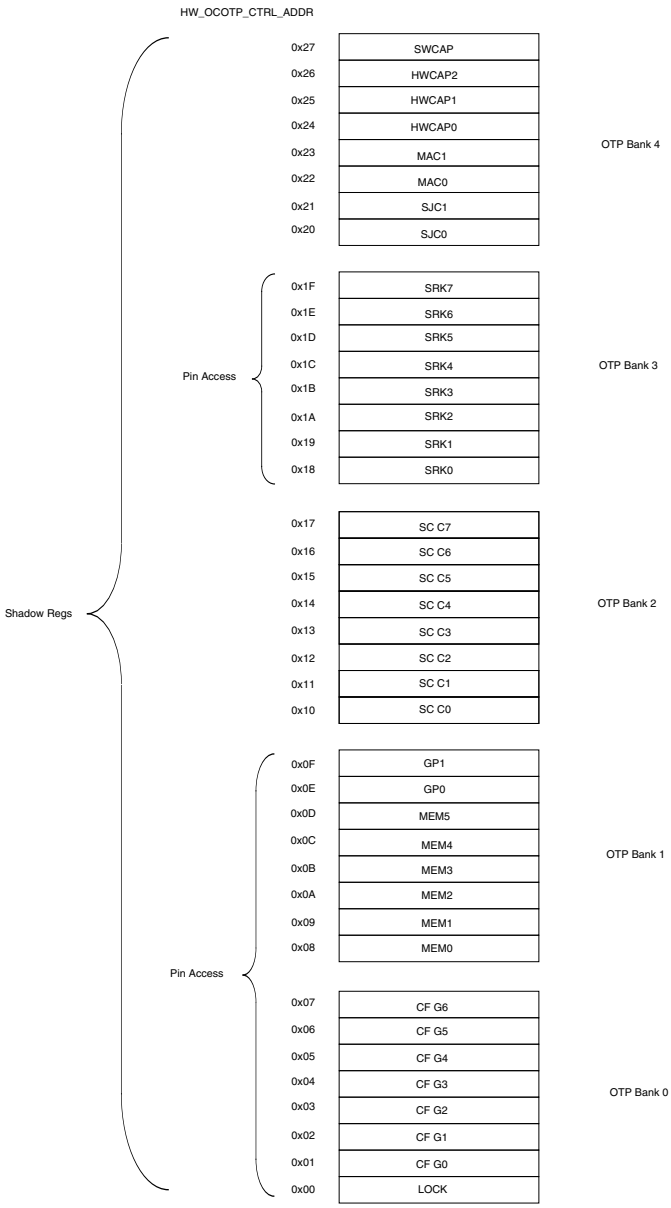


Figure 38-2. OTP Memory Footprint

38.3.3 OTP Read/Write Timing Parameters

There are 3 timing fields contained in the OCOTP_TIMING register that specify counter limit values that are used to time how long the state machine remains in the various states and also specify the SCLK signal timing. They are all specified in apb_clk cycles. Since the apb_clk frequency can be set to a range of values these parameters must be adjusted with the clock to yield the appropriate delay.

The OCOTP_TIMING[RELAX] field specifies how long to remain in the WR_SETUP1, WR_SETUP2, WR_SETUP3, WR_END2, RD_SETUP1 and RD_CLOSE states. It also specifies the SCLK low time (tCKLP) during fuse writes. This parameter should be set by the equation:

tRELAX = tCKLP = (OCOTP_TIMING[RELAX]/apb_frequency) > 10.5 ns

Even though the eFuse specification [eFUSE] specifies that tCKLP can be as short as 5ns it must be programmed to a minimum 10.5ns because the OCOTP_TIMING[RELAX] field is used to create other timing delays in addition to tCLKP. For all timing to be met, this is the minimum delay that must be programmed.

The OCOTP_TIMING[SCLK_COUNT field specifies the period of the SCK signal for fuse writes (during state WR) and is given in units of apb_clk cycles. This value should be specified so that the requirement for the time when the SCLK is asserted high is met: 4000ns < tCKHP < 6000ns is met. Even though a range is given for tCKHP, it is advised in [eFUSE] to program for a value of 5000ns. Therefore, this field should be set according to the equation:

tCKHP = ((OCOTP_TIMING[SCLK_COUNT]/apb_frequency) – tRELAX) = 5000ns.

Figure 38-3 illustrates the relationship between the SCLK signal in programming mode and the timing PIO register fields that affect it. The implementation uses one counter to generate the SCLK waveform within one period and a second counter counts the number of cycles to create for programming the designated word.

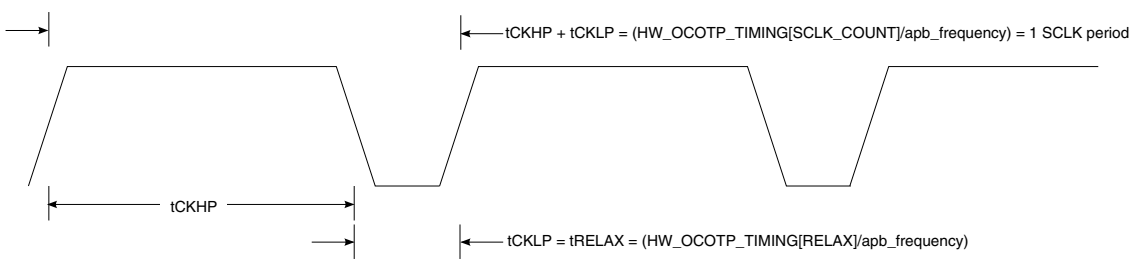


Figure 38-3. SCLK Signal Creation and Timing

The OCOTP_TIMING[RD_BUSY] value gives the time to wait in the RD_BUSY state (wait for banks to be opened for reading) according to the equation:

tRD_BUSY = (OCOTP_TIMING[RD_BUSY]/apb_frequency) > 250ns

38.3.4 Hardware Visible Fuses

The hwv_fuse bus emanates from the OCOTP block and goes to various other blocks inside the chip. This bus is made up of the shadow register bits for banks 0, 1, 2 and 4. Only a subset of these fuse bits are currently used by the hardware. The fuse bits are initially copied from the eFuse banks after reset is deasserted. When all fuse bits are loaded into their shadow registers, the OCOTP asserts the fuse_latched output signal.

The hwv_reg bus also comes from the OCOTP. Its source is the OCOTP_SCS register. For i.MX50 this register has 1 defined bit, the HAB_JDE bit, that is connected to the SJC block. The SCS bits are intended to be used as volatile fuse bits under software control. Additional bits will be defined as needed in future implementations.

The system-wide reset sequence must be coordinated by the system reset controller so that the hwv_fuse and hwv_reg busses are stable and reflect the values of the fuses before they are used by the rest of the system.

The Crypto key which comprises registers SCC0-SCC3 and the Unique key which comprises registers CFG0 and CFG1, can be read by other blocks in the design through two different interfaces.

1. This information can be sent 32 bits at a time using the ocotp_crypto_key_data, ocotp_crpyto_key_smpl, ocotp_unique_key_data and ocotp_unique_key_smpl signals. When the OCOTP first comes out of reset and loads the fuse bits it asserts the “key_smpl” signals for 4 cycles of the apb_clk for the Crypto key and 2 cycles for the Unique key, and synchronizes the data on the “key_data” signals so that the receiving logic can latch the key data 32 bits at a time on the rising edge of apb_clk. The data is sent beginning with the least significant word.
2. After fuse_latched is asserted, the key information is always present in the hwv_fuse bus. The Crypto key makes up bits hwv_fuse[639:512] and and the Unique key is in bits hwv_fuse[95:32].

38.3.5 Behavior During Reset

The OCOTP is always active. The shadow registers automatically load the appropriate OTP contents after reset is deasserted. During this load-time OCOTP_CTRL_BUSY is set. The load time is similar to that of a “reload shadow” operation.

38.4 OCOTP Memory Map/Register Definition

OCOTP Hardware Register Format Summary

OCOTP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_2000	OTP Controller Control Register (OCOTP_CTRL)	32	R/W	0000_0000h	38.4.1/ 2325
4100_2004	OTP Controller Control Register (OCOTP_CTRL_SET)	32	R/W	0000_0000h	38.4.1/ 2325
4100_2008	OTP Controller Control Register (OCOTP_CTRL_CLR)	32	R/W	0000_0000h	38.4.1/ 2325
4100_200C	OTP Controller Control Register (OCOTP_CTRL_TOG)	32	R/W	0000_0000h	38.4.1/ 2325
4100_2010	OTP Controller Timing Register (OCOTP_TIMING)	32	R/W	000C_1079h	38.4.2/ 2327
4100_2020	OTP Controller Write Data Register (OCOTP_DATA)	32	R/W	0000_0000h	38.4.3/ 2328
4100_2030	Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)	32	R	0000_0000h	38.4.4/ 2328
4100_2040	Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0)	32	R/W	0000_0000h	38.4.5/ 2330
4100_2050	Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1)	32	R/W	0000_0000h	38.4.6/ 2331
4100_2060	Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2)	32	R/W	0000_0000h	38.4.7/ 2331
4100_2070	Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3)	32	R/W	0000_0000h	38.4.8/ 2332
4100_2080	Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4)	32	R/W	0000_0000h	38.4.9/ 2332
4100_2090	Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5)	32	R/W	0000_0000h	38.4.10/ 2333
4100_20A0	Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6)	32	R/W	0000_0000h	38.4.11/ 2333
4100_20B0	Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0)	32	R/W	0000_0000h	38.4.12/ 2334
4100_20C0	Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1)	32	R/W	0000_0000h	38.4.13/ 2334
4100_20D0	Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2)	32	R/W	0000_0000h	38.4.14/ 2335

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_20E0	Value of OTP Bank1 Word3 (Memory Related Info.) (OCOTP_MEM3)	32	R/W	0000_0000h	38.4.15/ 2335
4100_20F0	Value of OTP Bank1 Word4 (Memory Related Info.) (OCOTP_MEM4)	32	R/W	0000_0000h	38.4.16/ 2336
4100_2100	Value of OTP Bank1 Word5 (Memory Related Info.) (OCOTP_MEM5)	32	R/W	0000_0000h	38.4.17/ 2336
4100_2110	Value of OTP Bank1 Word6 (General Purpose Customer Defined Info.) (OCOTP_GP0)	32	R/W	0000_0000h	38.4.18/ 2337
4100_2120	Value of OTP Bank1 Word7 (General Purpose Customer Defined Info.) (OCOTP_GP1)	32	R/W	0000_0000h	38.4.19/ 2337
4100_2130	Shadow Register for OTP Bank2 Word0 (SCC and CRYPTO Key) (OCOTP_SCC0)	32	R/W	0000_0000h	38.4.20/ 2338
4100_2140	Shadow Register for OTP Bank2 Word1 (SCC and CRYPTO Key) (OCOTP_SCC1)	32	R/W	0000_0000h	38.4.21/ 2338
4100_2150	Shadow Register for OTP Bank2 Word2 (SCC and CRYPTO Key) (OCOTP_SCC2)	32	R/W	0000_0000h	38.4.22/ 2339
4100_2160	Shadow Register for OTP Bank2 Word3 (SCC and CRYPTO Key) (OCOTP_SCC3)	32	R/W	0000_0000h	38.4.23/ 2339
4100_2170	Shadow Register for OTP Bank2 Word4 (SCC Key) (OCOTP_SCC4)	32	R/W	0000_0000h	38.4.24/ 2340
4100_2180	Shadow Register for OTP Bank2 Word5 (SCC Key) (OCOTP_SCC5)	32	R/W	0000_0000h	38.4.25/ 2340
4100_2190	Shadow Register for OTP Bank2 Word6 (SCC Key) (OCOTP_SCC6)	32	R/W	0000_0000h	38.4.26/ 2341
4100_21A0	Shadow Register for OTP Bank2 Word7 (SCC Key) (OCOTP_SCC7)	32	R/W	0000_0000h	38.4.27/ 2341
4100_21B0	Shadow Register for OTP Bank3 Word0 (SRK Hash) (OCOTP_SRK0)	32	R/W	0000_0000h	38.4.28/ 2342
4100_21C0	Shadow Register for OTP Bank3 Word1 (SRK Hash) (OCOTP_SRK1)	32	R/W	0000_0000h	38.4.29/ 2342
4100_21D0	Shadow Register for OTP Bank3 Word2 (SRK Hash) (OCOTP_SRK2)	32	R/W	0000_0000h	38.4.30/ 2343
4100_21E0	Shadow Register for OTP Bank3 Word3 (SRK Hash) (OCOTP_SRK3)	32	R/W	0000_0000h	38.4.31/ 2343
4100_21F0	Shadow Register for OTP Bank3 Word4 (SRK Hash) (OCOTP_SRK4)	32	R/W	0000_0000h	38.4.32/ 2344
4100_2200	Shadow Register for OTP Bank3 Word5 (SRK Hash) (OCOTP_SRK5)	32	R/W	0000_0000h	38.4.33/ 2344
4100_2210	Shadow Register for OTP Bank3 Word6 (SRK Hash) (OCOTP_SRK6)	32	R/W	0000_0000h	38.4.34/ 2345

Table continues on the next page...

OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_2220	Shadow Register for OTP Bank3 Word7 (SRK Hash) (OCOTP_SRK7)	32	R/W	0000_0000h	38.4.35/ 2345
4100_2230	Value of OTP Bank4 Word0 (Secure JTAG Response Field) (OCOTP_SJC_RESP0)	32	R/W	0000_0000h	38.4.36/ 2346
4100_2240	Value of OTP Bank4 Word1 (Secure JTAG Response Field) (OCOTP_SJC_RESP1)	32	R/W	0000_0000h	38.4.37/ 2346
4100_2250	Value of OTP Bank4 Word2 (MAC Address) (OCOTP_MAC0)	32	R/W	0000_0000h	38.4.38/ 2347
4100_2260	Value of OTP Bank4 Word3 (MAC Address) (OCOTP_MAC1)	32	R/W	0000_0000h	38.4.39/ 2347
4100_2270	Value of OTP Bank4 Word4 (HW Capabilities) (OCOTP_HWCAP0)	32	R/W	0000_0000h	38.4.40/ 2348
4100_2280	Value of OTP Bank4 Word5 (HW Capabilities) (OCOTP_HWCAP1)	32	R/W	0000_0000h	38.4.41/ 2348
4100_2290	Value of OTP Bank4 Word6 (HW Capabilities) (OCOTP_HWCAP2)	32	R/W	0000_0000h	38.4.42/ 2349
4100_22A0	Value of OTP Bank4 Word7 (SW Capabilities) (OCOTP_SWCAP)	32	R/W	0000_0000h	38.4.43/ 2349
4100_22B0	Software Controllable Signals Register (OCOTP_SCS)	32	R/W	0000_0000h	38.4.44/ 2350
4100_22B4	Software Controllable Signals Register (OCOTP_SCS_SET)	32	R/W	0000_0000h	38.4.44/ 2350
4100_22B8	Software Controllable Signals Register (OCOTP_SCS_CLR)	32	R/W	0000_0000h	38.4.44/ 2350
4100_22BC	Software Controllable Signals Register (OCOTP_SCS_TOG)	32	R/W	0000_0000h	38.4.44/ 2350
4100_22C0	OTP Controller Version Register (OCOTP_VERSION)	32	R	0106_0000h	38.4.45/ 2350

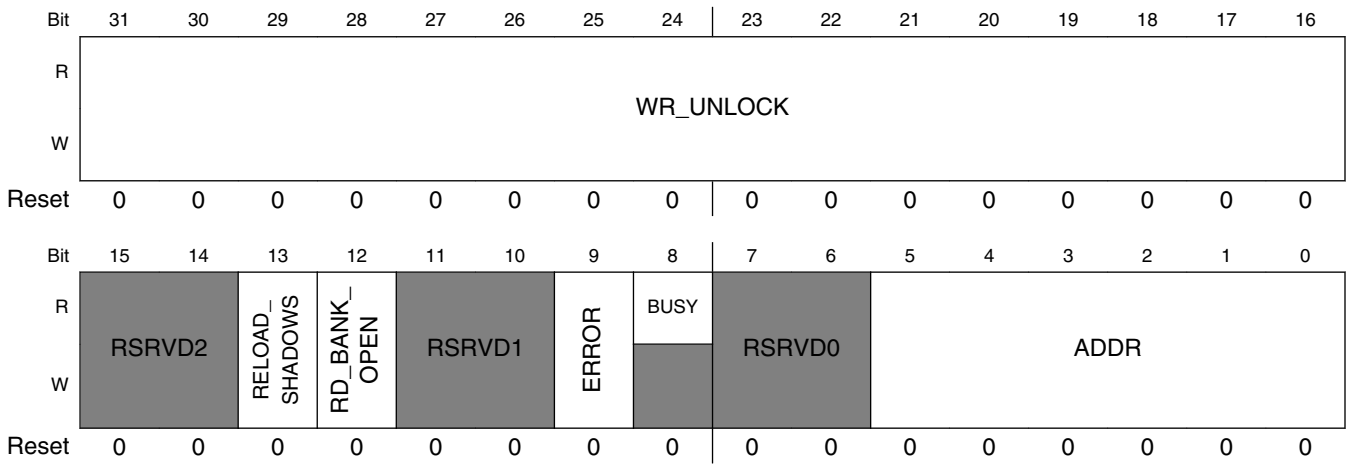
38.4.1 OTP Controller Control Register (OCOTP_CTRL*n*)

The OCOTP Control and Status Register specifies the copy state, as well as the control required for random access of the OTP memory

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjuction with the OCOTP_DATA register to perform write operations. Read operations are performed via the direct memory mapped registers. In the cases where OTP values are shadowed into local memory storage, the memory mapped location can

be read directly. In the cases where the OTP values are not shadowed into local memory, the read-preparation sequence involving RD_BANK_OPEN and BUSY/ERROR fields must be used before performing the read.

Addresses: OCOTP_CTRL is 4100_2000h base + 0h offset = 4100_2000h
OCOTP_CTRL_SET is 4100_2000h base + 4h offset = 4100_2004h
OCOTP_CTRL_CLR is 4100_2000h base + 8h offset = 4100_2008h
OCOTP_CTRL_TOG is 4100_2000h base + Ch offset = 4100_200Ch



OCOTP_CTRL*n* field descriptions

Field	Description
31–16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY). 0x3E77 **KEY** — Key needed to unlock OCOTP_DATA register.
15–14 RSRVD2	These bits always read back zero.
13 RELOAD_ SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically open banks (but will not set RD_BANK_OPEN) and set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller. There is no need to set RD_BANK_OPEN to force the reload. If RD_BANK_OPEN is already set, its still possible to set RELOAD_SHADOWS. In this case, the shadow registers will only be updated upon the clearing of RD_BANK_OPEN.
12 RD_BANK_ OPEN	Set to open the all the OTP banks for reading. When set, the controller sets BUSY to allow time for the banks to become available (approximatly 32 HCLK cycles later at which time the controller will clear BUSY). Once BUSY is clear, the various OTP words are accessible via their memory mapped address. Note that OTP words which are shadowed, can be read at anytime and will not be affected by RD_BANK_OPEN. This bit must be cleared after reading is complete. Keeping the OTP banks open causes additional current draw. BUSY must be clear before this setting will take affect. If there is a write transaction pending (holding BUSY), then the bank opening sequence will begin automatically upon the previous transaction clears BUSY. Note that if a read is performed from non-shadowed locations without RD_BANK_OPEN, ERROR will be set
11–10 RSRVD1	These bits always read back zero.

Table continues on the next page...

OCOTP_CTRL*n* field descriptions (continued)

Field	Description
9 ERROR	Set by the controller when either an access to a locked region is requested or a read is requested from non-shadowed efuse locations without the banks being open. Must be cleared before any further write access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or bank open operations (including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete (for writes), or the banks are open (for reads). After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7–6 RSRVD0	These bits always read back zero.
5–0 ADDR	OTP write access address register. Specifies one of 40 word address locations (0x00 - 0x27). If a valid write is accepted by the controller (see OCOTP_DATA for details on what constitutes a valid write), the controller makes an internal copy of this value (to avoid the OTP programming being corrupted). This internal copy will not update until the write access is complete.

38.4.2 OTP Controller Timing Register (OCOTP_TIMING)

The OCOTP Data Register is used for OTP Programming

This register specifies timing parameters for programming and reading the OCOTP fuse array.

Address: OCOTP_TIMING is 4100_2000h base + 10h offset = 4100_2010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSRVD0										RD_BUSY						RELAX				SCLK_COUNT											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	1

OCOTP_TIMING field descriptions

Field	Description
31–22 RSRVD0	These bits always read back zero.
21–16 RD_BUSY	This count value specifies the time to wait while the banks are busy after requesting a read.
15–12 RELAX	This count value specifies the time to add to all default timing parameters other than the Tckhp. It is given in number of apb_clk periods.
11–0 SCLK_COUNT	This count value specifies the sclk high time. It is given in number of apb_clk periods. Valid field values are greater 15.

38.4.3 OTP Controller Write Data Register (OCOTP_DATA)

The OCOTP Data Register is used for OTP Programming

This register is used in conjuction with OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

Address: OCOTP_DATA is 4100_2000h base + 20h offset = 4100_2020h

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

OCOTP_DATA field descriptions

Field	Description
31–0 DATA	Used to initiate a write to OTP. Please see the "Software Write Sequence" section for operating details.

38.4.4 Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)

Shadowed memory mapped access to OTP Bank 0, word 0 (ADDR = 0x00).

Address: OCOTP_LOCK is 4100_2000h base + 30h offset = 4100_2030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNALLOCATED						PIN	DCPKEY_ALT	SCC_ALT	DCPKEY	SWCAP_SHADOW	SWCAP	HWCAP_SHADOW	HWCAP	MAC_SHADOW	MAC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SJC_SHADOW	SJC	SRK_SHADOW	SRK	SCC_SHADOW	SCC	GP_SHADOW	GP	MEM_MISC_SHADOW	MEM_TRIM_SHADOW	MEM_TRIM	CFG_MISC_SHADOW	CFG_BOOT_SHADOW	CFG_BOOT	CFG_TESTER_SHADOW	CFG_TESTER
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OCOTP_LOCK field descriptions

Field	Description
31–26 UNALLOCATED	Value of un-used portion of LOCK word
25 PIN	Status of Pin access lock bit. When set, pin access is disabled.
24 DCPKEY_ALT	Status of alternate bit for DCPKEY lock
23 SCC_ALT	Status of alternate bit for SCC lock
22 DCPKEY	Status of read lock bit for DCP APB crypto key access. When set, the DCP will disallow reads of its crypto keys via its APB interface.
21 SWCAP_ SHADOW	Status of shadow register lock for the region contained in the SWCAP registers. When set, the over-riding (writing) of this region's shadow bits is blocked.
20 SWCAP	Status of register lock for the fuse region contained in the SWCAP registers. When set, these fuse bits are write blocked.
19 HWCAP_ SHADOW	Status of shadow register lock for the region contained in the HWCAP registers. When set, the over-riding (writing) of this region's shadow bits is blocked.
18 HWCAP	Status of register lock for the fuse region contained in the CRYPTO registers. When set, these fuse bits are write blocked.
17 MAC_SHADOW	Status of shadow register lock for the region contained in the MAC registers. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are always readable.
16 MAC	Status of register lock for the fuse region contained in the MAC registers. When set, these fuse bits are write blocked.
15 SJC_SHADOW	Status of shadow register lock for the fuse region contained in the SJC (sjc_response) registers. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are readable as long as the LOCK_SJC bit is not set.
14 SJC	Status of register lock for the region contained in the SJC (sjc_response) registers. When set, these fuse bits are write blocked. Also, their shadow registers are read blocked.
13 SRK_SHADOW	Status of shadow register lock for the region contained in the SRK registers. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are always readable.
12 SRK	Status of register lock for the fuse region contained in the SRK registers. When set, these fuse bits are write blocked.
11 SCC_SHADOW	Status of shadow register lock for the region contained in the SCC registers. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are readable as long as the LOCK_SCC bit is not set.
10 SCC	Status of register lock for the fuse region contained in the SCC registers. When set, these fuse bits are write blocked. Also, their shadow registers are read blocked.
9 GP_SHADOW	Status of shadow register lock for the region contained in the GP (General Purpose) registers. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are always readable.
8 GP	Status of register lock for the fuse region contained in the GP (General Purpose). When set, these fuse bits are write blocked.

Table continues on the next page...

OCOTP_LOCK field descriptions (continued)

Field	Description
7 MEM_MISC_ SHADOW	Status of shadow register lock for the region contained in the MEM registers 5-6. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are always readable.
6 MEM_TRIM_ SHADOW	Status of shadow register lock for the region contained in the MEM registers 0-4. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are always readable.
5 MEM_TRIM	Status of register lock for the fuse region contained in the MEM registers 0-4. When set, these fuse bits are write blocked.
4 CFG_MISC_ SHADOW	Status of shadow register lock for the region contained in the CFG registers 5-6. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are always readable.
3 CFG_BOOT_ SHADOW	Status of shadow register lock for the region contained in the CFG registers 3-4. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are always readable.
2 CFG_BOOT	Status of register lock for the fuse region contained in the CFG registers 3-4. When set, these fuse bits are write blocked.
1 CFG_TESTER_ SHADOW	Status of shadow register lock for the region contained in the CFG registers 0-2. When set, the over-riding (writing) of this region's shadow bits is blocked. These shadow registers are always readable.
0 CFG_TESTER	Status of register lock for the fuse region contained in the CFG registers 0-2. When set, these fuse bits are write blocked.

38.4.5 Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 1 (ADDR = 0x01).

Address: OCOTP_CFG0 is 4100_2000h base + 40h offset = 4100_2040h

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

OCOTP_CFG0 field descriptions

Field	Description
31–0 BITS	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 1 (ADDR = 0x01). These bits become read-only after the OCOTP_LOCK[CFG_SHADOW] bit is set.

38.4.6 Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

shadowed memory mapped access to OTP Bank 0, word 2 (ADDR = 0x02).

Address: OCOTP_CFG1 is 4100_2000h base + 50h offset = 4100_2050h

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

OCOTP_CFG1 field descriptions

Field	Description
31–0 BITS	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 2 (ADDR = 0x02). These bits become read-only after the OCOTP_LOCK[CFG_SHADOW] bit is set.

38.4.7 Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 3 (ADDR = 0x03).

Address: OCOTP_CFG2 is 4100_2000h base + 60h offset = 4100_2060h

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

OCOTP_CFG2 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 3 (ADDR = 0x03). These bits become read-only after the OCOTP_LOCK[CFG_SHADOW] bit is set.

38.4.8 Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Non-shadowed memory mapped access to OTP Bank 0, word 4 (ADDR = 0x04).

Address: OCOTP_CFG3 is 4100_2000h base + 70h offset = 4100_2070h

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

OCOTP_CFG3 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 4 (ADDR = 0x04). These bits become read-only after the OCOTP_LOCK[CFG_SHADOW] bit is set.

38.4.9 Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 5 (ADDR = 0x05).

Address: OCOTP_CFG4 is 4100_2000h base + 80h offset = 4100_2080h

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

OCOTP_CFG4 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 5 (ADDR = 0x05). These bits become read-only after the OCOTP_LOCK[CFG_SHADOW] bit is set.

38.4.10 Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 6 (ADDR = 0x06).

Address: OCOTP_CFG5 is 4100_2000h base + 90h offset = 4100_2090h

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

OCOTP_CFG5 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 6 (ADDR = 0x06). These bits become read-only after the OCOTP_LOCK[CFG_SHADOW] bit is set.

38.4.11 Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

Address: OCOTP_CFG6 is 4100_2000h base + A0h offset = 4100_20A0h

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

OCOTP_CFG6 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 7 (ADDR = 0x07). These bits become read-only after the OCOTP_LOCK[CFG_SHADOW] bit is set.

38.4.12 Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 0 (ADDR = 0x08).

Address: OCOTP_MEM0 is 4100_2000h base + B0h offset = 4100_20B0h

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

OCOTP_MEM0 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 31:0 (copy of OTP bank 1, word 0 (ADDR = 0x08)). These bits become read-only after the OCOTP_LOCK[HWSW_SHADOW] or OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

38.4.13 Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 1 (ADDR = 0x09).

Address: OCOTP_MEM1 is 4100_2000h base + C0h offset = 4100_20C0h

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

OCOTP_MEM1 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 63:32 (copy of OTP bank 1, word 1 (ADDR = 0x09)). These bits become read-only after the OCOTP_LOCK[HWSW_SHADOW] or OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

38.4.14 Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 2 (ADDR = 0x0A).

Address: OCOTP_MEM2 is 4100_2000h base + D0h offset = 4100_20D0h

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

OCOTP_MEM2 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 95:64 (copy of OTP bank 1, word 2 (ADDR = 0x0A)). These bits become read-only after the OCOTP_LOCK[HWSW_SHADOW] or OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

38.4.15 Value of OTP Bank1 Word3 (Memory Related Info.) (OCOTP_MEM3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 3 (ADDR = 0x0B).

Address: OCOTP_MEM3 is 4100_2000h base + E0h offset = 4100_20E0h

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

OCOTP_MEM3 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 127:96 (copy of OTP bank 1, word 3 (ADDR = 0x0B)). These bits become read-only after the OCOTP_LOCK[HWSW_SHADOW] or OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

38.4.16 Value of OTP Bank1 Word4 (Memory Related Info.) (OCOTP_MEM4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 4 (ADDR = 0x0C).

Address: OCOTP_MEM4 is 4100_2000h base + F0h offset = 4100_20F0h

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

OCOTP_MEM4 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 159:128 (copy of OTP bank 1, word 4 (ADDR = 0x0C)). These bits become read-only after the OCOTP_LOCK[HWSW_SHADOW] or OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

38.4.17 Value of OTP Bank1 Word5 (Memory Related Info.) (OCOTP_MEM5)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 5 (ADDR = 0x0D).

Address: OCOTP_MEM5 is 4100_2000h base + 100h offset = 4100_2100h

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

OCOTP_MEM5 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 191:160 (copy of OTP bank 1, word 5 (ADDR = 0x0D)). These bits become read-only after the OCOTP_LOCK[HWSW_SHADOW] or OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

38.4.18 Value of OTP Bank1 Word6 (General Purpose Customer Defined Info.) (OCOTP_GP0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 6 (ADDR = 0x0E).

Address: OCOTP_GP0 is 4100_2000h base + 110h offset = 4100_2110h

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

OCOTP_GP0 field descriptions

Field	Description
31–0 BITS	Shadow register for SW capability bits 31:0 (copy of OTP bank 1, word 6 (ADDR = 0x0E)). These bits become read-only after the OCOTP_LOCK[HWSW_SHADOW] or OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

38.4.19 Value of OTP Bank1 Word7 (General Purpose Customer Defined Info.) (OCOTP_GP1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 7 (ADDR = 0x0F).

Address: OCOTP_GP1 is 4100_2000h base + 120h offset = 4100_2120h

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

OCOTP_GP1 field descriptions

Field	Description
31–0 BITS	Shadow register for SW capability bits 31:0 (copy of OTP bank 1, word 6 (ADDR = 0x0E)). These bits become read-only after the OCOTP_LOCK[HWSW_SHADOW] or OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

38.4.20 Shadow Register for OTP Bank2 Word0 (SCC and CRYPTO Key) (OCOTP_SCC0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 2, word 0 (ADDR = 0x10).

Address: OCOTP_SCC0 is 4100_2000h base + 130h offset = 4100_2130h

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

OCOTP_SCC0 field descriptions

Field	Description
31–0 BITS	Shadow register for the SCC and CRYPTO Key word0 (Copy of OTP Bank 2, word 0 (ADDR = 0x14)). These bits become read-only after the OCOTP_LOCK[SCC_SHADOW] bit is set. If OCOTP_LOCK[SCC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR].

38.4.21 Shadow Register for OTP Bank2 Word1 (SCC and CRYPTO Key) (OCOTP_SCC1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 2, word 1 (ADDR = 0x11).

Address: OCOTP_SCC1 is 4100_2000h base + 140h offset = 4100_2140h

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

OCOTP_SCC1 field descriptions

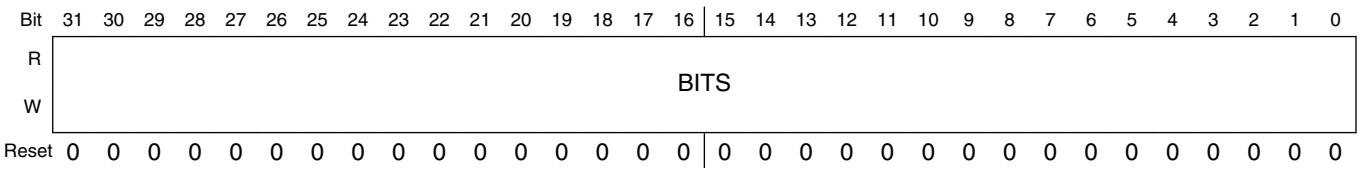
Field	Description
31–0 BITS	Shadow register for the SCC and CRYPTO Key word1 (Copy of OTP Bank 2, word 1 (ADDR = 0x15)). These bits become read-only after the OCOTP_LOCK[SCC_SHADOW] bit is set. If OCOTP_LOCK[SCC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR].

38.4.22 Shadow Register for OTP Bank2 Word2 (SCC and CRYPTO Key) (OCOTP_SCC2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 2, word 2 (ADDR = 0x12).

Address: OCOTP_SCC2 is 4100_2000h base + 150h offset = 4100_2150h



OCOTP_SCC2 field descriptions

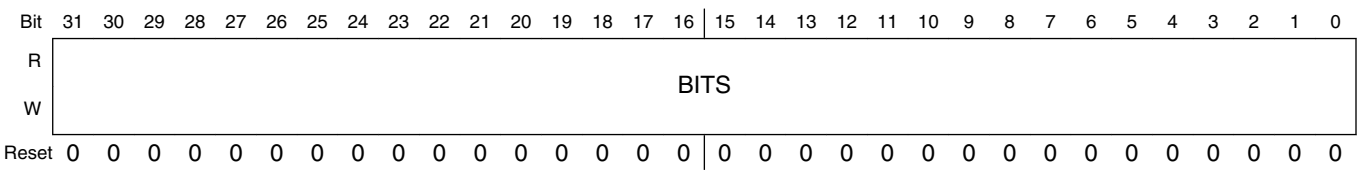
Field	Description
31–0 BITS	Shadow register for the SCC and CRYPTO Key word2 (Copy of OTP Bank 2, word 2 (ADDR = 0x16)). These bits become read-only after the OCOTP_LOCK[SCC_SHADOW] bit is set. If OCOTP_LOCK[SCC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR].

38.4.23 Shadow Register for OTP Bank2 Word3 (SCC and CRYPTO Key) (OCOTP_SCC3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 2, word 3 (ADDR = 0x13).

Address: OCOTP_SCC3 is 4100_2000h base + 160h offset = 4100_2160h



OCOTP_SCC3 field descriptions

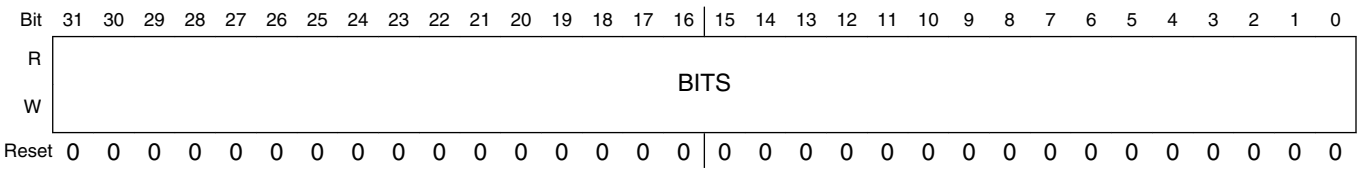
Field	Description
31–0 BITS	Shadow register for the SCC and CRYPTO Key word3 (Copy of OTP Bank 2, word 3 (ADDR = 0x17)). These bits become read-only after the OCOTP_LOCK[SCC_SHADOW] bit is set. If OCOTP_LOCK[SCC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR].

38.4.24 Shadow Register for OTP Bank2 Word4 (SCC Key) (OCOTP_SCC4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 2, word 4 (ADDR = 0x14).

Address: OCOTP_SCC4 is 4100_2000h base + 170h offset = 4100_2170h



OCOTP_SCC4 field descriptions

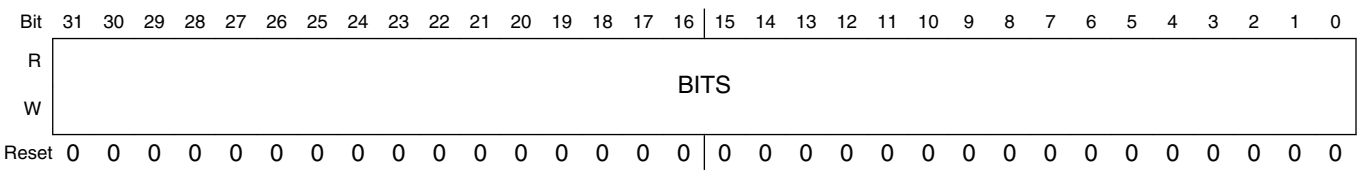
Field	Description
31–0 BITS	Shadow register for the SCC Key word4 (Copy of OTP Bank 2, word 4 (ADDR = 0x18)). These bits become read-only after the OCOTP_LOCK[SCC_SHADOW] bit is set. If OCOTP_LOCK[SCC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR].

38.4.25 Shadow Register for OTP Bank2 Word5 (SCC Key) (OCOTP_SCC5)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 2, word 5 (ADDR = 0x15).

Address: OCOTP_SCC5 is 4100_2000h base + 180h offset = 4100_2180h



OCOTP_SCC5 field descriptions

Field	Description
31–0 BITS	Shadow register for the SCC Key word5 (Copy of OTP Bank 2, word 5 (ADDR = 0x19)). These bits become read-only after the OCOTP_LOCK[SCC_SHADOW] bit is set. If OCOTP_LOCK[SCC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR].

38.4.26 Shadow Register for OTP Bank2 Word6 (SCC Key) (OCOTP_SCC6)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 2, word 6 (ADDR = 0x16).

Address: OCOTP_SCC6 is 4100_2000h base + 190h offset = 4100_2190h

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

OCOTP_SCC6 field descriptions

Field	Description
31–0 BITS	Shadow register for the SCC Key word6 (Copy of OTP Bank 2, word 6 (ADDR = 0x1A)). These bits become read-only after the OCOTP_LOCK[SCC_SHADOW] bit is set. If OCOTP_LOCK[SCC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR].

38.4.27 Shadow Register for OTP Bank2 Word7 (SCC Key) (OCOTP_SCC7)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 2, word 7 (ADDR = 0x17).

Address: OCOTP_SCC7 is 4100_2000h base + 1A0h offset = 4100_21A0h

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

OCOTP_SCC7 field descriptions

Field	Description
31–0 BITS	Shadow register for the SCC key word7 (Copy of OTP Bank 2, word 7 (ADDR = 0x1C)). These shadow bits become read-only after the OCOTP_LOCK[SCC_SHADOW] bit is set. If OCOTP_LOCK[SCC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR].

38.4.28 Shadow Register for OTP Bank3 Word0 (SRK Hash) (OCOTP_SRK0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 0 (ADDR = 0x18).

Address: OCOTP_SRK0 is 4100_2000h base + 1B0h offset = 4100_21B0h

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

OCOTP_SRK0 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word0 (Copy of OTP Bank 3, word 0 (ADDR = 0x1C)). These bits become read-only after the OCOTP_LOCK[SRK_SHADOW] bit is set.

38.4.29 Shadow Register for OTP Bank3 Word1 (SRK Hash) (OCOTP_SRK1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 1 (ADDR = 0x19).

Address: OCOTP_SRK1 is 4100_2000h base + 1C0h offset = 4100_21C0h

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

OCOTP_SRK1 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word1 (Copy of OTP Bank 3, word 1 (ADDR = 0x1D)). These bits become read-only after the OCOTP_LOCK[SRK_SHADOW] bit is set.

38.4.30 Shadow Register for OTP Bank3 Word2 (SRK Hash) (OCOTP_SRK2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 2 (ADDR = 0x1A).

Address: OCOTP_SRK2 is 4100_2000h base + 1D0h offset = 4100_21D0h

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

OCOTP_SRK2 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word2 (Copy of OTP Bank 3, word 2 (ADDR = 0x1E)). These bits become read-only after the OCOTP_LOCK[SRK_SHADOW] bit is set.

38.4.31 Shadow Register for OTP Bank3 Word3 (SRK Hash) (OCOTP_SRK3)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 3 (ADDR = 0x1B).

Address: OCOTP_SRK3 is 4100_2000h base + 1E0h offset = 4100_21E0h

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

OCOTP_SRK3 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word3 (Copy of OTP Bank 3, word 3 (ADDR = 0x1F)). These bits become read-only after the OCOTP_LOCK[SRK_SHADOW] bit is set.

38.4.32 Shadow Register for OTP Bank3 Word4 (SRK Hash) (OCOTP_SRK4)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 4 (ADDR = 0x1C).

Address: OCOTP_SRK4 is 4100_2000h base + 1F0h offset = 4100_21F0h

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

OCOTP_SRK4 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word4 (Copy of OTP Bank 3, word 4 (ADDR = 0x20)). These bits become read-only after the OCOTP_LOCK[SRK_SHADOW] bit is set.

38.4.33 Shadow Register for OTP Bank3 Word5 (SRK Hash) (OCOTP_SRK5)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 5 (ADDR = 0x1D).

Address: OCOTP_SRK5 is 4100_2000h base + 200h offset = 4100_2200h

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

OCOTP_SRK5 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word5 (Copy of OTP Bank 3, word 5 (ADDR = 0x21)). These bits become read-only after the OCOTP_LOCK[SRK_SHADOW] bit is set.

38.4.34 Shadow Register for OTP Bank3 Word6 (SRK Hash) (OCOTP_SRK6)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 6 (ADDR = 0x1E).

Address: OCOTP_SRK6 is 4100_2000h base + 210h offset = 4100_2210h

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

OCOTP_SRK6 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word6 (Copy of OTP Bank 3, word 6 (ADDR = 0x22)). These bits become read-only after the OCOTP_LOCK[SRK_SHADOW] bit is set.

38.4.35 Shadow Register for OTP Bank3 Word7 (SRK Hash) (OCOTP_SRK7)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 7 (ADDR = 0x1F).

Address: OCOTP_SRK7 is 4100_2000h base + 220h offset = 4100_2220h

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

OCOTP_SRK7 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word7 (Copy of OTP Bank 3, word 7 (ADDR = 0x23)). These bits become read-only after the OCOTP_LOCK[SRK_SHADOW] bit is set.

38.4.36 Value of OTP Bank4 Word0 (Secure JTAG Response Field) (OCOTP_SJC_RESP0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 0 (ADDR = 0x20).

Address: OCOTP_SJC_RESP0 is 4100_2000h base + 230h offset = 4100_2230h

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

OCOTP_SJC_RESP0 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 5, word 0 (ADDR = 0x24). If LOCK[SJC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR]

38.4.37 Value of OTP Bank4 Word1 (Secure JTAG Response Field) (OCOTP_SJC_RESP1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 1 (ADDR = 0x21).

Address: OCOTP_SJC_RESP1 is 4100_2000h base + 240h offset = 4100_2240h

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

OCOTP_SJC_RESP1 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 4, word 1 (ADDR = 0x25). If LOCK[SJC] is set, returns 0xBADA_BADA and sets OCOTP_CTRL[ERROR]

38.4.38 Value of OTP Bank4 Word2 (MAC Address) (OCOTP_MAC0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 2 (ADDR = 0x22).

Address: OCOTP_MAC0 is 4100_2000h base + 250h offset = 4100_2250h

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

OCOTP_MAC0 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 4, word 2 (ADDR = 0x26).

38.4.39 Value of OTP Bank4 Word3 (MAC Address) (OCOTP_MAC1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 3 (ADDR = 0x23).

Address: OCOTP_MAC1 is 4100_2000h base + 260h offset = 4100_2260h

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

OCOTP_MAC1 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 4, word 3 (ADDR = 0x27).

38.4.40 Value of OTP Bank4 Word4 (HW Capabilities) (OCOTP_HWCAP0)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 4 (ADDR = 0x24).

Address: OCOTP_HWCAP0 is 4100_2000h base + 270h offset = 4100_2270h

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

OCOTP_HWCAP0 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 4, word 4 (ADDR = 0x24).

38.4.41 Value of OTP Bank4 Word5 (HW Capabilities) (OCOTP_HWCAP1)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 5 (ADDR = 0x25).

Address: OCOTP_HWCAP1 is 4100_2000h base + 280h offset = 4100_2280h

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

OCOTP_HWCAP1 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 4, word 5 (ADDR = 0x25).

38.4.42 Value of OTP Bank4 Word6 (HW Capabilities) (OCOTP_HWCAP2)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 6 (ADDR = 0x26).

Address: OCOTP_HWCAP2 is 4100_2000h base + 290h offset = 4100_2290h

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

OCOTP_HWCAP2 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 4, word 6 (ADDR = 0x26).

38.4.43 Value of OTP Bank4 Word7 (SW Capabilities) (OCOTP_SWCAP)

Copied from the OTP automatically after reset. Can be re-loaded by setting OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP Bank 4, word 7 (ADDR = 0x27).

Address: OCOTP_SWCAP is 4100_2000h base + 2A0h offset = 4100_22A0h

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

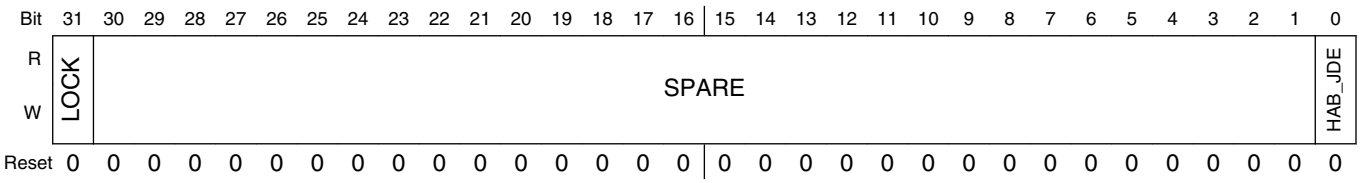
OCOTP_SWCAP field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 4, word 7 (ADDR = 0x27).

38.4.44 Software Controllable Signals Register (OCOTP_SCS*n*)

This register holds volatile configuration values that can be set and locked by software. All values are returned to their defualt values after POR.

Addresses: OCOTP_SCS is 4100_2000h base + 2B0h offset = 4100_22B0h
OCOTP_SCS_SET is 4100_2000h base + 2B4h offset = 4100_22B4h
OCOTP_SCS_CLR is 4100_2000h base + 2B8h offset = 4100_22B8h
OCOTP_SCS_TOG is 4100_2000h base + 2BCh offset = 4100_22BCh



OCOTP_SCS*n* field descriptions

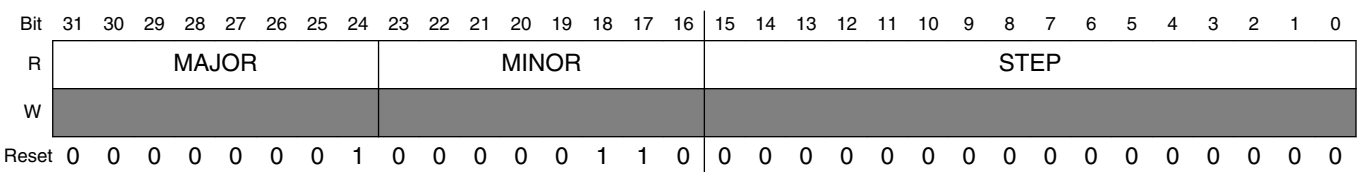
Field	Description
31 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30–1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

38.4.45 OTP Controller Version Register (OCOTP_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

Address: OCOTP_VERSION is 4100_2000h base + 2C0h offset = 4100_22C0h



OCOTP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 39
On-Chip RAM (OCRAM)

39.1 Overview

The on-chip RAM block is implementd as an slave module on the 64-bit system AXI bus. Designed as a simple on-chip memory block, it has only one bank of single port SRAM and supports only one AXI port. For the AXI port, the read and write transactions are handled by two independent modules. As it is possible to have simultaneous read and write request from the AXI bus, an arbiter with round-robin scheme is implemented to handle this. After arbitration, the granted read or write access command can then be issued to the memory cell through a read/write MUX.

A couple of options are provided to adding pipeline or wait-state in read/write access, in order to ensures flexible timing control at high/low frequency working conditions.

The internal block are shown in Figure 39-1.

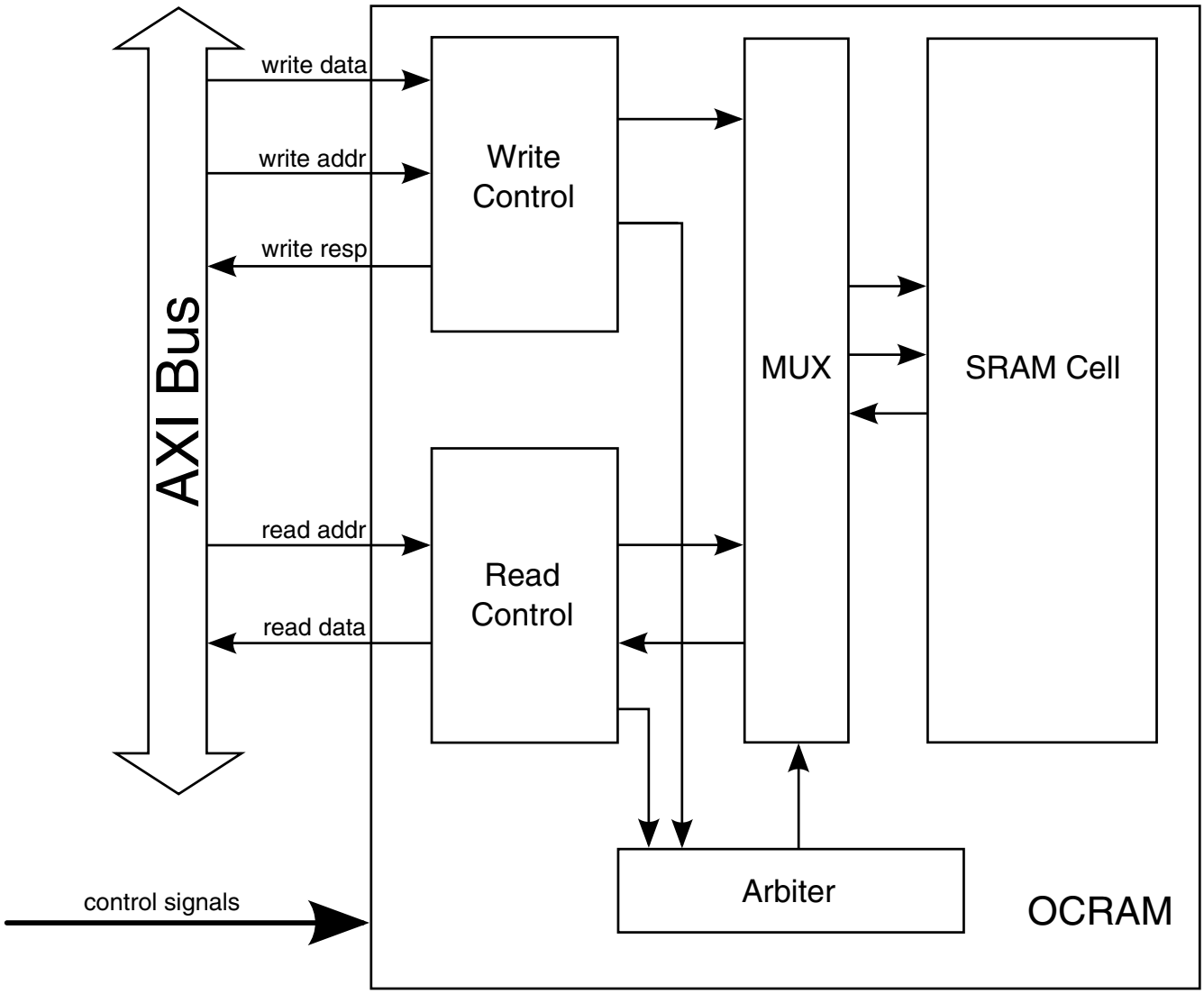


Figure 39-1. On-chip RAM Block Diagram

39.1.1 Memory Map

The on-chip RAM size for i.MX50 is 128 Kbytes, organized as 16K x 64 bits, mapped from 0 x 0000_0000 to 0x0001_FFFF.

Memory Alias is also supported for on-chip RAM. Address from 0 x 0002_0000 to 0 x 3FFF_FFFF are alias addresses for it, any read/write operation to this area will be mapped back to the 128 Kbyte area.

The 32-bit AXI address are arranged as below:

Table 39-1. On-Chip RAM Address Bits

AHB ADDR Bits	USAGE	DESCRIPTION
29:3	Address	Selects one of the 16K words in the memory (64-bit a word).
2:0	Byte Address	Selects/masks out specific bytes within a word.

39.1.2 Read/Write Arbitration

Generally, the arbitration is using the round-robin method. The detailed rules used in arbitration can be described as follows:

- If there is no granted read or write in the last cycle, and there is only read request or write request, the request will be granted.
- If there is no granted read or write in the last cycle, and there are both read or write requests coming at the same time, the read request will be granted first.
- If a granted read/write transaction has just finished, the write/read request will have the higher priority in next cycle.
- If the first read/write access request in a transaction is granted, all the data transfer in this burst will be finished before next arbitration begins, i.e., the round-robin arbitration mechanism is based on AXI transaction, not based on data access.

39.2 Advanced Features

Below are some advanced features designed to avoid timing issues when the on-chip RAM is working at high frequency. All of them can be disabled/enabled by set/clear the corresponding fields of the control register for on-chip RAM. These control registers are implemented in the DIGCTL block withing the SoC. Refer to the chapter entitled "Digital Control (DIGCTL) and On-Chip RAM" for detailed bit definitions.

39.2.1 Read Data Wait State

When the wait state is enabled, it will cost 2 cycles for each read access, (each beat of a read burst). This can avoid the potential timing problem caused by the relatively longer memory access time at higher frequency.

When this feature is disabled, it only costs 1 clock cycle to finish a read transaction, i.e., get read data back in the next cycle of read request becomes valid on the bus.

39.2.2 Read Address Pipeline

When this feature is enabled, the read address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the read access on the memory cell at high frequency. Enable this feature would cost at most 1 more clock cycle for each AXI read transaction, that is, at most 1 more clock cycle for each read burst with multiple beats of data.

When this feature is disabled, the read address from the AXI master can be accepted by the on-chip RAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait).

39.2.3 Write Data Pipeline

When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enable this feature would cost at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write data from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle).

39.2.4 Write Address Pipeline

When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enable this feature would cost at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write address from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle).

39.3 Programmable Registers

There are no programmable registers in this block.

Chapter 40
1-Wire Block (OWIRE)

40.1 Overview

The OWIRE provides the communication link to a generic 1-Kbit add-only memory. The block sends or receives one bit at a time with an option for software to manage the data using bytes. The required protocol for accessing the generic OWIRE device is defined by Maxim-Dallas. The generic OWIRE device holds battery characteristics information.

The following figure shows a block diagram of the OWIRE.

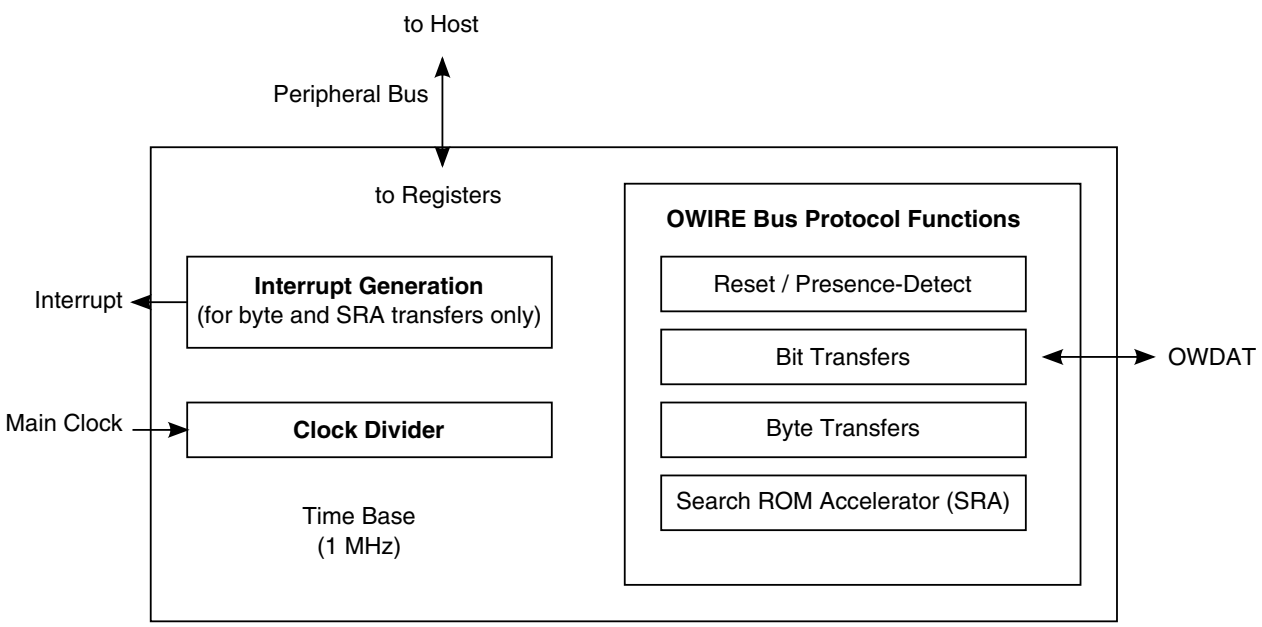


Figure 40-1. OWIRE Block Diagram

40.1.1 Features

The OWIRE includes the following features:

- Performs the OWIRE bus protocol to communicate with an external OWIRE device.
- Provides a clock divider to a generate a OWIRE bus reference clock (derived from the main clock provided internally to the block).
- Supports byte transfers with optional interrupts for more efficient programming.
- Provides a search ROM accelerator mode to speed the search ROM protocol.

40.1.2 Modes of Operation

The OWIRE supports the following operations:

- Normal Operating Modes (See Normal Operating Modes.)
  - Bit or Byte Transfers
  - Reset/Presence-detect Pulse
  - Search ROM Accelerator Mode
- Low Power Mode (See Low Power Mode.)

40.2 External Signals

Table 40-1 shows the signal that interfaces with a generic OWIRE device.

Table 40-1. OWIRE Block Signal

Signal	I/O	Function
OWDAT	I/O	One-Wire bus Requires an external pull-up resistor. The recommended resistor value is specified by the generic OWIRE device used in a given system.

40.3 Functional Description

The OWIRE interfaces with a generic 1-Kbit add-only memory, through a simple 1-bit bus. Software uses the OWIRE bus to program and read the 1-Kbyte memory.

The protocol involves first issuing one of four ROM function commands before the EPROM is accessible:

- Read ROM
- Match ROM
- Search ROM
- Skip ROM

Through the OWIRE bus, the host software interfaces with the generic OWIRE device and allows the required commands to be issued to control the EPROM of a generic OWIRE device. The host (through the OWIRE interface) is the bus master, and the generic OWIRE device(s) are the slave(s)

40.3.1 Normal Operating Modes

The OWIRE supports the following bus protocol functions:

- Reset/Presence-detect Pulse using the Control register (See Reset/Presence-detect Pulse.)
- Bit Transfers using the Control register (See Bit Transfers .)
- Byte Transfers using the TX/RX register (See Byte Transfers.)
- Search ROM Accelerator Mode using the Command register and the TX/RX register (See Search ROM Accelerator Mode.)

40.3.1.1 Reset/Presence-detect Pulse

The OWIRE provides for an automated initialization sequence for the OWIRE bus. Software initiates the initialization sequence by setting OWIRE_CONTROL[RPP]. The automated initialization sequence is as follows:

1. Generate a reset pulse.
2. Listen for a response from an external device by sampling for the OWIRE device presence bit.
3. After an amount of time determined by the OWIRE standard, latch the presence bit (true or false) in OWIRE_CONTROL[PST].

If an external device is detected (PST = 1), software can begin communications on the OWIRE bus.

The presence pulse is used by the OWIRE to determine if at least one generic OWIRE device is connected. Software determines if more than one generic OWIRE device exists.

40.3.1.2 Bit Transfers

After the initialization sequence (see Reset/Presence-detect Pulse), software can write and read one bit at a time using the Control register.

40.3.1.2.1 Write-0 Sequence

The Write-0 sequence writes a zero bit to the generic OWIRE device. Setting the OWIRE_CONTROL[WR0] bit initiates the Write-0 pulse sequence. Once the write is complete, the WR0 bit is automatically cleared.

40.3.1.2.2 Write-1 / Read Sequence

The Write-1 sequence writes a one bit to the generic OWIRE device. Setting the OWIRE_CONTROL[WR1] bit initiates the Write-1 pulse sequence. Once the write is complete, the WR1 bit is automatically cleared.

Because the Write-1 and Read timings are identical, this sequence also reads a bit from the bus. The sampled value is stored in the read status bit OWIRE_CONTROL[RDST] and is valid after the WR1 bit is self-cleared.

40.3.1.3 Byte Transfers

After the initialization sequence (see Reset/Presence-detect Pulse), software can transfer a byte at a time using the TX/RX register. Writing to the register connects to the Transmit buffer; reading to the register connects to the Receive buffer. See Figure 40-2.

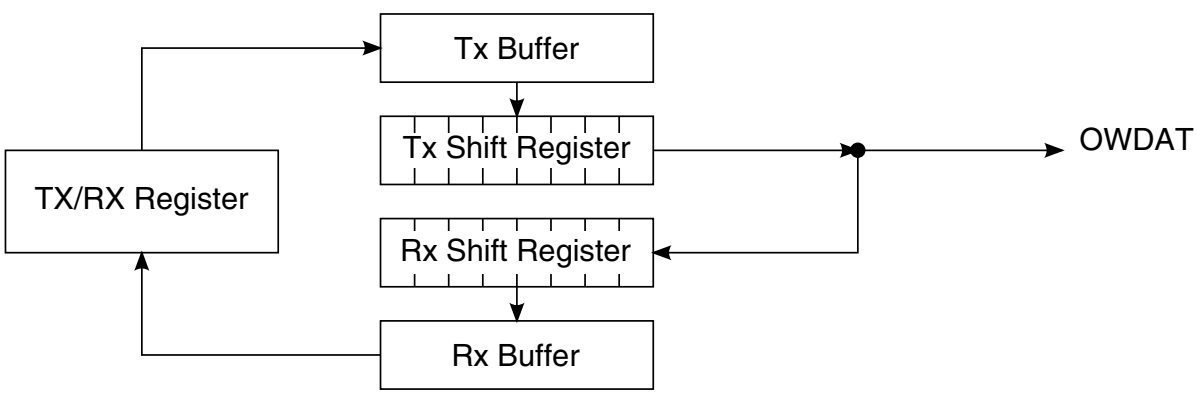


Figure 40-2. Byte Transfers

The Transmit buffer connects to an internal Transmit Shift Register where data is shifted serially onto the bus LSB first. Similarly, the Receive buffer connects to an internal Receive Shift Register where data is sampled serially from the bus.

Software can read a byte from the generic OWIRE device as follows:

1. Write 0xFF to the Transmit/Receive register location (connected to the Transmit buffer).

2. Wait for the receive-buffer-full (OWIRE_INTERRUPT[RBF]) interrupt (or poll the flag bit directly if the interrupt is disabled). During this time, the hardware is writing ones on the bus while sampling the wired-AND of the data from the device. The read data is shifted into the Receive Shift Register. When a byte is collected in the Receive Shift Register, the data is transferred to the Receive buffer, and the RBF flag is set.
3. Read from the Transmit/Receive register location (connected to the Receive buffer) upon receiving the RBF interrupt.

If the Receive buffer is full, new data is not shifted from the Receive Shift Register until the current data is read. To prevent the loss of data, software must read the TX/RX register to clear the receive-buffer-full flag (OWIRE_INTERRUPT[RBF]). This allows the Receive Shift Register to shift new data into the Receive buffer.

40.3.1.4 Search ROM Accelerator Mode

In *search ROM accelerator* mode the OWIRE relieves software from having to perform single-bit operations on the bus and helps determine if more than one generic OWIRE device exists.

The OWIRE enters search ROM accelerator mode when OWIRE_COMMAND[SRA] is set. This protocol specifies that the bus master read two bits (a bit and its complement), then writes a bit to specify which devices should remain on the bus for further processing.

Note that this mode requires that a reset followed by the search ROM command (0xF0) has already been issued on the OWIRE bus.

The OWIRE automatically exits search ROM accelerator mode if the OWIRE bus is re-initialized; see Reset/Presence-detect Pulse.

40.3.2 Low Power Mode

The OWIRE. automatically goes into low power mode whenever it is not communicating with a generic OWIRE device. The main clock is gated off in low power mode.

As soon as software writes to any register, the OWIRE exits low power mode.

40.3.3 Clocks

The OWIRE takes a main clock as a block input and passes it through a clock divider. (See the block diagram in Figure 40-1.) Software must program the divider factor to generate a 1-MHz clock that is used as an internal time base for the block, as given by the following equation:

time_base = main_clock ÷ (TIME_DIVIDER[DVDR] + 1)

For example, if the main clock frequency is 30 MHz, the value to write to the divider register is 29. If the main clock input frequency is not an integer, the programmer must ensure the time base frequency is within the range given by the following equation:

0.98 MHz ≤ time_base ≤ 1.02 MHz

NOTE

A main clock frequency below 10 MHz causes improper function of the block.

40.3.4 Reset

The OWIRE supports two levels of reset: hardware and software.

40.3.4.1 Hardware Reset

Whenever a device reset occurs, a hard reset is performed on the OWIRE, clearing all values written to all registers.

40.3.4.2 Software Reset

Software initiates a software reset by setting the reset bit OWIRE_RESET[RST]. A software reset clears all data written to the registers except for the Command and Interrupt registers (OWIRE_COMMAND, OWIRE_INTERRUPT).

NOTE

The reset register (OWIRE_RESET) itself is not cleared during a software reset. Software must clear the RST bit to release the software reset.

40.3.5 Interrupts

The OWIRE generates interrupts through the programming of the Interrupt Enable register; see Interrupt Enable Register (OWIRE_INTERRUPT_EN). The OWIRE can generate interrupts under the following conditions:

- Receive shift register or buffer full
- Transmit shift register or buffer empty
- Presence detect

Once any of these conditions are met, the Interrupt register (see Interrupt Register (OWIRE_INTERRUPT)) sets the corresponding bit and generates an interrupt if enabled in the Interrupt Enable register. The IAS bit within the Interrupt Enable register determines if the interrupt generated is active low, or active high. By default all interrupts are active high, and software should not modify IAS.

40.4 Programmable Registers

This section provides the block memory map and detailed descriptions of all registers.

OWIRE memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FA_4000	Control register (OWIRE_CONTROL)	16	R/W	0000h	40.4.1/ 2364
63FA_4002	Time Divider register (OWIRE_TIME_DIVIDER)	16	R/W	0000h	40.4.2/ 2365
63FA_4004	Reset register (OWIRE_RESET)	16	R/W	0000h	40.4.3/ 2365
63FA_4006	Command Register (OWIRE_COMMAND)	16	R/W	0000h	40.4.4/ 2366
63FA_4008	Transmit/Receive Register (OWIRE_TX/RX)	16	R/W	0000h	40.4.5/ 2367
63FA_400A	Interrupt Register (OWIRE_INTERRUPT)	16	R	000Eh	40.4.6/ 2367
63FA_400C	Interrupt Enable Register (OWIRE_INTERRUPT_EN)	16	R/W	0000h	40.4.7/ 2369

40.4.1 Control register (OWIRE_CONTROL)

The control register is used to initiate the reset/presence-detect sequence and bit transfers. The register also provides the presence-detect status and bit-read status.

Address: OWIRE_CONTROL is 63FA_4000h base + 0h offset = 63FA_4000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								RPP	PST	WR0	WR1	RDST	0		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OWIRE_CONTROL field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7 RPP	Reset/Presence-detect Pulse. This bit is self-clearing and is cleared after the presence is determined. See Reset/Presence-detect Pulse. When writing: 0 Do nothing. 1 Generate Reset Pulse and sample the bus for the presence pulse from the external device. When reading: 0 Reset pulse complete. 1 Sequence not complete.
6 PST	Presence Status. This bit is valid after the RPP bit is self-cleared. 0 Device is not present. 1 Device is present.
5 WR0	Write 0. This bit is self-clearing and is cleared when the write of the bit is complete. See Write-0 Sequence. When writing: 0 Do nothing. 1 Write a 0 bit to the interface. When reading: 0 Write sequence complete. 1 Sequence not complete.
4 WR1	Write-1 / Read. This bit is self-clearing and is cleared when the write sequence is complete. See Write-1 / Read Sequence. When writing: 0 Do nothing 1 Write a 1 bit to the interface and sample the bus. When reading:

Table continues on the next page...

OWIRE_CONTROL field descriptions (continued)

Field	Description
	0 Sequence complete. 1 Sequence not complete.
3 RDST	Read Status. This bit is valid after the WR1 bit is self cleared. 0 A 0 has been sampled. 1 A 1 has been sampled.
2–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

40.4.2 Time Divider register (OWIRE_TIME_DIVIDER)

The time divider register is used for dividing the main clock input down to 1 MHz.

Address: OWIRE_TIME_DIVIDER is 63FA_4000h base + 2h offset = 63FA_4002h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								DVDR							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OWIRE_TIME_DIVIDER field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–0 DVDR	**Divider Factor.** The internal clock divider uses this field to generate the required time base for the block. See Clocks. 0x00 1 (default) 0x01 2 0xFF 256

40.4.3 Reset register (OWIRE_RESET)

The reset register is used to perform a software reset of the OWIRE.

Address: OWIRE_RESET is 63FA_4000h base + 4h offset = 63FA_4004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0															RST
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OWIRE_RESET field descriptions

Field	Description
15–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 RST	Software Reset. See Software Reset. 0 Do not perform a software reset. 1 Initiate a software reset and hold the block in the software-reset state.

40.4.4 Command Register (OWIRE_COMMAND)

The OWIRE can be configured to run in Search ROM Accelerator mode using the command register. See Search ROM Accelerator Mode.

Address: OWIRE_COMMAND is 63FA_4000h base + 6h offset = 63FA_4006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0														SRA	0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OWIRE_COMMAND field descriptions

Field	Description
15–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 SRA	**Search ROM Accelerator. This bit is cleared when the reset-presence-pulse bit CONTROL[RPP] is set.** 0 Deactivate the search ROM accelerator. 1 Switch to search ROM accelerator mode.
0 Reserved	This read-only field is reserved and always has the value zero. Reserved

40.4.5 Transmit/Receive Register (OWIRE_TX/RX)

Data sent and received from the OWIRE passes through the Transmit/Receive (TX/RX) register location. The OWIRE is double-buffered with separate transmit and receive buffers connected to the TX/RX register. See Byte Transfers.

Address: OWIRE_TX/RX is 63FA_4000h base + 8h offset = 63FA_4008h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								DATA							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OWIRE_TX/RX field descriptions

Field	Description
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–0 DATA	Data byte. When writing: The data byte is written to the Transmit buffer. When reading: A data byte is read from the Receive buffer. The data is valid only when INTERRUPT[RBF] is set.

40.4.6 Interrupt Register (OWIRE_INTERRUPT)

Flags for the reset/presence-detect sequence and byte transfer operations are located in the Interrupt Register. These flags can generate an interrupt if the corresponding enable bit is set in the Interrupt Enable Register.

If interrupts are enabled, reading the Interrupt Register deactivates the interrupt even if all current flags are not cleared; therefore, the interrupt service routine should clear all pending flags during each routine call.

NOTE

When a byte is written to the Transmit/Receive Register, software then waits for a Transmit Shift Register Empty (TSRE) interrupt to occur. When the TSRE flag is set, the Receive Buffer Full (RBF) flag is also set. The RBF flag does not trigger an interrupt, assuming it is disabled. However, software should read the Transmit/Receive Register to clear the

RBF flag in order to give a proper status of the pending interrupts.

Address: OWIRE_INTERRUPT is 63FA_4000h base + Ah offset = 63FA_400Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0										RSRF	RBF	TSRE	TBE	PDR	PD
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

OWIRE_INTERRUPT field descriptions

Field	Description
15–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5 RSRF	Receive Shift Register Full. Hardware automatically clears this flag when data in the Receive Shift Register is transferred to the Receive buffer. 0 The Receive Shift Register is empty or currently receiving data. 1 A byte is waiting in the Receive Shift Register to be transferred to the Receive buffer.
4 RBF	Receive Buffer Full. This flag is cleared when software reads the byte from the TX/RX register. This flag prevents new data from being shifted into the Receive buffer from the Receive Shift Register. 0 No new data 1 A byte is waiting to be read from the TX/RX register.
3 TSRE	Transmit Shift Register Empty. Hardware automatically clears this flag when data in the Transmit buffer is transferred to the Transmit Shift Register. 0 Sending data 1 The Transmit Shift Register is empty and is ready to receive the next byte from the Transmit buffer.
2 TBE	Transmit Buffer Empty. This flag is cleared when software writes a byte to the TX/RX register. 0 The Transmit buffer is currently sending data to the Transmit Shift Register. 1 Nothing to transmit
1 PDR	Presence Detect Result. When a presence-detect (PD) interrupt occurs, this bit reflects the result of the presence-detect sequence. Note that this bit does not generate an interrupt. 0 Device found 1 Device not found
0 PD	Presence Detect. After an OWIRE reset has been issued, this flag is set after the appropriate amount of time for a presence detect pulse to have occurred. This flag is cleared when software reads the Interrupt Register. 0 A reset/presence-detect sequence has not been issued. 1 Reset/presence-detect sequence has completed. The result is provided in the PDR bit.

40.4.7 Interrupt Enable Register (OWIRE_INTERRUPT_EN)

The Interrupt Enable Register allows the system programmer to specify the source of interrupts. During a reset (hardware or software), all bits in this register are cleared, disabling all interrupt sources.

Address: OWIRE_INTERRUPT_EN is 63FA_4000h base + Ch offset = 63FA_400Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0										ERSF	ERBF	ETSE	ETBE	IAS	EPD
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OWIRE_INTERRUPT_EN field descriptions

Field	Description
15–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5 ERSF	Enable Receive Shift Register Full Interrupt. 0 Disable interrupt. 1 Enable interrupt.
4 ERBF	Enable Receive Buffer Full Interrupt. 0 Disable interrupt. 1 Enable interrupt.
3 ETSE	Enable Transmit Shift Register Empty Interrupt. 0 Disable interrupt. 1 Enable interrupt.
2 ETBE	Enable Transmit Buffer Empty Interrupt. 0 Disable interrupt. 1 Enable interrupt.
1 IAS	Interrupt Trigger Active State. This bit determines the polarity for all interrupts. Note that this bit is not an interrupt-enable bit. 0 Active high 1 Active low
0 EPD	Enable Presence Detect. 0 Disable interrupt. 1 Enable interrupt.

Chapter 41
Performance Monitor (PerfMon)

41.1 Introduction

This chapter describes the Performance Monitor (PerfMon) function included on the i.MX50.

41.2 Overview

The i.MX50 implements a real time systerm performance monitor, it provides the following functions:

- Real-time performance statistics collection on AXI buses
- Single or Multiple master transactions monitoring
- Various interrupts support: address trap, latency threshold and so on.
- AXI protocol and out-of-order sequence support

The performance monitor is illustrated in Figure 41-1.

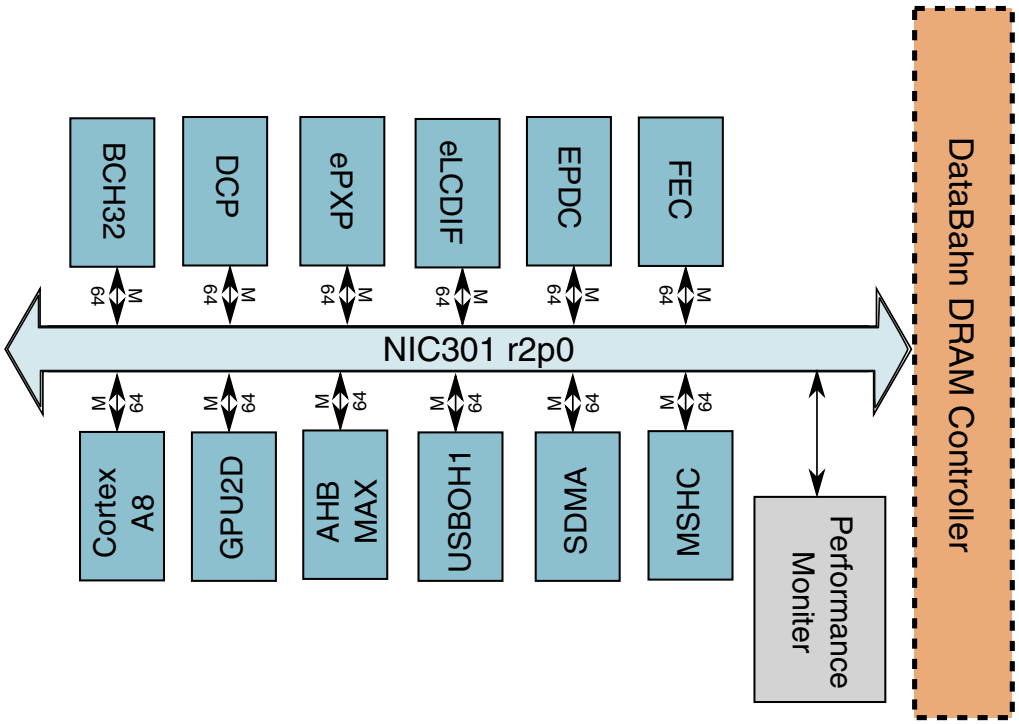


Figure 41-1. Block diagram of PerfMon

41.3 Operation

The role of the performance monitor in an AXI system is to collect real time bus transaction statistics, including transaction average and maximum latency, total data bandwidth, total active cycles. The statistics collection can be configured for single master or multiple masters on the same AXI bus, and can be enable for either read or write transactions.

Table 41-1. Masters and IDs Supported in i.MX50 Perfmon

Master	ARM	DCP	ePXP	USB	GPU2D	BCH32	AHBMAX	EPDC	eLCDIF	SDMA	FEC	MSHC
MID	0	1	2	3	4	5	6	7	8	10	10	11

AXI protocol supports out-of-order transaction completion. In i.MX50, out-of-order transaction is programmable in Databahn DRAM controller. Tag ID is used to trace each transaction and match the data phase with the correct address phase. Performance Monitor supports out-of-order transactions. AXI protocol separates the address/control phase and data phase, and has the ability to issue multiple outstanding addresses. It gives

an ID tag to every transaction across the interface , which consists 4bit master ID and 4 bit subID. Every master has its own unique master ID, and each master could have up to 16 sub IDs, which means up to 16 outstanding address phases.

PerfMon collects the accumulated latency on cycle counts and number of transfers, the average latency can be calculated by dividing the total latency over number of transfers. Since the AXI protocol separates the address and data phases, the address and data phases could be overlapped, and individual data phase could overlap with multiple address phase. Statistics on total cycles, total data transfer in bytes will be collected as well. All statistics are collected for either read or write transactions, which could be switched by the enable bit in the control register.

To calculate the maximum latency and support out-of-order transaction, PerfMon needs to cache the tag ID for all the outstanding requests. The potential number of outstanding requests depends on how deep the command queue in the memory controller (Databahn) which includes two level of command queues. A scoreboard will be implemented to match the tag ID of the ongoing data phase with the ID from the outstanding requests. PerfMon snapshots the master ID (8 bits), burst length (4 bits), burst type (2 bits), burst size(3 bits) for the content register of maximum latency, but no transfer address (32 bits).

Registers in PerfMon are accessible through APBH bridge, and PerfMon registers sits on APBH clock domain. Shade registers are added for the coherence among statistics/status registers, and the snapshot bit in PerfMon control register will trigger the shade register being filled at the same cycle for these registers: total transfers, total latency, maximum latency, content register for max_latency, total data bytes, total active cycles, and total cycles.

Various interrupts supported in this module are as follows:

- Address trap IRQ-IRQ will be asserted if any transaction address hit within the pre-configured address range.
- Address range inversion IRQ-IRQ will be asserted if any transaction address hit outside the pre-configured address range.
- Max latency threshold IRQ-IRQ will be asserted if maximum latency is above the value defined in the threshold register.
- Bus error IRQ-IRQ will be asserted if any AXI transaction triggers an error response.

41.4 Programmable Registers

PERFMON Hardware Register Format Summary

PERFMON memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_4000	PerfMon Control Register (PERFMON_CTRL)	32	R/W	C000_0000h	41.4.1/ 2374
4101_4004	PerfMon Control Register (PERFMON_CTRL_SET)	32	R/W	C000_0000h	41.4.1/ 2374
4101_4008	PerfMon Control Register (PERFMON_CTRL_CLR)	32	R/W	C000_0000h	41.4.1/ 2374
4101_400C	PerfMon Control Register (PERFMON_CTRL_TOG)	32	R/W	C000_0000h	41.4.1/ 2374
4101_4010	PerfMon Master Enable Register (PERFMON_MASTER_EN)	32	R/W	0000_0000h	41.4.2/ 2376
4101_4020	PerfMon Trap Range Low Address Register (PERFMON_TRAP_ADDR_LOW)	32	R/W	0000_0000h	41.4.3/ 2377
4101_4030	PerfMon Trap Range High Address Register (PERFMON_TRAP_ADDR_HIGH)	32	R/W	0000_0000h	41.4.4/ 2378
4101_4040	PerfMon Latency Threshold Register (PERFMON_LAT_THRESHOLD)	32	R/W	0000_0000h	41.4.5/ 2379
4101_4050	PerfMon AXI Active Cycle Count Register (PERFMON_ACTIVE_CYCLE)	32	R	0000_0000h	41.4.6/ 2379
4101_4060	PerfMon Transfer Count Register (PERFMON_TRANSFER_COUNT)	32	R	0000_0000h	41.4.7/ 2380
4101_4070	PerfMon Total Latency Count Register (PERFMON_TOTAL_LATENCY)	32	R	0000_0000h	41.4.8/ 2380
4101_4080	PerfMon Total Data Count Register (PERFMON_DATA_COUNT)	32	R	0000_0000h	41.4.9/ 2381
4101_4090	PerfMon Maximum Latency Register (PERFMON_MAX_LATENCY)	32	R	0000_0000h	41.4.10/ 2381
4101_40A0	PerfMon Debug Register (PERFMON_DEBUG)	32	R/W	0000_0001h	41.4.11/ 2382
4101_40B0	PerfMon Version Register (PERFMON_VERSION)	32	R	0101_0000h	41.4.12/ 2383

41.4.1 PerfMon Control Register (PERFMON_CTRL*n*)

The PerfMon Control Register specifies the reset state and the interrupt controls for the Performance Monitor.

This register controls various functions of the system performance monitor.

EXAMPLE

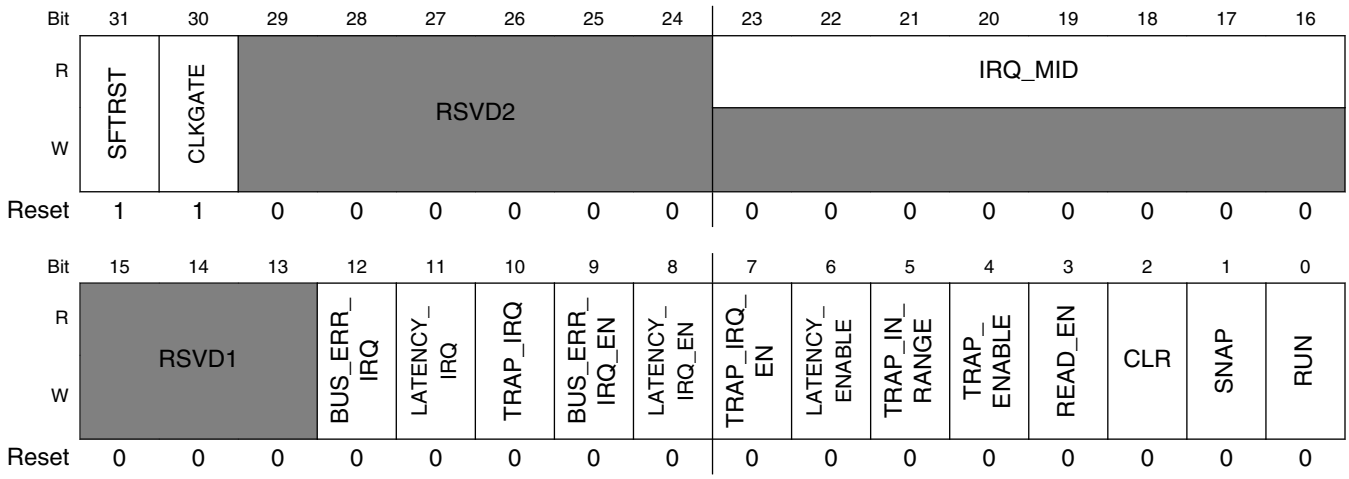
```
// turn off soft reset and clock gating
PERFMON_CTRL_CLR(BM_PERFMON_CTRL_SFTRST| BM_PERFMON_CTRL_CLKGATE);
```

Addresses: PERFMON_CTRL is 4101_4000h base + 0h offset = 4101_4000h

PERFMON_CTRL_SET is 4101_4000h base + 4h offset = 4101_4004h

PERFMON_CTRL_CLR is 4101_4000h base + 8h offset = 4101_4008h

PERFMON_CTRL_TOG is 4101_4000h base + Ch offset = 4101_400Ch



PERFMON_CTRL*n* field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. 0x0 **RUN** — Allow PerfMon to operate normally. 0x1 **RESET** — Hold PerfMon in reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. 0x0 **RUN** — Allow PerfMon to operate normally. 0x1 **NO_CLKS** — Do not clock PerfMon gates in order to minimize power consumption.
29–24 RSVD2	Always set this bit field to zero.
23–16 IRQ_MID	This field contains the master ID and sub ID associated with the interrupt. If multiple IRQs, it is the ID associated the first IRQ transaction, and will not update until the IRQ being cleared.
15–13 RSVD1	Always set this bit field to zero.
12 BUS_ERR_IRQ	This bit is set if there is error on any AXI transaction and the BUS_ERR_IRQ_EN bit is set.
11 LATENCY_IRQ	This status bit is set if maximum latency is above the value defined in the latency threshold register and the LATENCY_ENABLE bit is set. Writing a one to its SCT clear address to clear it.
10 TRAP_IRQ	This status bit is set to indicate that an address trap occurs. This bit is cleared by software by writing a one to its SCT clear address. This bit is set if axi address is within the pre-configured address range and the trap function is enabled with TRAP_ENABLE bit.
9 BUS_ERR_IRQ_EN	Enables the PerfMon AXI BUS ERROR IRQ. When an error occurs and this bit is set, an interrupt is sent to the ARM core.

Table continues on the next page...

PERFMON_CTRL*n* field descriptions (continued)

Field	Description
8 LATENCY_IRQ_EN	Enables the PerfMon Latency threshold IRQ. When an error occurs and this bit is set, an interrupt is sent to the ARM core.
7 TRAP_IRQ_EN	Enables the PerfMon Address Trap IRQ. When an trap occurs and this bit is set, an interrupt is sent to the ARM core.
6 LATENCY_ENABLE	Enables the PerfMon AXI latency threshold functions. When a transfer latency larger than the threshold and this bit is set, the LATENCY_IRQ status bit will be set.
5 TRAP_IN_RANGE	Determines whether the debug trap function causes a match when the master address is inside (low-address <= current-address <= high-address) the specified range. 0 = The trap occurs when the master address falls outside of the range. 1 = The check is inside the range.
4 TRAP_ENABLE	Enables the PerfMon AXI address trap functions. When a trap occurs and this bit is set, the TRAP_IRQ status bit will be set.
3 READ_EN	Set this bit to One to monitor all Read transactions, set to zero to monitor all Write transactions. 0 = performance monitoring on all WRITE activities. 1 = performance monitoring on all READ activities.
2 CLR	Set this bit to clear all the PerfMon's statistics registers. This bit will be reset to 0 once the clear process is done.
1 SNAP	Set this bit to snap shot PerfMon's statistics registers into the shadow registers for reads. PerfMon will continue collecting and updating statistics registers. This bit will be reset to 0 once the snapshot processing is done.
0 RUN	Set this bit to one to enable the PerfMon operation. 0x0 **HALT** — No PerfMon command in progress. 0x1 **RUN** — Process Performance monitoring.

41.4.2 PerfMon Master Enable Register (PERFMON_MASTER_EN)

The Master Enable Register defines single or multiple MasterIDs to be monitored by Perfmon.

This register configs the MasterID mask to be monitored. The master ID of Core is 0, DCP is 1, ePXP is 2, USBOH1 is 3, GPU2D is 4, BCH32 is 5 , AHB MAX is 6, EPDC is 7, eLCDIF is 8, SDMA is 9, FEC is 10, and MSHC is 11.

Address: PERFMON_MASTER_EN is 4101_4000h base + 10h offset = 4101_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD0																MID15	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PERFMON_MASTER_EN field descriptions

Field	Description
31–16 RSVD0	Always set this bit field to zero.
15 MID15	Set to 1 to enable performance monitoring and statistics collection on MasterID 15.
14 MID14	Set to 1 to enable performance monitoring and statistics collection on MasterID 14.
13 MID13	Set to 1 to enable performance monitoring and statistics collection on MasterID 13.
12 MID12	Set to 1 to enable performance monitoring and statistics collection on MasterID 12.
11 MID11	Set to 1 to enable performance monitoring and statistics collection on MasterID 11.
10 MID10	Set to 1 to enable performance monitoring and statistics collection on MasterID 10.
9 MID9	Set to 1 to enable performance monitoring and statistics collection on MasterID 9.
8 MID8	Set to 1 to enable performance monitoring and statistics collection on MasterID 8.
7 MID7	Set to 1 to enable performance monitoring and statistics collection on MasterID 7.
6 MID6	Set to 1 to enable performance monitoring and statistics collection on MasterID 6.
5 MID5	Set to 1 to enable performance monitoring and statistics collection on MasterID 5.
4 MID4	Set to 1 to enable performance monitoring and statistics collection on MasterID 4.
3 MID3	Set to 1 to enable performance monitoring and statistics collection on MasterID 3.
2 MID2	Set to 1 to enable performance monitoring and statistics collection on MasterID 2.
1 MID1	Set to 1 to enable performance monitoring and statistics collection on MasterID 1.
0 MID0	Set to 1 to enable performance monitoring and statistics collection on MasterID 0.

41.4.3 PerfMon Trap Range Low Address Register (PERFMON_TRAP_ADDR_LOW)

The Debug Trap Range Low Address Register defines the lower bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AXI cycle occurs within this range.

This register sets the lower address that defines the debug trap function. When this function is enabled, any active AXI cycle which accesses this range will trigger an interrupt to the ARM core.

Address: PERFMON_TRAP_ADDR_LOW is 4101_4000h base + 20h offset = 4101_4020h

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

PERFMON_TRAP_ADDR_LOW field descriptions

Field	Description
31–0 ADDR	This field contains the 32-bit lower address for the debug trap range.

41.4.4 PerfMon Trap Range High Address Register (PERFMON_TRAP_ADDR_HIGH)

The Trap Range High Address Register defines the upper bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AXI cycle occurs within this range.

This register sets the upper address that defines the debug trap function. When this function is enabled, any active AXI cycle which accesses this range will trigger an interrupt to the ARM core.

Address: PERFMON_TRAP_ADDR_HIGH is 4101_4000h base + 30h offset = 4101_4030h

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

PERFMON_TRAP_ADDR_HIGH field descriptions

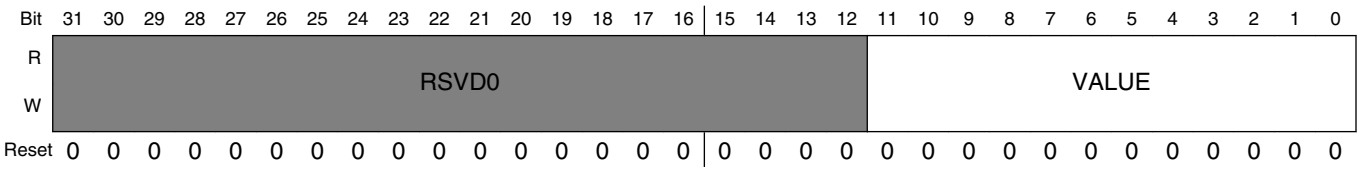
Field	Description
31–0 ADDR	This field contains the 32-bit upper address for the debug trap range.

41.4.5 PerfMon Latency Threshold Register (PERFMON_LAT_THRESHOLD)

The Latency Threshold Register defines the threshold for AXI transaction latency that can be enabled to trigger an interrupt to the ARM core when an AXI transaction latency is above this value.

This register sets the threshold value of AXI transaction. When this function is enabled, any active AXI cycle which has a latency above this value will trigger an interrupt to the ARM core.

Address: PERFMON_LAT_THRESHOLD is 4101_4000h base + 40h offset = 4101_4040h



PERFMON_LAT_THRESHOLD field descriptions

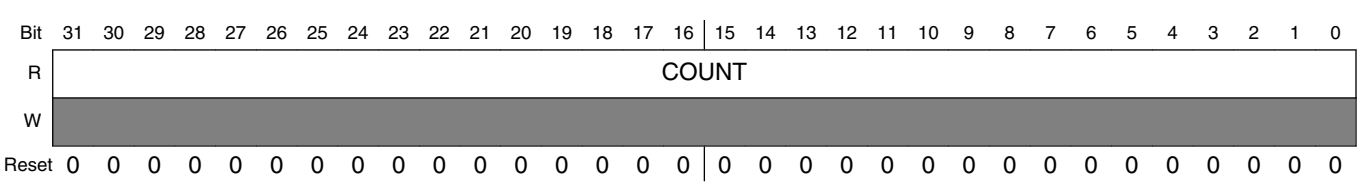
Field	Description
31–12 RSVD0	Always set this bit field to zero.
11–0 VALUE	This field contains the 12-bit transaction latency threshold.

41.4.6 PerfMon AXI Active Cycle Count Register (PERFMON_ACTIVE_CYCLE)

The Active Cycle Count Register counts the number of AXI cycles during which a transaction is active on AXI bus.

This register counts the number of AXI cycles in which a master was requesting a transfer, and the slave had not responded. This is including cycles in which it was requesting transfers but was not accepted by slave yet. The Master Enable Register PERFMON_MASTER_EN are used to mask which master(s) cycles are actually recorded here.

Address: PERFMON_ACTIVE_CYCLE is 4101_4000h base + 50h offset = 4101_4050h



PERFMON_ACTIVE_CYCLE field descriptions

Field	Description
31–0 COUNT	This field contains the total active AXI cycle counts. If overflow, the value will be 0xFFFFFFFF.

41.4.7 PerfMon Transfer Count Register (PERFMON_TRANSFER_COUNT)

The Transfer Count Register defines the total number of transfers has been completed during the whole monitoring period.

Divide the register PERFMON_LATENCY_CYCLE value over this value of this register will give the average latency for each transfer.

Address: PERFMON_TRANSFER_COUNT is 4101_4000h base + 60h offset = 4101_4060h

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

PERFMON_TRANSFER_COUNT field descriptions

Field	Description
31–0 VALUE	This field contains the total amount transfers. If overflow, the value will be 0xFFFFFFFF.

41.4.8 PerfMon Total Latency Count Register (PERFMON_TOTAL_LATENCY)

The Total Latency Count Register defines the total number of latency in cycles during the monitoring period.

Divide the register value over value of register PERFMON_TRANSFER_COUNT will give the average latency for each transfer.

Address: PERFMON_TOTAL_LATENCY is 4101_4000h base + 70h offset = 4101_4070h

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

PERFMON_TOTAL_LATENCY field descriptions

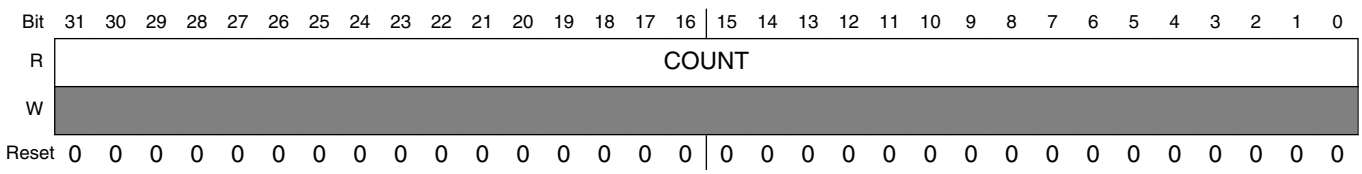
Field	Description
31–0 COUNT	This field contains the total latency in cycle counts. If overflow, the value will be 0xFFFFFFFF.

41.4.9 PerfMon Total Data Count Register (PERFMON_DATA_COUNT)

The Data Count Register defines the total number of data transferred in bytes during the monitoring period.

Total transfer data count in bytes.

Address: PERFMON_DATA_COUNT is 4101_4000h base + 80h offset = 4101_4080h



PERFMON_DATA_COUNT field descriptions

Field	Description
31–0 COUNT	This field contains the total bytes of data transferred. If overflow, the value will be 0xFFFFFFFF.

41.4.10 PerfMon Maximum Latency Register (PERFMON_MAX_LATENCY)

The Maximum Latency Register contains the maximum latency in cycles during the monitoring period.

This register contains the maximum latency counts in cycle and control signals for this transaction.

Address: PERFMON_MAX_LATENCY is 4101_4000h base + 90h offset = 4101_4090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ABURST		ALEN				ASIZE			TAGID[-8:1]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TAGID[bit 0]	RSVD0			COUNT											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PERFMON_MAX_LATENCY field descriptions

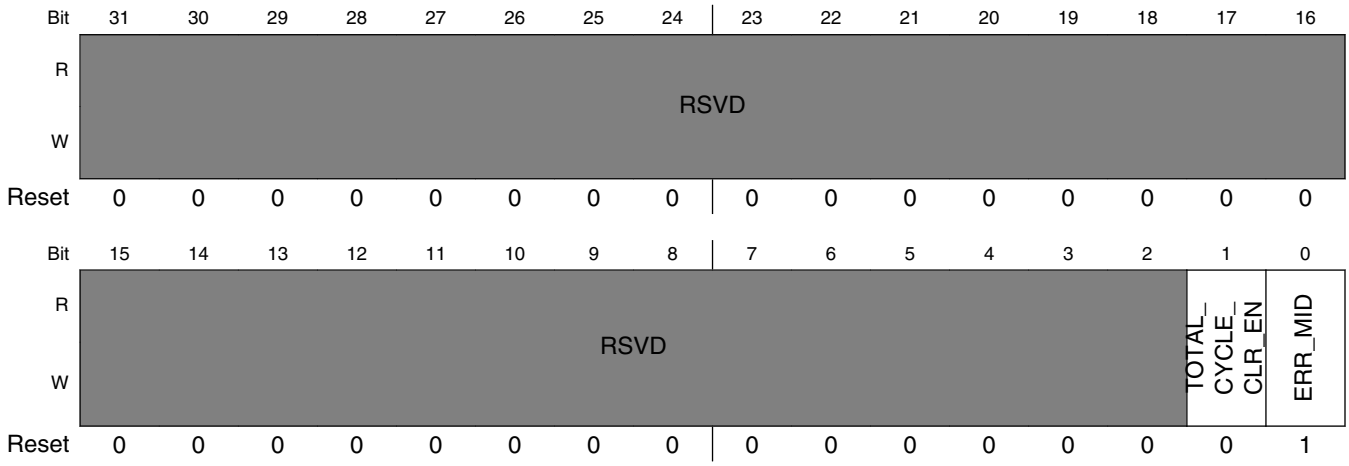
Field	Description
31–30 ABURST	This field contains axi_aburst signal associated with the worst latency transaction.
29–26 ALEN	This field contains axi_alen signal associated with the worst latency transaction.
25–23 ASIZE	This field contains axi_asize signal associated with the worst latency transaction.
22–15 TAGID	This field contains the master id and sub id associated with the worst latency transaction.
14–12 RSVD0	Always set this bit field to zero.
11–0 COUNT	This field contains the worst transfer latency. If overflow, the value will be 0xFFF.

41.4.11 PerfMon Debug Register (PERFMON_DEBUG)

The Debug Register is for internal use only.

Divide the register PERFMON_LATENCY_CYCLE value over this value of this register will give the average latency for each transfer.

Address: PERFMON_DEBUG is 4101_4000h base + A0h offset = 4101_40A0h



PERFMON_DEBUG field descriptions

Field	Description
31–2 RSVD	Always set this bit field to zero.
1 TOTAL_CYCLE_CLR_EN	This field is for debug purpose. Set to 1 to clear the internal total cycle register no matter there is pending request or not when the clear bit in control register is set.
0 ERR_MID	This field is for debug purpose. Set to 0 will not record the MID for bus error irq.

41.4.12 PerfMon Version Register (PERFMON_VERSION)

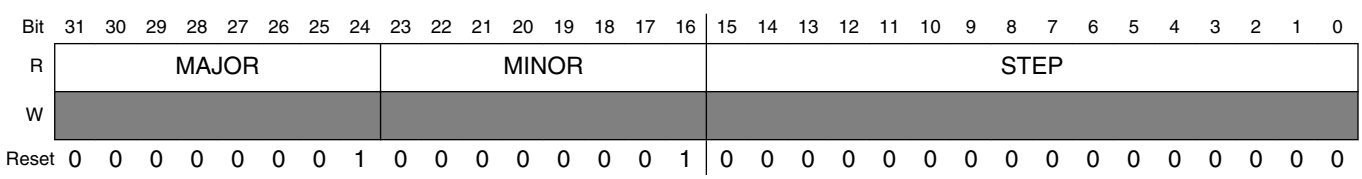
The Version Register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
if (PERFMON_VERSION.B.MAJOR != 1) Error();
```

Address: PERFMON_VERSION is 4101_4000h base + B0h offset = 4101_40B0h



PERFMON_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 42
Pulse Width Modulation (PWM)

42.1 Overview

The Pulse Width Modulation (PWM) has a 16-bit counter, and is optimized to generate sound from stored sample audio images and it can also generate tones. It uses 16-bit resolution and a 4 x 16 data FIFO.

This section presents an overview of the PWM. The figure below illustrates the PWM diagram.

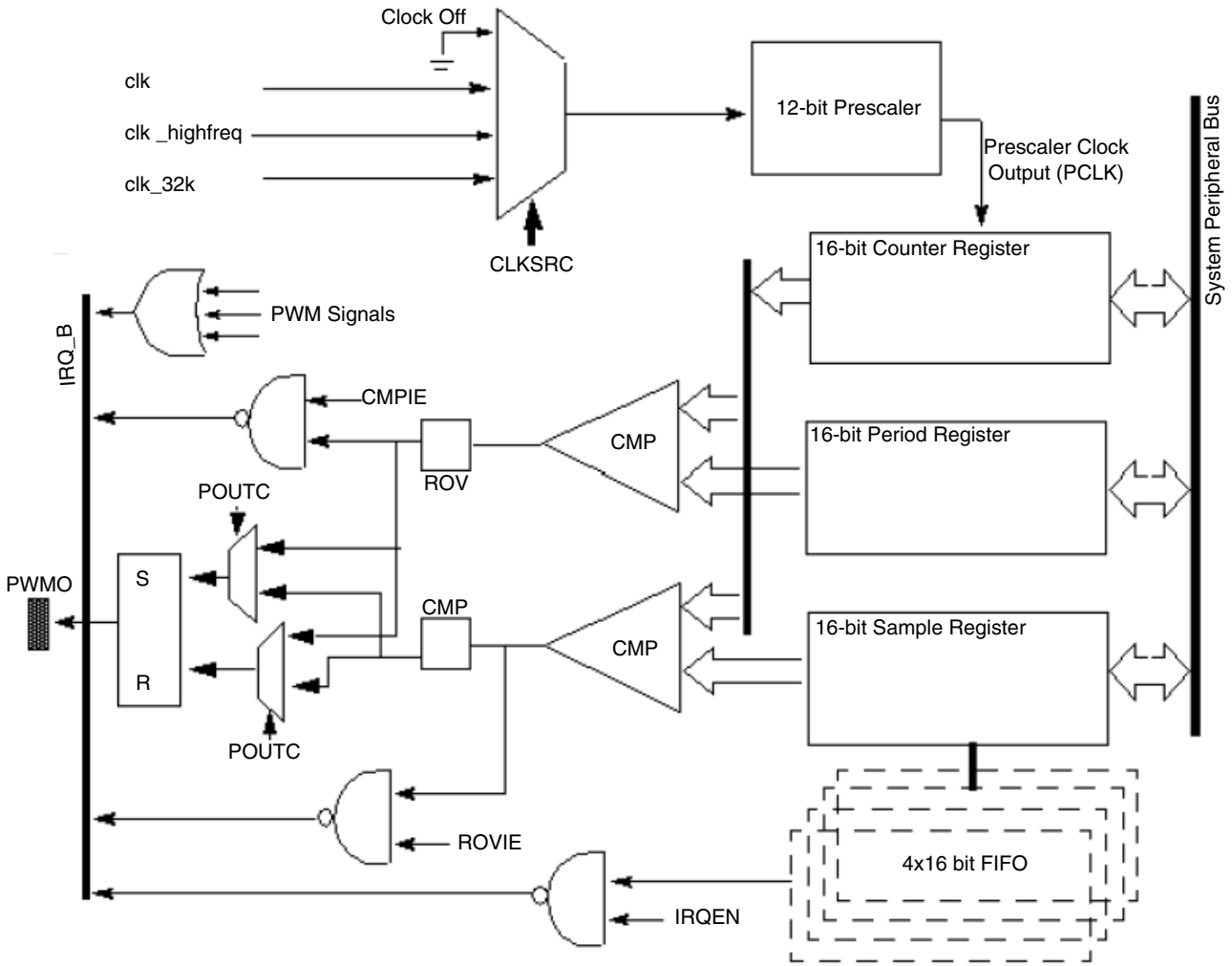


Figure 42-1. Pulse-Width Modulator Block Diagram

The following features characterize the PWM:

- 16-bit up-counter with clock source selection
- 4 x 16 FIFO to minimize interrupt overhead
- 12-bit prescaler for division of clock
- Sound and melody generation
- Active high or active low configured output
- Can be programmed to be active in low-power mode
- Can be programmed to be active in debug mode
- Interrupts at compare and rollover

42.2 Signal Description

The PWM follows IP Bus protocol when interfacing with the processor core. It does not have any interface signals with any other block inside the chip except for clock and reset inputs from the Clock Control Module (CCM), System Reset Controller (SRC), and interrupt signals to the processor interrupt handler. There is a single output signal going outside the chip boundary.

42.2.1 External Signals

PWM has a single output signal to the chip boundary named ipp_do_pwmo.

The following table outlines the external signals.

Table 42-1. External Signals

Name	Direction	Function	Reset State	Pull up
PWMxO	Output	This is the functional output of the PWM. The modulated signal of the block is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the PWM. The smallest period can be two ipg_clk clock periods with duty cycle of 50%.	0	-

42.3 Functional Description

The following sections detail the PWM operation and function.

42.3.1 Operation

The output of the PWM is a toggling signal whose frequency and duty cycle can be modulated by programming the appropriate registers. It has a 16-bit up counter which counts from 0x0000 until the counter value equals the [Value in Period register] + 1. After this match occurs the Counter is reset to 0x0000.

At the beginning of a count period cycle, the PWMO pin is set to one (default) and the counter begins counting up from 0x0000. The sample value in the sample FIFO is compared on each count of prescaler clock. When the sample and count values match, the PWMO signal is cleared to zero (default). The counter continues counting until the period match occurs and subsequently another period cycle begins.

When the PWM is enabled the counter starts running and generates an output with the reset values in the period and sample registers. It is recommended that the programming of these registers be done before PWM is enabled.

A hardware reset results in all the PWM count and sample registers begin cleared and the FIFO being flushed. The control register shows that FIFO is empty and it can be written into, and the PWM is disabled. A software reset has the same results, however the state of the STOPEN, DOZEN, WAITEN, and DBGEN bits in the control register are not affected. Software reset can be asserted even when the PWM is in disabled state.

42.3.1.1 Clocks

The clock that feeds the prescaler can be selected from:

- High frequency Clock (ipg_clk_highfreq) pat_ref or CKIH

  This is a high frequency clock, provided by the Clock Control Module (CCM). This clock should be in the low power mode when the ipg_clk is turned off. Thus, the PWM can be run on this clock in the low power mode.

- Low Reference Clock (ipg_clk_32k) CKIL

  This is the 32 KHz low reference clock which is provided by the CCM. This clock should be on in the low power mode when ipg_clk is turned off. Thus, PWM can be run on this clock in the low power mode.

- Global Functional Clock (ipg_clk)

  This clock should be on in normal operations. In low power modes it can be switched off.

The clock input source is determined by the CLKSRC field of the PWM control register. The CLKSRC value should only be changed when the PWM is disabled.

A change in the value of the PRESCALER field of the control register is immediately reflected on its output clock frequency.

42.3.1.2 FIFO

Digital sample values can be loaded into the pulse-width modulator as 16-bit words. The endianess can be changed using the BCTR and HCTR bits of the control register. A 4-word (16-bit) FIFO minimizes interrupt overhead. A maskable interrupt is generated when the number of data words fall below the water level set by the FWM field in the control register.

A write in the sample register results in the value being stored into the FIFO if it is not full. A write when the FIFO is full sets FWE (FIFO write error) bit in the status register and the FIFO contents remain unchanged. The FIFO can be written into when the PWM is disabled. The FIFOAV field shows how many data words are currently contained in the FIFO and whether or not it can be written into.

A read on the sample register yields the current FIFO value that is being used, or will be used, by the PWM for generation on the output signal. Therefore, a write and a subsequent read on the sample register may result in different values being obtained.

42.3.1.3 Rollover and Compare Event

The counter is reset to 0x0000 after its value equals the PERIOD + 1 and resumes counting thereafter. This event is referred to as a rollover. When PERIOD = 0x0000, the counter is reset after count reaches 0x0001. Therefore PERIOD = 0xffff or 0xfffe results in the counter value being reset after count till 0xffff. During a rollover event the output is either set (default), reset or has no effect according to the programming of the POUTC field in the control register. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

When the counter value reaches the sample value, the output of the PWM is reset (default), set or has no effect according to the programming of the POUTC field of control register. This event is referred to as a compare event. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

If the rollover event sets the PWM output signal, the compare event will reset it and vice versa for a particular programming configuration of POUTC field.

42.3.1.4 Low Power Mode Behavior

In low power modes, if the clock from the selected clock source is available, the PWM counter continues to run and an output is produced, depending on whether the control bit for that mode is set or not. In the absence of the clock itself, or if the corresponding low power bit in the control register is 0, the counter is reset and resumes counting when it exits the low power mode.

42.3.1.5 Debug Mode Behavior

In debug mode, PWM has the option of continuing to run or be halted. If the DBGEN bit is not set in the PWM_PWMCR, the PWM is halted. If the DBGEN bit is set, then the PWM will continue to run in the debug mode.

42.4 Programmable Registers

The PWM includes six user-accessible 32-bit registers.

PWM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FB_4000	PWM Control Register (PWM-1_PWMCR)	32	R/W	0000_0000h	42.4.1/ 2391
53FB_4004	PWM Status Register (PWM-1_PWMSR)	32	w1c	0000_0008h	42.4.2/ 2393
53FB_4008	PWM Interrupt Register (PWM-1_PWMIR)	32	R/W	0000_0000h	42.4.3/ 2394
53FB_400C	PWM Sample Register (PWM-1_PWMSAR)	32	R/W	0000_0000h	42.4.4/ 2395
53FB_4010	PWM Period Register (PWM-1_PWMPR)	32	R/W	0000_FFFEh	42.4.5/ 2396
53FB_4014	PWM Counter Register (PWM-1_PWMCNR)	32	R	0000_0000h	42.4.6/ 2396
53FB_8000	PWM Control Register (PWM-2_PWMCR)	32	R/W	0000_0000h	42.4.1/ 2391
53FB_8004	PWM Status Register (PWM-2_PWMSR)	32	w1c	0000_0008h	42.4.2/ 2393
53FB_8008	PWM Interrupt Register (PWM-2_PWMIR)	32	R/W	0000_0000h	42.4.3/ 2394

Table continues on the next page...

PWM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FB_800C	PWM Sample Register (PWM-2_PWMSAR)	32	R/W	0000_0000h	42.4.4/ 2395
53FB_8010	PWM Period Register (PWM-2_PWMPR)	32	R/W	0000_FFFEh	42.4.5/ 2396
53FB_8014	PWM Counter Register (PWM-2_PWMCNR)	32	R	0000_0000h	42.4.6/ 2396

42.4.1 PWM Control Register (PWMx_PWMCR)

The PWM control register (PWM_PWMCR) is used to configure the operating settings of the PWM. It contains the prescaler for the clock division.

Addresses: PWM-1_PWMCR is 53FB_4000h base + 0h offset = 53FB_4000h

PWM-2_PWMCR is 53FB_8000h base + 0h offset = 53FB_8000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				FWM		STOPEN	DOZEN	WAITEN	DBGEN	BCTR	HCTR	POUTC		CLKSRC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALER												SWR	REPEAT		EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWMx_PWMCR field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value zero. Reserved. These reserved bits are always read as zero.
27–26 FWM	FIFO Water Mark. These bits are used to set the data level at which the FIFO empty flag will be set and the corresponding interrupt generated 00 FIFO empty flag is set when there are more than or equal to 1 empty slots in FIFO 01 FIFO empty flag is set when there are more than or equal to 2 empty slots in FIFO 10 FIFO empty flag is set when there are more than or equal to 3 empty slots in FIFO 11 FIFO empty flag is set when there are more than or equal to 4 empty slots in FIFO
25 STOPEN	Stop Mode Enable. This bit keeps the PWM functional while in stop mode. When this bit is cleared, the input clock is gated off in stop mode. This bit is not affected by software reset. It is cleared by hardware reset.

Table continues on the next page...

PWMx_PWMCR field descriptions (continued)

Field	Description
	0 Inactive in stop mode 1 Active in stop mode
24 DOZEN	Doze Mode Enable. This bit keeps the PWM functional in doze mode. When this bit is cleared, the input clock is gated off in doze mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in doze mode 1 Active in doze mode
23 WAITEN	Wait Mode Enable. This bit keeps the PWM functional in wait mode. When this bit is cleared, the input clock is gated off in wait mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in wait mode 1 Active in wait mode
22 DBGEN	Debug Mode Enable. This bit keeps the PWM functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in debug mode 1 Active in debug mode
21 BCTR	Byte Data Swap Control. This bit determines the byte ordering of the 16-bit data when it goes into the FIFO from the sample register. 0 byte ordering remains the same 1 byte ordering is reversed
20 HCTR	Half-word Data Swap Control. This bit determines which half word data from the 32-bit IP Bus interface is written into the lower 16 bits of the sample register. 0 Half word swapping does not take place 1 Half words from write data bus are swapped
19–18 POUTC	PWM Output Configuration. This bit field determines the mode of PWM output on the output pin. 00 Output pin is set at rollover and cleared at comparison 01 Output pin is cleared at rollover and set at comparison 10 PWM output is disconnected 11 PWM output is disconnected
17–16 CLKSRC	Select Clock Source. These bits determine which clock input will be selected for running the counter. After reset the system functional clock is selected. The input clock can also be turned off if these bits are set to 00. This field value should only be changed when the PWM is disabled 00 Clock is off 01 ipg_clk 10 ipg_clk_highfreq 11 ipg_clk_32k
15–4 PRESCALER	Counter Clock Prescaler Value. This bit field determines the value by which the clock will be divided before it goes to the counter. 0x000 Divide by 1 0x001 Divide by 2 0xfff Divide by 4096

Table continues on the next page...

PWMx_PWMCR field descriptions (continued)

Field	Description
3 SWR	Software Reset. PWM is reset when this bit is set to 1. It is a self clearing bit. A write 1 to this bit is a single wait state write cycle. When the block is in reset state this bit is set and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values except for the STOPEN, DOZEN, WAITEN, and DBGEN bits in this control register. 0 PWM is out of reset 1 PWM is undergoing reset
2–1 REPEAT	Sample Repeat. This bit field determines the number of times each sample from the FIFO is to be used. 00 Use each sample once 01 Use each sample twice 10 Use each sample four times 11 Use each sample eight times
0 EN	PWM Enable. This bit enables the PWM. If this bit is not enabled, the clock prescaler and the counter is reset. When the PWM is enabled, it begins a new period, the output pin is set to start a new period while the prescaler and counter are released and counting begins. 0 PWM disabled 1 PWM enabled

42.4.2 PWM Status Register (PWMx_PWMSR)

The PWM status register (PWM_PWMSR) contains seven bits which display the state of the FIFO and the occurrence of rollover and compare events. The FIFOAV bit is read-only but the other four bits can be cleared by writing 1 to them. The FE, ROV, and CMP bits are associated with FIFO-Empty, Roll-over, and Compare interrupts, respectively.

Addresses: PWM-1_PWMSR is 53FB_4000h base + 4h offset = 53FB_4004h

PWM-2_PWMSR is 53FB_8000h base + 4h offset = 53FB_8004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0									FWE	CMP	ROV	FE	FIFOAV		
W										w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

PWMx_PWMSR field descriptions

Field	Description
31–7 Reserved	This read-only field is reserved and always has the value zero. Reserved. These reserved bits are always read as zero.

Table continues on the next page...

PWMx_PWMSR field descriptions (continued)

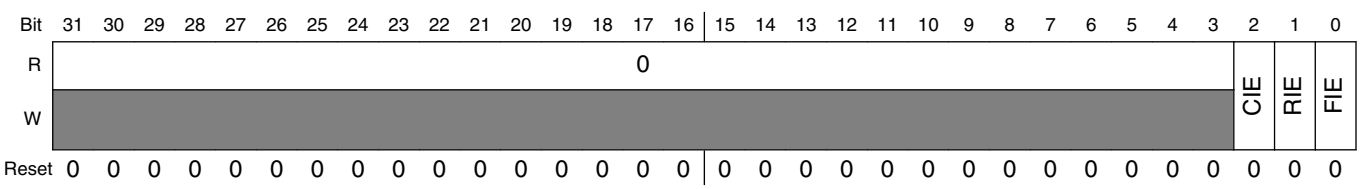
Field	Description
6 FWE	FIFO Write Error Status. This bit shows that an attempt has been made to write FIFO when it is full. 0 FIFO write error not occurred 1 FIFO write error occurred
5 CMP	Compare Status. This bit shows that a compare event has occurred. 0 Compare event not occurred 1 Compare event occurred
4 ROV	Roll-over Status. This bit shows that a roll-over event has occurred. 0 Roll-over event not occurred 1 Roll-over event occurred
3 FE	FIFO Empty Status Bit. This bit indicates the FIFO data level in comparison to the water level set by FWM field in the control register. 0 Data level is above water mark 1 When the data level falls below the mark set by FWM field
2–0 FIFOAV	FIFO Available. These read-only bits indicate the data level remaining in the FIFO. An attempted write to these bits will not affect their value and no transfer error is generated. 000 No data available 001 1 word of data in FIFO 010 2 words of data in FIFO 011 3 words of data in FIFO 100 4 words of data in FIFO 101 unused 110 unused 111 unused

42.4.3 PWM Interrupt Register (PWMx_PWMIR)

The PWM Interrupt register (PWM_PWMIR) contains three bits which control the generation of the compare, rollover and FIFO empty interrupts.

Addresses: PWM-1_PWMIR is 53FB_4000h base + 8h offset = 53FB_4008h

PWM-2_PWMIR is 53FB_8000h base + 8h offset = 53FB_8008h



PWMx_PWMIR field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value zero. Reserved. These reserved bits are always read as zero.
2 CIE	Compare Interrupt Enable. This bit controls the generation of the Compare interrupt. 0 Compare Interrupt not enabled 1 Compare Interrupt enabled
1 RIE	Roll-over Interrupt Enable. This bit controls the generation of the Rollover interrupt. 0 Roll-over interrupt not enabled 1 Roll-over Interrupt enabled
0 FIE	FIFO Empty Interrupt Enable. This bit controls the generation of the FIFO Empty interrupt. 0 FIFO Empty interrupt disabled 1 FIFO Empty interrupt enabled

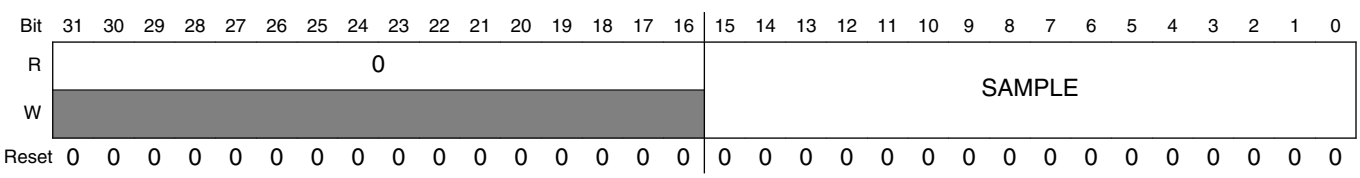
42.4.4 PWM Sample Register (PWMx_PWMSAR)

The PWM sample register (PWM_PWMSAR) is the input to the FIFO. 16-bit words are loaded into the FIFO. The FIFO can be written and read when the PWM is disabled. The PWM will run at the last set duty-cycle setting if all the values of the FIFO has been utilized, until the FIFO is reloaded or the PWM is disabled. When a new value is written, the duty cycle changes after the current period is over.

A value of zero in the sample register will result in the ipp_pwm_pwmo output signal always being low/high (POUTC = 00 it will be low and POUTC = 01 it will be high), and no output waveform will be produced. If the value in this register is higher than the PERIOD + 1, the output will never be set/reset depending on POUTC value.

Addresses: PWM-1_PWMSAR is 53FB_4000h base + Ch offset = 53FB_400Ch

PWM-2_PWMSAR is 53FB_8000h base + Ch offset = 53FB_800Ch



PWMx_PWMSAR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 SAMPLE	Sample Value. This is the input to the 4x16 FIFO. The value in this register denotes the value of the sample being currently used.

42.4.5 PWM Period Register (PWMx_PWMPR)

The PWM period register (PWM_PWMPR) determines the period of the PWM output signal. After the counter value matches PERIOD + 1, the counter is reset to start another period.

PWMO (Hz) = PCLK(Hz) / (period +2)

A value of zero in the PWM_PWMPR will result in a period of two clock cycles for the output signal. Writing 0xFFFF to this register will achieve the same result as writing 0xFFFE.

A change in the period value due to a write in PWM_PWMPR results in the counter being reset to zero and the start of a new count period.

Addresses: PWM-1_PWMPR is 53FB_4000h base + 10h offset = 53FB_4010h

PWM-2_PWMPR is 53FB_8000h base + 10h offset = 53FB_8010h

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

PWMx_PWMPR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 PERIOD	Period Value. These bits determine the Period of the count cycle. The counter counts up to [Period Value] +1 and is then reset to 0x0000.

42.4.6 PWM Counter Register (PWMx_PWMCNR)

The read-only pulse-width modulator counter register (PWM_PWMCNR) contains the current count value and can be read at any time without disturbing the counter.

Addresses: PWM-1_PWMCNR is 53FB_4000h base + 14h offset = 53FB_4014h

PWM-2_PWMCNR is 53FB_8000h base + 14h offset = 53FB_8014h

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

PWMx_PWMCNR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. These are reserved bits and writing a value will not affect the functionality of PWM and are always read as zero.
15–0 COUNT	Counter Value. These bits are the counter register value and denotes the current count state the counter register is in.

Chapter 43
Enhanced Pixel Pipeline (ePXP)

43.1 Overview

The enhanced pixel pipeline is used to perform alpha blending of graphic or video buffers with graphics data before sending to an LCD display or TV encoder. The ePXP also supports image rotation for hand-held devices that require both portrait and landscape image support.

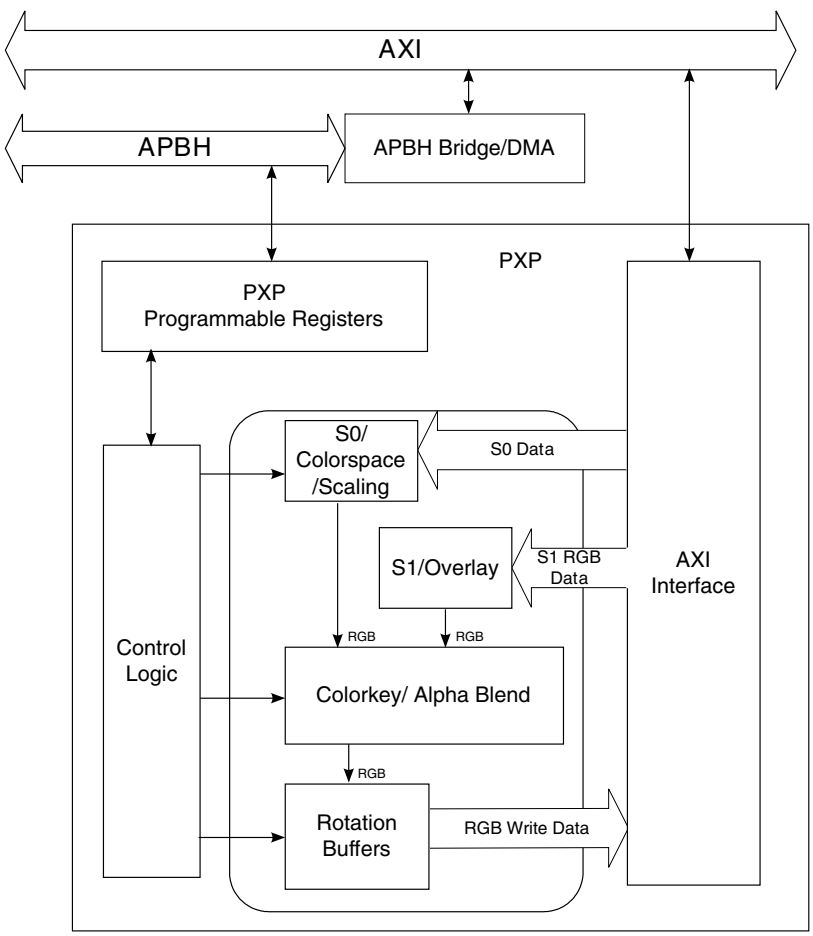


Figure 43-1. ePXP Block Diagram

The ePXP is organized as having a background image (S0) and one or more overlay images that can be blended with the background. Each overlay image must be a multiple of the block size in pixels in both height and width and the offset of the overlay into the background image must also be a multiple of the block size in pixels. As the ePXP processes data, it reads each NxN block from the background image and finds the highest priority (lowest numbered) overlay that is co-located at that block coordinate. The ePXP then fetches the overlay and performs the alpha blending and color key operations on the two blocks. The resulting pixel block is then written to the corresponding block in the output buffer.

For the S0 plane, the ePXP supports RGB images (unscaled) or color space conversion (YUV->RGB) and scaling of YUV images. The S1 plane consists of up to eight overlay regions consisting of 16- or 32-bit RGB data. The S0 and S1 planes may then be combined by alpha blending, color key substitution, or raster operations (ROPs) to form the output image. Finally, the resulting image may be clockwise rotated in 90 degree increments and/or flipped horizontally and/or vertically. The ePXP also supports letterboxing and interlacing of progressive content (by writing alternate lines to different frame buffers).

The flow of data through the ePXP is shown below.

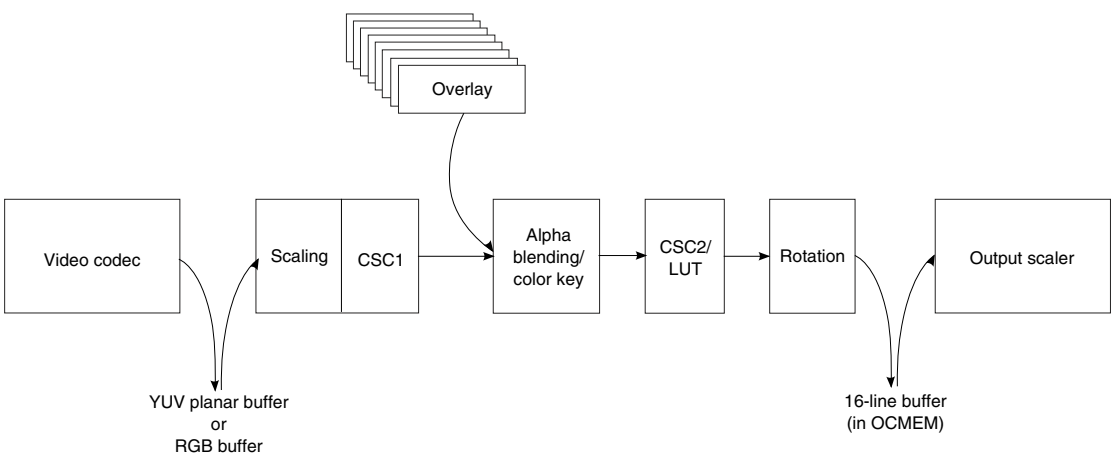


Figure 43-2. ePXP Data flow

43.1.1 Image Support

The ePXP's S0 buffer supports the following image formats:

- 24-bit unpacked RGB (32bpp)
- 24-bit packed RGB (24bpp)

- 16-bit RGB in either 555 or 565 format
- YUV/YCbCr 4:2:2 1/2/3 plane format
- YUV/YCbCr 4:2:0 2/3 plane format

The ePXP's S1 buffer supports the following image formats:

- 32-bit RGB (with or without alpha)
- 16-bit RGB in either 555, 565, or 1555 (alpha)

The ePXP's output buffer supports:

- 32-bit RGB (with alpha)
- 24-bit packed RGB (24bpp)
- 16-bit RGB in either 565, 555, or 1555 format
- YUV 4:4:4/4:2:2 1-plane
- YUV 4:2:2/4:2:0 2-plane
- Y8/4 1-plane monochrome
- Interlaced output processing

Internally, all image data is handled as 32bpp data (either RGB or YUV, depending on output mode selection) for all steps after the color space conversion. Input RGB images are always converted to the equivalent 32bpp format before processing.

43.1.2 Block Size Selection

The ePXP can be configured to process blocks that are either 8 x 8 pixels or 16 x 16 pixels. The granularity of image size and location of video or overlay buffers within the final destination frame buffer has twice the precision when selecting 8 x 8 pixel block sizes. When selecting a 16 x 16 pixel block size, the accesses to fetch S0 and S1 images and write the final frame buffer are more efficient since twice as much data is requested and processed per memory request. When optimizing the system for memory bandwidth and image processing time, configure the ePXP to process 16x16 pixel blocks.

The control registers that are in block size units need to be programmed consistently with the ePXP_CTRL[BLOCK_SIZE] control bit setting. If the source image is 32 x 32 pixels, then the width and height setting will be 4 when selecting 8x8 pixel block size and 2 when selecting 16 x 16 block size.

43.1.3 ePXP Limitations/Issues

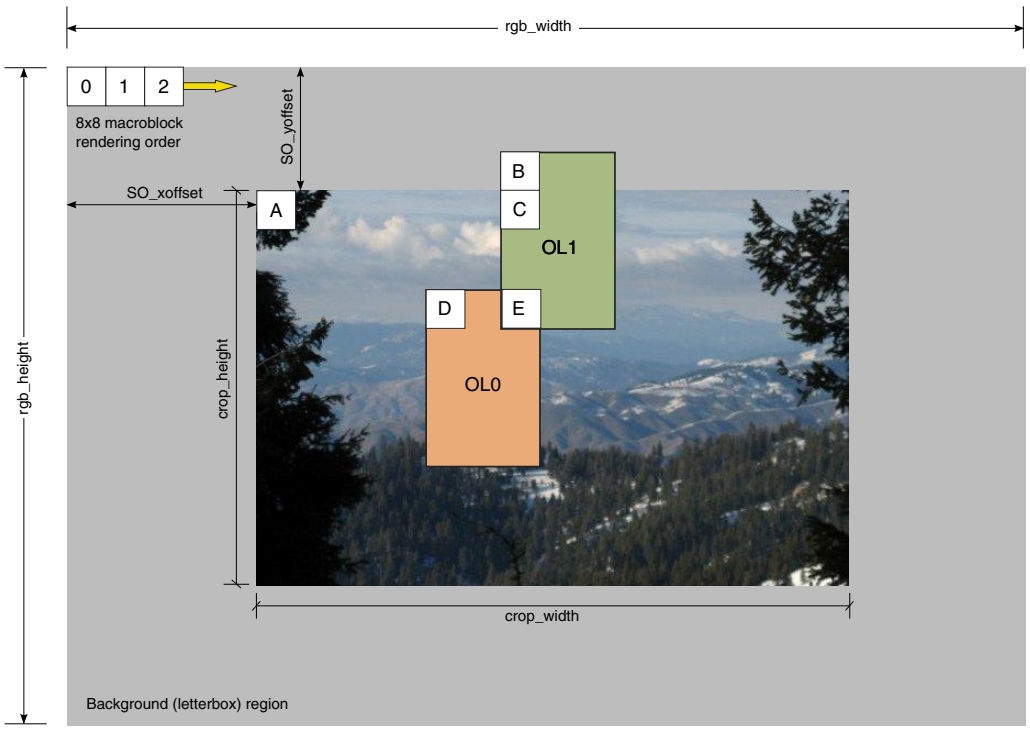
- The ePXP's scalar uses a bilinear scaling algorithm and can scale YUV images from 1/4x to 4096x in 12-bit fractional steps.

- When using the NEXT register, the interrupt enable setting should remain the same for all frames. If not, the ePXP will change the interrupt enable register value and possibly cause the loss of an interrupt.
- The ePXP cannot rotate/flip video in the interlaced modes.
- When performing input interlacing, the input image and overlays must be multiples of 8 x 16 (in 8x8 block size mode) or 16 x 32 (16 x 16 pixel blocks) pixels. Overlays must also reside on the same boundaries.
- Rotations of 180/270 are not supported when performing LCD handshakes.

43.2 Operation

The ePXP operates by rendering the output frame buffer in 8x8 or 16x16 pixel macroblocks in display order (left to right, then top to bottom). At each output macroblock location, the ePXP determines whether the S0 buffer is visible based on the cropping register and S0 offset parameters. If the S0 plane is visible, the ePXP will fetch and process the required data from the S0 image, otherwise, the S0's contribution to the output macroblock will be the ePXP_S0BACKGROUND register value. This value is effectively the color of the letterboxed region or background color.

The ePXP will also determine if an overlay is present for that macroblock location, and if so, instruct the S1 buffer to fetch the required data. If multiple overlays cover the macroblock, the ePXP will select only the lowest numbered overlay and direct the S1 buffer to load the data for this overlay. For areas with no overlays, the S1 buffer contributes nothing to the rendered image. The following figure shows the order in which the output blocks are generated (blocks 0, 1, 2) and indicates how various blocks are rendered (blocks A-E).



0,1,2: Pixel blocks rendered with the background color. The numbering and arrow indicate the order of macroblock rendering.
A: SO Image rendered.
B: Overlay 1 blended with background.
C: Overlay 1 blended with SO image.
D: Overlay 0 blended with SO image.
E: Overlay 0 blended with SO image (OL0 takes precedence over OL1).

Figure 43-3. Output Blocks

It is important to understand how the ePXP renders each output macroblock to properly understand how it accomplishes cropping, letterboxing, and overlay blending. The following sections will provide more details on these operations.

The ePXP also has the ability to rotate/flip images for cases when the pixel scan order is not in the traditional left-to-right/top-to-bottom raster scan (landscape raster). This can occur when a handheld device with a traditional landscape scan is rotated into a portrait orientation (in which the scan order is now bottom-to-top/left-to-right or vice versa) or when a cell phone oriented display (portrait raster) is rotated into a landscape orientation for viewing videos. In these cases, the ePXP still renders the image in scan-order format (as sent to the device), but it will traverse the input images based on the transformations required.

The following sections detail each of the ePXP's functional capabilities.

43.2.1 Pixel Handling

All pixels are internally represented as 24-bit RGB or YUV/YCbCr values with an 8-bit alpha value at all stages in the ePXP after the color space converter (CSC). The output format selected determines the colorspace representation of pixels after the CSC stage of the pixel pipeline. It should be noted that pixels in the YUV/YCbCr color space cannot be blended with S1 overlay pixels since this channel only processes RGB pixel formats.

Input pixels are converted into the RGB format using the following rules:

- 32-bit ARGB8888 pixels are read directly with no conversion for both the S0 and overlay images.
- 32-bit RGB888 pixels are assumed to have an alpha value of 0xFF (full opaque).
- 16-bit RGB565 and RGB555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of 0xFF (opaque). The expansion process replicates the upper pixel bits into the lower pixel bits (for instance a 16-bit RGB565 triplet of 0x1F/0x20/0x07 would be expanded to 0xFF/0x82/0x39).
- 16-bit RGB1555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of either 0x00 or 0xFF, based on the 1-bit alpha value in the pixel. The ALPHA_MULTIPLY function is useful in this scenario to allow scaling of the opaque pixels to a semi-transparent value.

Input pixels are processed in the YUV/YCbCr format using the following rules:

- All pixels are processed through the scaling engine for conversion to the YUV/YCbCr 4:4:4 24-bit format.
- When not resizing the input image, the input pixels are still processed through the scaling engine to convert 4:2:2 or 4:2:0 formats to 4:4:4 formats. Essentially, the chroma values are resized to contain a unique chroma sample for each pixel site.
- S1, or overlays cannot be supported in this case since S1 RGB pixels cannot be converted to YUV/YCbCr.

Output pixels will retain the effective alpha value of the overlay or can be set to a programmed alpha value using the ALPHA field of the ePXP_OUTSIZE register. 16-bit pixels values are formed from the most significant bits of the 24-bit pixel values.

43.2.2 S0 Cropping/Masking

The ePXP's cropping operation should be viewed as a mask on the output image through which the background S0 plane can be viewed. Using this definition clarifies a subtlety on the usage of cropping an image when the image is scaled. When scaling is not used, the input and output image sizes are the same, therefore, the operation is analogous to cropping the input source image.

The background output image can be cropped to a width and height independent of the image size at a given offset into the image (all sizes are in terms of block size pixel units) using the values in the ePXP_S0CROP register. The XBASE and YBASE provide the coordinates of the first block to be displayed from the source image and the WIDTH and HEIGHT parameters specify an effective size of the resulting image in the output buffer.

Cropping must be enabled by setting the CROP bit in the ePXP_CTRL register to a 1. When not set, the visible portions of the S0 image will be rendered based on the WIDTH and HEIGHT specified in the S0SIZE field. The following figure indicates how the various cropping parameters relate to the source and RGB images (non-scaled case).

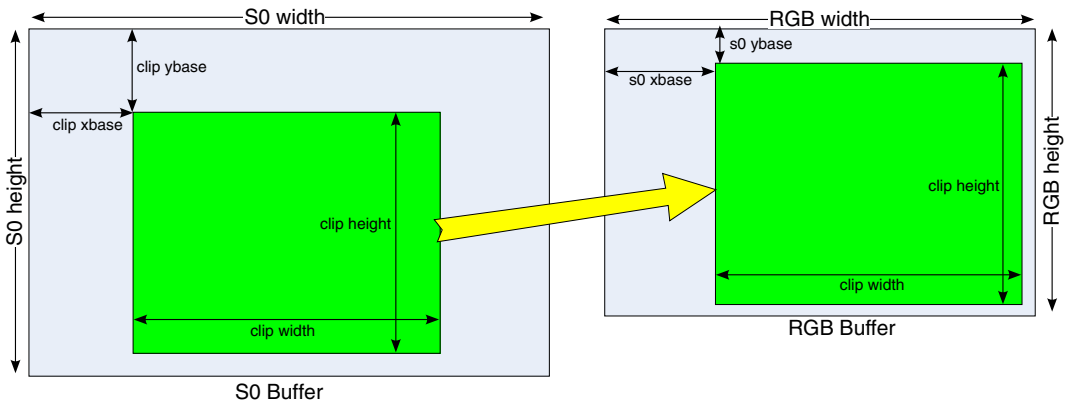


Figure 43-4. Cropping Parameters

It is important to note that when scaling an image, software must specify a valid cropping region since the ePXP will default to using the source image size. When downscaling, this is not an issue, but with upscaling the resulting image will be a scaled up version of the source, but cropped to the same size as the source image as shown below.

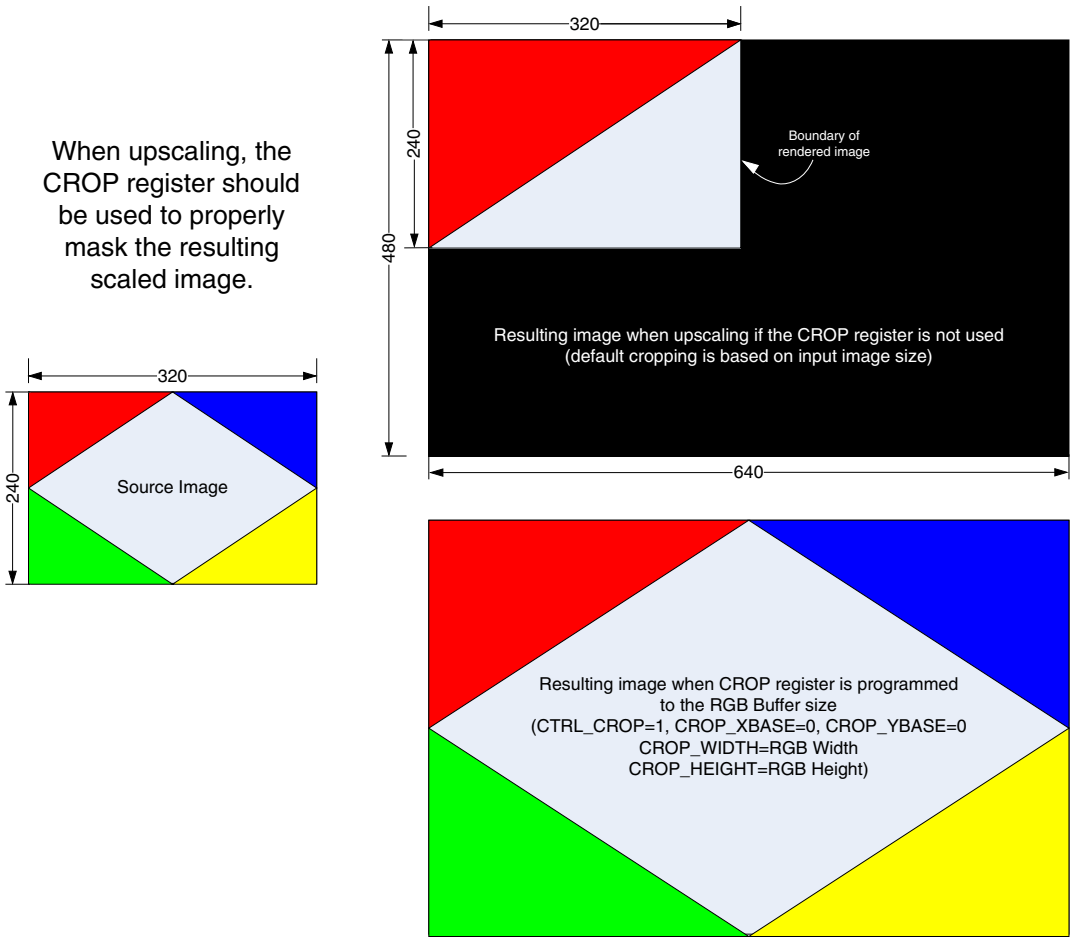


Figure 43-5. Upscaling

The cropping extents should fall completely within the S0 buffer to avoid displaying incorrect data. The ePXP hardware does not check for these conditions and will render the image as shown in the following two diagrams. (Note that the cropping width and height can be viewed as applying to the input buffer only because it is not scaled. In actuality, it is applied to the output buffer).

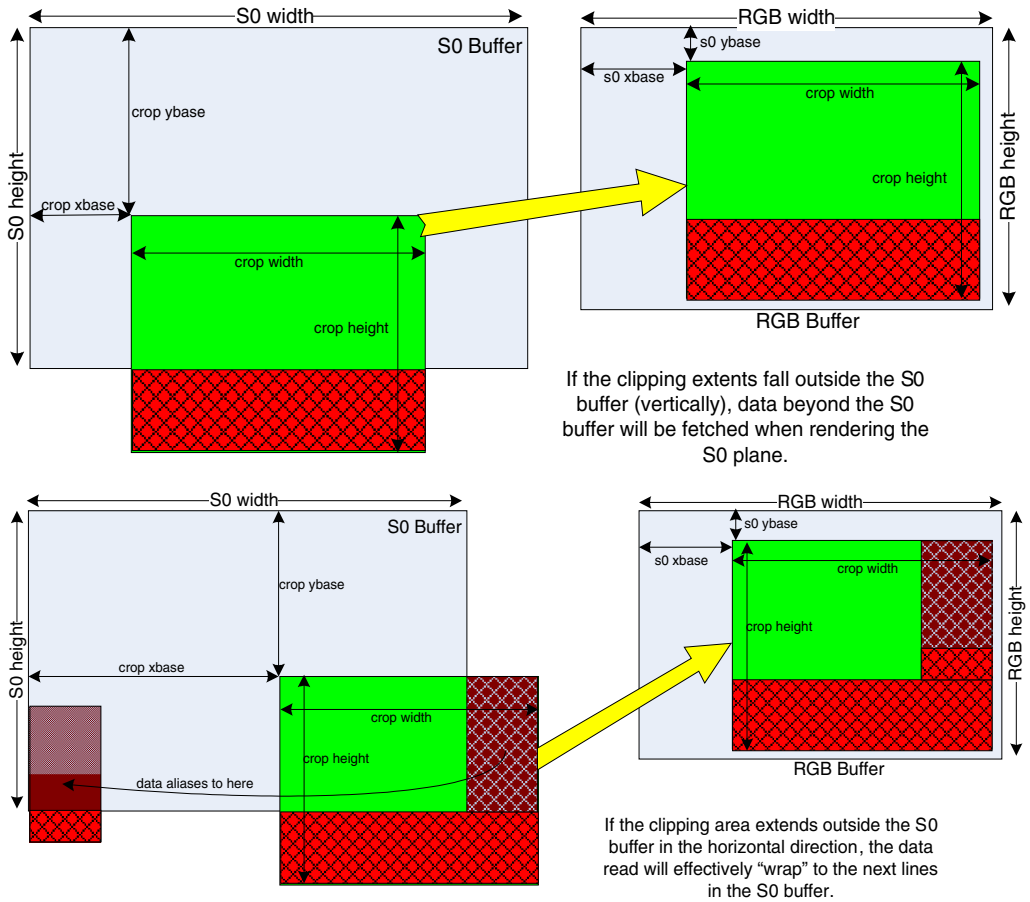


Figure 43-6. Rendering with Extents Ouside of S0 Buffer

43.2.3 Scaling

The ePXP can scale YUV images from 1/4x to about 4096x using a bilinear scaling algorithm. The hardware is capable of scaling with 12-bit fractional resolution, or in 1/4096th pixel increments with independent scaling ratios for the X and Y direction. The scaler also implements an initial offset, which can be useful when scaling by powers of 2 in order to ensure that the resulting pixels are averages (select ½ pixel offset) of the source pixels instead of producing a decimated or replicated image. Also, the offset can be used to achieve per pixel addressing on the input source image.

The scaling parameter is specified to the hardware in terms of the inverse of the scaling ratio desired. This can also be viewed as the step size between computed sample values. For instance, when scaling by 2x the inverse is 1/2, therefore the scaler will increment by 1/2 pixel steps across the input image and compute the bilinear average for each sample point.

The scaling values are represented by 12-bit fractional values in the scaling register and hardware. The scaling ratios are computed as the input size divided by the output size. The resulting decimal value must then be converted into a 12-bit fixed point value by multiplying by 212 or 4096 to produce the value programmed into the scaling registers.

To scale an image from 400 x 300 to 320 x 200, the horizontal XSCALE factor is computed as:

XSCALE = (Input Size / Output Size) x 4K = 400/320 x 4K = 5120 = 0x1400

The vertical YSCALE can be similarly computed as:

YSCALE = (Input Size/Output Size) x 4 K = 300/200 x 4K = 6144= 0x1800

The scaler will use the CROP_XBASE and CROP_YBASE values as an offset into the source S0 image for the origin of the input image to be scaled. The CROP_WIDTH and CROP_HEIGHT parameters will be used to determine the extent of the scaled image in the output buffer. It is tempting to view the cropping width and height as being applied to the input buffer, but this is incorrect -- the ePXP uses these values as a mask on the output buffer to determine which regions of the output buffer require data from the scaled input image.

To enable scaling, the ePXP_CTRL[SCALE] bit must be set and the desired scaling ratios written into the ePXP_S0SCALE registers. Initial offsets should be programmed into the ePXP_S0OFFSET register.

43.2.3.1 Bilinear Image Scaling Filter

The ePXP implements a bilinear scaling filter to resize an input image to a different resolution for display output. The bilinear filter is a weighted average of the four nearest pixels that can be sourced to approximate the pixel in the output frame buffer. It should be noted that when scaling from a large image to a smaller image by a factor of at least 2, then some lines will be decimated. In the extreme case of a 4:1 reduction in image size, half the lines will be decimated unless a two pass scaling technique is employed. This decimation results in a bilinear filtering algorithm that uses 2 pixels while stepping 4 pixels for each scaling increment.

When scaling YUV data, the UV values are offset by 0x80 (top bit inverted) to shift the signed UV bits into an unsigned equivalent with a range of 0 to 255. YCbCr data does not have to be shifted since it is defined as an unsigned byte. The YCBCR_MODE bit of the COEFF0 register controls whether this operation is applied to the input UV bytes.

After scaling, the offset is removed so that the range for UV data is signed from -128 to 127.

The reason for this adjustment is based on the implementation of an unsigned scaling engine, and therefore, is to ensure that the scaled values are handled properly. Consider the following table:

Format	Pixel0	Pixel1	Average	Result
decimal	-2	+2	0	Correct
CbCr	0x7E	0x82	0x80	Correct (0x80 is 0 in CbCr)
UV	0xFE	0x02	0x80	Incorrect (0x80 is -128 in UV)
decimal	-32	+16	-8	Correct
CbCr	0x60	0x90	0x78	Correct (0x78 is -8 in CbCr)
UV	0xE0	0x10	0x78	Incorrect (0x78 is +120 in UV)

To compute the output pixel value at position as indicated by P, consider the diagram below.

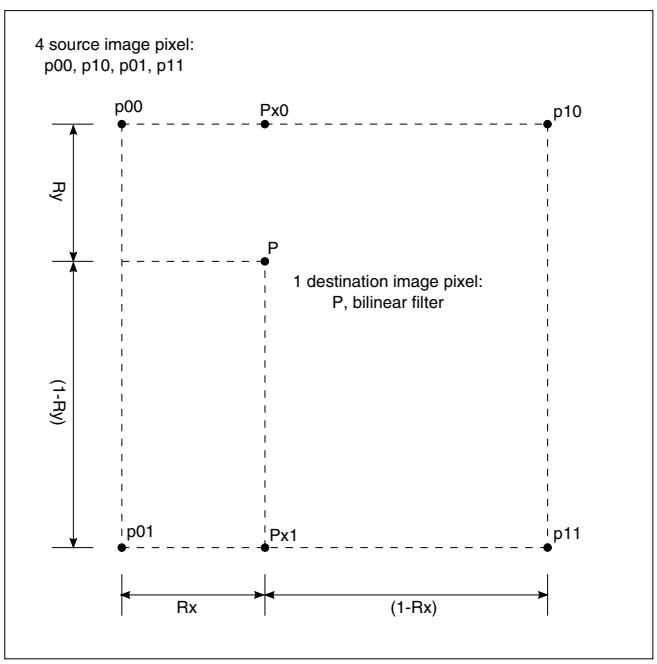


Figure 43-7. Output Pixel Computation

A step function is used to indicate the position of the pixel "P" in the output frame. This position may not coincide with a single pixel position in the input frame buffer. In this case, the four closest pixels in the input frame are used to approximate the value of the pixel in the output frame.

The ePXP scaler first computes a linear filter in the X axis to create the two intermediate pixel values Px0 and Px1. The step function's X fractional component is used to provide the weighting factor for blending p00 with p10 to provide Px0. Likewise, Px1 is also derived from a linear filter using p01 and p11.

The equations for Px0 and Px1 are as follows:

Px0 = p00 x (1 - Rx) + p10 x Rx

Px1 = p01 x (1 - Rx) + p11 x Rx

The ePXP scaler uses the intermediate X pixels Px0 and Px1 and implements a bilinear filter on these two pixel values to produce the final pixel value at position P. The remainder of the step function for the Y axis is used to compute the weighted average pixel result. The equation for final filtered pixel is:

P = Px0 x (1 - Ry) + Px1 x Ry

43.2.3.2 YUV 4:2:2 Image Scaling

The following figure illustrates the positioning of YUV samples for the 4:2:2 formats. There are twice as many Y luma samples then U and V chroma samples horizontally.

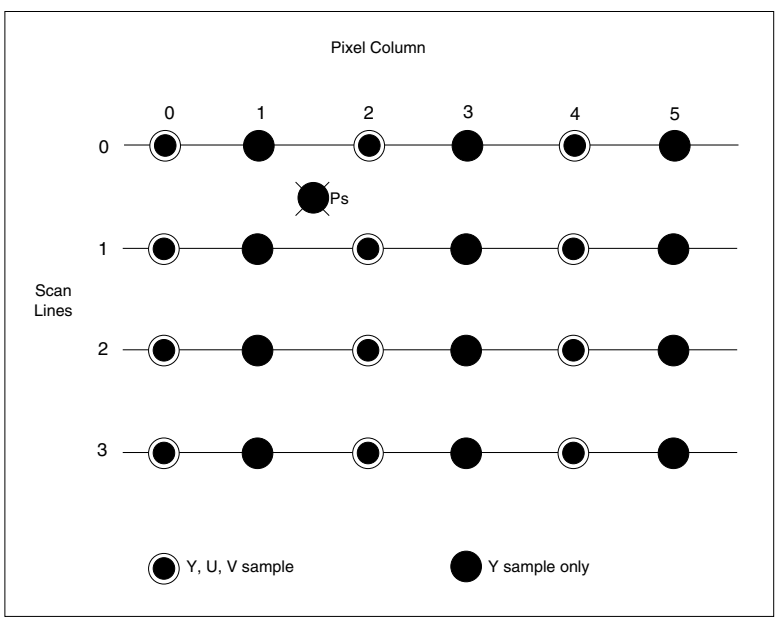


Figure 43-8. YUV Samples, 4:2:2

Consider the scaled output pixel Ps (pixel scaled) which has an accumulated step function of X=1.5 and Y=0.5. The remainder for the step function is Rx = 0.5 and Ry = 0.5. Or, the sub pixel position of output pixel Ps is half way between line 0 and 1 and half way between column 1 and 2.

The Y output component of Ps is simply the bilinear function of the four nearest Y samples from the input image. Specifically, the Y values at [1,0], [2,0], [1,1], and [2,1] are used to compute the Y for Ps.

For the U and V components of Ps, there are no samples present in the column position 1. The bilinear filter uses chroma components located at [0,0], [2,0], [0,1] and [2,1]. Since the chroma components are not sub sampled vertically, the remainder used to combine pixels vertically is Ry=0.5 (the same as for Y). However, horizontally, the scaling engine shifts the remainder by a factor of 2. So an X axis step function value of X=1.5 has a remainder Rx=0.75. Source chroma values are not replicated, they are completely interpolated using the four nearest chroma samples to approximate U and V at Ps.

43.2.3.3 YUV 4:2:0 Image Scaling

The following figure illustrates the positioning of YUV samples for the 4:2:0 formats. Chroma is sub sampled both horizontally and vertically. In this format, the chroma frame buffers contain ¼ the data that the luma frame buffers store.

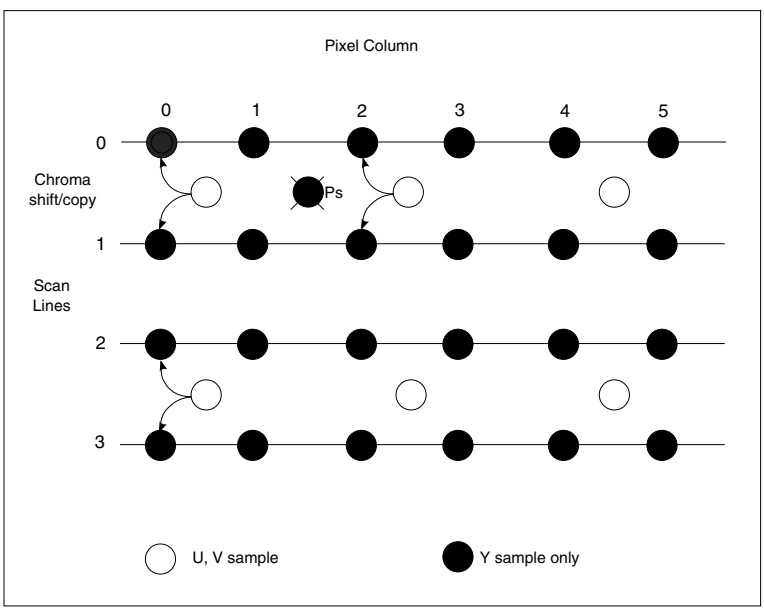


Figure 43-9. YUV Samples, 4:2:0

43.2.4 Color Space Conversion (CSC)

There are two modules in the ePXP to convert pixels between color spaces. They are referred to as CSC1 and CSC2 (for lack of a better naming convention). CSC1 exists after the scaling unit and is dedicated to converting from YUV to RGB. CSC2 is a full duplex color space converter in that is can convert into either RGB or YUV (or YCbCr) color

spaces depending on the desired output pixel format. All coefficients are programmed as two's complement numbers and both CSC units can be bypassed if CSC is not desired at either position of these CSC units in the pixel data path.

43.2.4.1 CSC1 Operation

The CSC1 module receives scaled YUV/YCbCr444 pixels from the scale engine and converts the pixels to the RGB888 color space only if CSC1 is enabled. The CSC1 module will convert only to the RGB color space and it can be bypassed to allow YUV pixels through the data path. These pixels are loaded into the pixel FIFO for processing by subsequent modules in the pixel data path.

The following equations are used to perform YUV/YCbCr -> RGB conversion. The constants will be stored in the ePXP control registers as two's complement values to allow flexibility in the implementation and to allow for differences in the video encode and decode operations. In addition, this provides a software mechanism to manipulate brightness or contrast.

R = C0(Y + Yoffset) + C1(V + UVoffset)

G = C0(Y + Yoffset) + C3(U + UVoffset) + C2(V + UVoffset)

B = C0(Y + Yoffset) + C4(U + UVoffset)

Note: In the equations above, U and V are synonymous with Cb and Cr in regards to the color space format of the source frame buffer.

Saturation of each color channel is checked and corrected for excursions outside the nominal YUV/YCbCr color spaces. Overflow for the three channels are saturated at 0x255 and underflow is saturated at 0x00.

The table below indicates the expected coefficients for YUV and YCbCr modes of operation:

Table 43-2. Coefficients for YUV and YCbCr Operation

Coefficient	YUV	YCbCr
Yoffset	0x000	0x1F0 (-16)
UVoffset	0x000	0x180 (-128)
C0	0x100 (1.00)	0x12A (1.164)
C1	0x123 (1.140)	0x198 (1.596)
C2	0x76B (-0.581)	0x730 (-0.813)
C3	0x79B (-0.394)	0x79C (-0.392)
C4	0x208 (2.032)	0x204 (2.017)

43.2.4.2 YUV versus YCbCr Support

By default, the ePXP color space coefficients are set to support the conversion of YUV data to RGB data. If YCbCr input is present, software must change the coefficient registers appropriately (see the register definitions for values). Software must also set the YCBCR_MODE bit in the ePXP_COEFF0 register to ensure proper conversion of YUV versus YCBCR data.

43.2.4.3 CSC2 Operation

The CSC2 module receives pixels in any color space and can convert the pixels into any of RGB, YUV, or YCbCr color spaces. All coefficients are programmable and in the two's complement notation. The output pixels are passed onto the LUT and rotation engine for further processing.

The following equations indicate the CSC2 modules ALU architecture.

Selecting RGB output in ePXP_CSC2CTRL[CSC_MODE] configures the ALU in the following manner:

R = A1(Y - D1) + A2(U - D2) + A3(V - D3)

G = B1(Y - D1) + B2(U - D2) + B3(V - D3)

B = C1(Y - D1) + C2(U - D2) + C3(V - D3)

Selecting YUV output configures the ALU in the alternate manner:

Y = A1 x R + A2 x G + A3 x B + D1

U = B1 x R + B2 x G + B3 x B + D2

V = C1 x R + C2 x G + C3 x B + D3

Saturation of each color channel is checked and corrected for excursions outside the nominal color space. Overflow for the three channels is saturated at 0x255 and underflow is saturated at 0x00.

The table below indicates the coefficients used to convert from RGB to YCbCr or YUV. These decimal values should be converted to their two's complement representation.

Table 43-3. Coefficients for RGB to YUV/YCbCr Conversion

Coefficient	RGB -> YUV	RGB -> YCbCr
A1, A2, A3	0.299, 0.587, 0.114	0.257, 0.504, 0.098

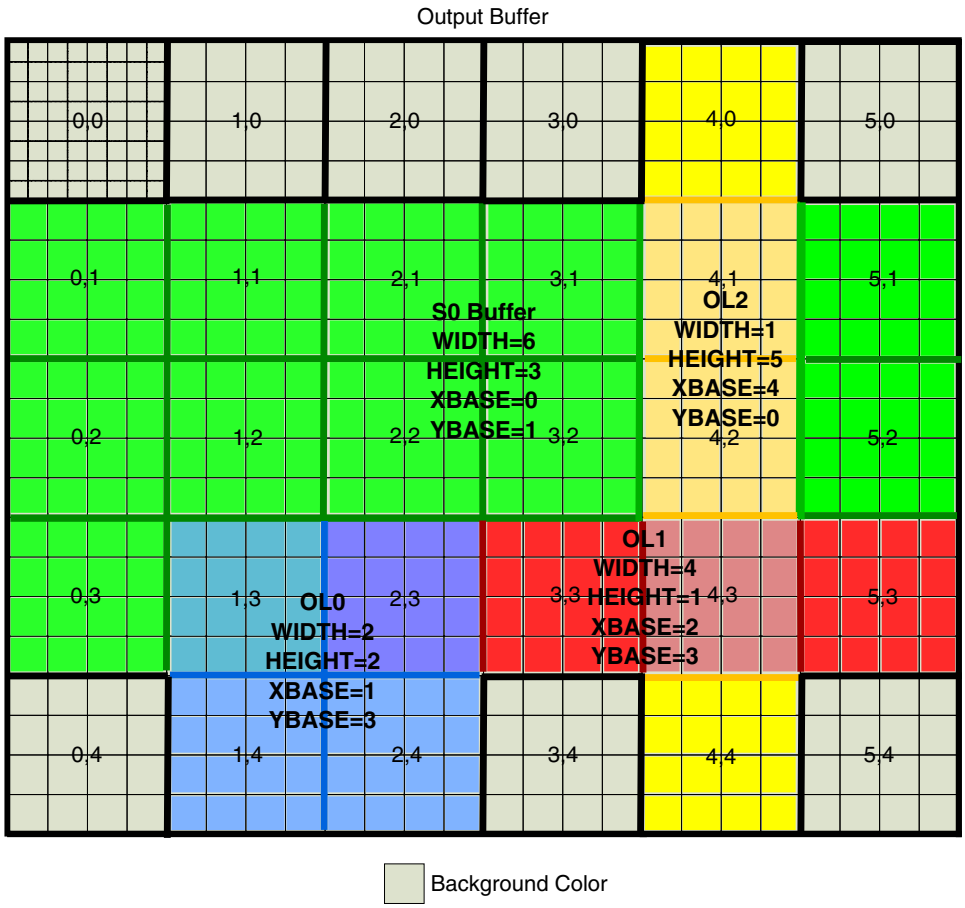
Table continues on the next page...

Table 43-3. Coefficients for RGB to YUV/YCbCr Conversion (continued)

Coefficient	RGB -> YUV	RGB -> YCbCr
B1, B2, B3	-0.147, -0.289, 0.436	-0.148, -0.291, 0.439
C1, C2, C3	0.615, -0.515, -0.100	0.439, -0.368, -0.071
D1, D2, D3	0, 0, 0	16, 128, 128

43.2.5 Overlays

The ePXP supports up to eight overlays that can be used to merge graphic data with video (or other graphic data). Each overlay consists of a rectangular area that is a multiple of Ôn' (where n is the block size) pixels in both the vertical and horizontal directions. Overlays must also be located on NxN boundaries within the output image. As the ePXP processes each NxN macroblock, it determines if any of the enabled overlays cover the block and then merges the overlay data with the background image as specified in the overlay's control registers. If multiple overlays overlap for a given NxN block, the ePXP will select the lowest numbered one for the blending operation. If the desired affect is to blend the overlays together, this can be accomplished as a multi-step process using the IN_PLACE functionality. (See In-place Rendering.)



The S0 buffer and each overlay can be placed within the output buffer using their XBASE and YBASE registers and the dimensions of each region are set using their WIDTH and HEIGHT parameters. Overlay 0 has the highest priority (effectively it is the highest in the stacking order) and the S0 buffer and background color have the lowest priority.

Overlays can be blended with the background or S0 planes, but not with each other. Effectively only a single overlay is active for each 8x8 pixel block.

Figure 43-10. Overlays

Each overlay can perform one of three classes of operations between the overlay and the underlying background (S0) image: alpha blending, color keying, or raster operations.

An overlay can be enabled by writing the address of the overlay image to the OLn register, the overlay's size and location information into the ePXP_OLnSIZE register, and then setting the OLnPARAM_ENABLE bit. The ePXP_OL_n_PARAM registers also contain further controls to select the modes of operation.

43.2.6 Alpha Blending

The alpha value for an individual pixel represents a mathematical weighting factor applied to the S1 pixel. An alpha value of 0x00 corresponds to a transparent pixel and a value of 0xFF corresponds to an opaque pixel.

The effective alpha value for an overlay pixel is determined by the ALPHA bit-field and the two ALPHA control bits in the ePXP_OLnPARAM register. If the ALPHA_CTRL field is set to ALPHA_OVERRIDE, the alpha value for the pixel is taken from the ALPHA bit-field. This can be useful for applying a constant alpha to an entire image or for image formats that do not include an alpha value. If ALPHA_MULTIPLY is selected, the pixel's alpha value will be multiplied by the ALPHA value in order to allow scaling of the pixel's alpha or to provide better control for pixel formats such as RGB1555, which only contains a single bit of alpha.

For each color channel, the equation used to blend two source pixels is defined below:

$E\alpha$ = Embedded alpha associated with S1 pixel

$\alpha = G\alpha \times E\alpha + 0x80$

$G\alpha$ = PIO programmed global alpha (8-bit value)

The result for the red channel as an example is as follows:

$Yr[7:0] = (\alpha \times S1.r) + ((1 - \alpha) \times S0.r)$

When alpha is 0xFF, the S1 pixel will not be blended with S0, but S1 will be passed as the output pixel and will not be blended with S0. In this case, S0 will be discarded. Likewise, if alpha is 0x00 for a given pixel, S0 will be loaded as the output pixel.

Alpha values in the overlays are loaded from the source image for all pixel formats. For formats that do not support an alpha value, the pixel is assigned an alpha value of 0xFF (opaque). This can be modified by the overlay processing by setting either the ALPHA_MULTIPLY or ALPHA_OVERRIDE bit in the associated ePXP_OLnPARAM register.

43.2.7 Color Key

Pixels may be made transparent to the corresponding overlay by using the S0 color key registers. If an S0 pixel matches the range specified by the ePXP_S0COLORKEYLOW and ePXP_S0COLORKEYHIGH registers, the pixel from the associated overlay will be displayed. If no overlay is present for that block, a black pixel will be generated since the default overlay pixel is 0x00000000 (transparent black pixel).

The most common use for this is when a bitmap does not support an alpha-field or for applications such as "green screen" where an image is substituted for a solid background color as shown below.

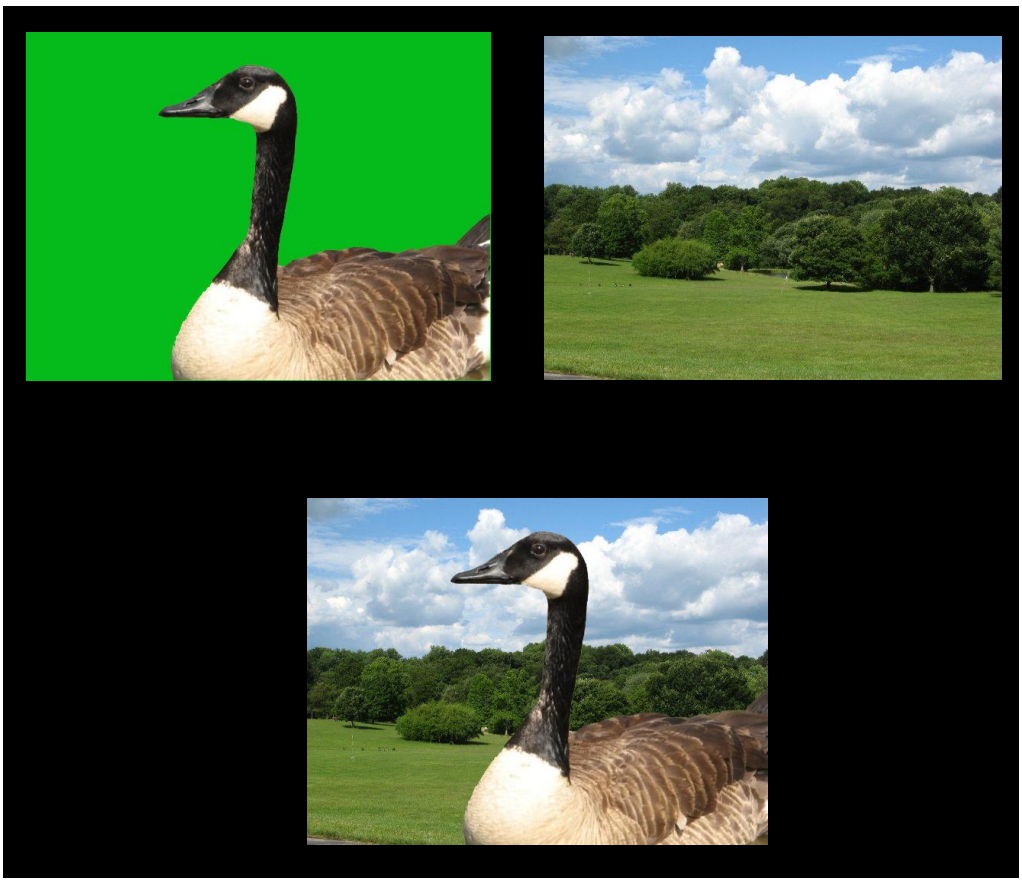


Figure 43-11. Background Image Substitution

The green portion of the overlay image can be color keyed to display the contents of the S0 buffer for locations that match the color range. For this example, the color range is

OL Colorkey: 00<R<80 70<G<ff 00<B<80

Conversely, background color keying could also have been used if the images had been swapped.

If color keying is enabled for an overlay, any pixels matching the color key parameters will be handled as color keyed pixels. Non-matching pixels will be alpha blended or handled by ROP operations as normal.

43.2.8 Raster Operations (ROPs)

In addition to alpha blending and color keying, the ePXP's alpha blender also supports a set of raster operations that may be performed between the active overlay and the background image. The operations are done on a per-pixel basis and are performed using the 24-bit overlay and background image values. The following table lists the supported ROP operations

Table 43-4. Supported ROP Operations

Mnemonic	Value	Operation
MASKOL	0x0	OL & S0
MASKNOTOL	0x1	~OL & S0
MASKOLNOT	0x2	OLL & ~S0
MERGEOL	0x3	OL | S0
MERGEOLNET	0x4	~OL | S0
MERGEOLNOT	0x5	OL | ~S0
NOTCOPYOL	0x6	~OL
NOT	0x7	~S0
NOTMASKOL	0x8	~(OL & S0) (nand)
NOTMERGEOL	0x9	~(OL | S0) (nor)
XOROL	0xA	OL ^ S0 (xor)
NOTXOROL	0xB	~(OL ^ S0) (xnor)

These operations are specified in the overlay's ePXP_PARAM register and must be enabled by setting the ALPHA_CTRL field to ROPs.

43.2.9 Lookup Table (LUT) Memory

The gamma correction lookup table is used to modify pixels in a manor that is not linear and that cannot be achieved by the color space conversion modules. Nonlinear response to input pixels can be achieved based on how the lookup table is programmed. Only the third byte is process by the lookup table. The third byte is represented by the Y value or the R value in the data path since pixel data is either YUV[23:0] or RGB[23:0] where the Y or R byte encompass bits [23:16] respectfully. So, bits 16:0 are always bypassed. Currently, the LUT is intended to process Y data when either of the monochrome output pixel formats are selected. However, this resource can be enabled and used for any conceivable purpose.

Some applications for the LUT are:

- Gamma Correction

- Pixel inversion, Converting black text on a white page to white on black.
- Pixel rounding and conversion to alternate pixel formats, like Y8 to 5-bit monochrome format.

43.2.10 Histogram

The histogram feature provides statistics about pixel values for a complete ePXP operation. The histogram logic compares the Y channel pixels at the output of the LUT resource against the values programmed in the histogram parameter registers. If the LUT is bypassed, the high order byte, or Y byte, is still processed by the histogram logic, so the histogram comparison is not dependent on using the LUT resource. Generally, this resource is used to produce statistics when the output mode is monochrome. It can be used to determine if the output pixels are limited to a specific set of values for a given ePXP process.

43.2.11 Rotation

Rotation is an inherently inefficient operation, especially for a graphics device operating in a raster-scan fashion since the resulting memory fetches would be non-contiguous. The ePXP solves this problem by operating on NxN pixel blocks. This allows the ePXP to rotate a subportion of the image, where it can fetch N lines of pixels, process them, and then write N lines of pixels regardless of the rotation orientation.

Rotation is mainly useful for reorganizing the frame buffer for handheld LCD displays for cases when the user rotates the device from a portrait to landscape orientation. Consider the following scenario:

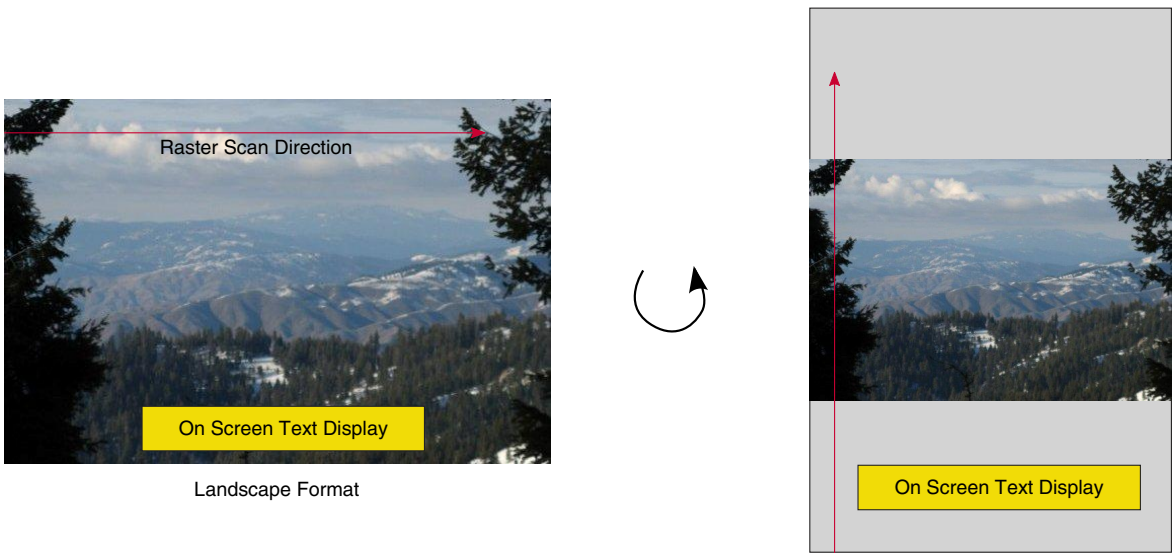


Figure 43-12. Portrait to Landscape

While this looks like a trivial operation, consider what the frame buffer must look like in memory before being sent to the LCD in raster-scan format

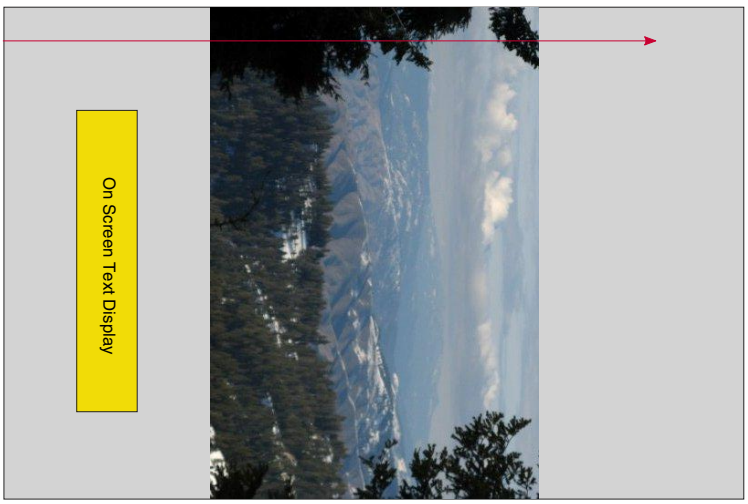
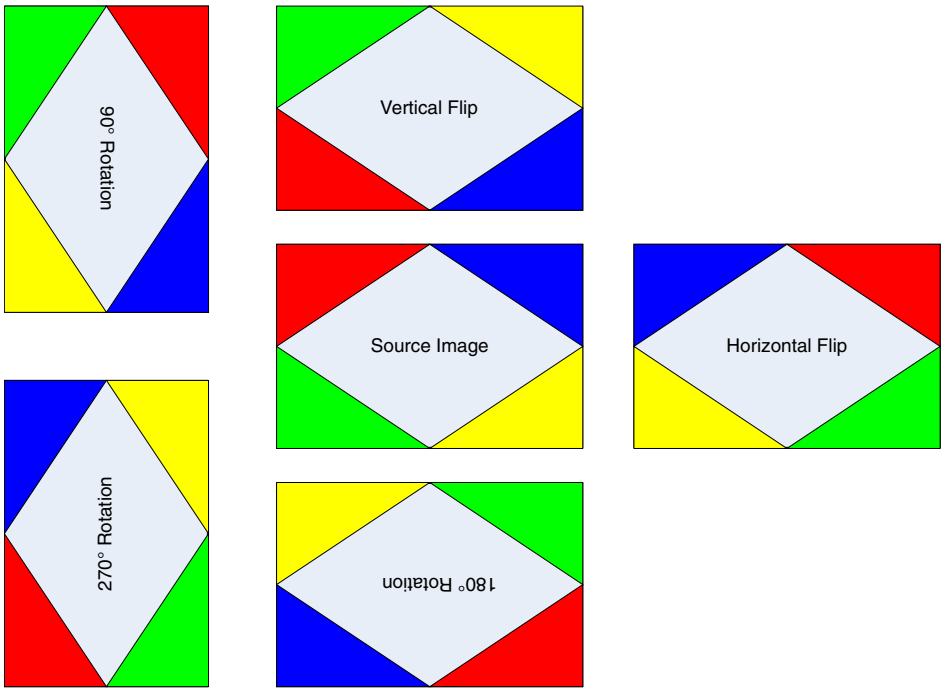


Figure 43-13. Frame Buffer in Portrait

Not only must the image be rotated, but any on-screen graphics must also be rendered in a different orientation. By building rotation into the rendering process, the ePXP allows software to construct the image in the traditional portrait format and simply rotate the image/overlays for the LCD interface during composition.

The rotation operations are defined as rotations in a clockwise direction and the flip operations will flip the pixels in the specified direction.

Figure 43-14. Rotation and Flip Operations



The ePXP supports rotation in 90 degree increments as well as horizontal and vertical flip operations. These can be done in any combination (for example, 90 degree rotation with both vertical and horizontal flip). When a flip operation is specified in combination with a rotation operation, the ePXP will render the output such that the effect of the flip operation(s) occur BEFORE the rotation operation.

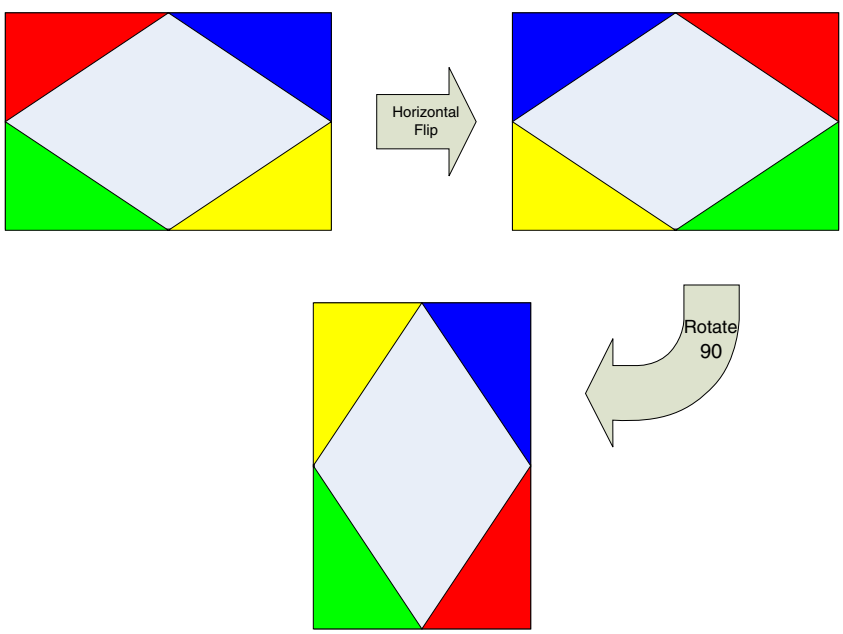


Figure 43-15. Flip with Rotation

Rotations and flip operations are enabled by setting the VFLIP, HFLIP, and ROTATE fields of the ePXP_CTRL register.

43.2.12 In-place Rendering

The ePXP also has the ability to process an image and write the resulting buffer back to the original S0 buffer. This is referred to as in place rendering. This scenario may be useful when software wishes to alpha blend multiple images where the overlays effectively overlap each other.

When the IN_PLACE control bit is set to 1, the control logic will optimize the ePXP's operations to only process the blocks that match an overlay region since all other pixels will be unmodified. This considerably reduces the processing time as well as the memory bandwidth used.

In place rendering is enabled by setting the IN_PLACE bit in the ePXP_CTRL registers. Note the following restrictions when rendering in place:

- The source buffer is used as the destination buffer (RGBBUF is not used)
- Only RGB S0 images are supported (not YUV)
- The output RGB format must be programmed to the same value as the input RGB format

43.2.13 Interlaced Video Support

The ePXP has some minimal ability to generate interlaced video content from a progressive source. There two available options, based on the bandwidth requirements and how software is managing video frames. The ePXP can either interlace on the input side (by reading every other line of input data) or on the output side (by writing the individual lines of video into two separate fields). Generally, output interleaving should be used since it is the most flexible mode (it allows scaling and full overlay support) and it only requires a single pass of the ePXP to generate two separate output fields. Input interleaving can be beneficial in cases where the ePXP is running at 60 fps, since it requires fewer fetches to produce the output data.

The ePXP will perform input interlacing when the INTERLACED_INPUT field is programmed to either FIELD0 or FIELD1 (to select the desired field). When performing output interlacing, the ePXP will write field0 data to the OUTBUF pointer and the field1 data to the OUTBUF2 pointer. The OUTPUT_INTERLACING field of the ePXP_CTRL register controls which of these fields (or both) are generated.

Output interlacing AND 2-plane output modes are not supported concurrently.

43.2.14 eLCDIF Interlock

The ePXP and eLCDIF support a mode where the internal SRAM can be used for the frame buffer to minimize external memory bandwidth required. This is accomplished by creating two buffers in SRAM that each correspond to n-lines of the frame buffer. The buffers must be consecutive and allocated as a single block of data.

When this mode is enabled, the ePXP will process one row of NxN pixel blocks and write the results to the first SRAM buffer (buffer 0). The ePXP will then alternate between writing subsequent rows to buffer 0 and buffer 1. After the ePXP generates the data for one buffer, the eLCDIF will begin reading that buffer and send the contents to the display device. Once the eLCDIF finishes reading a buffer, it will start displaying from the other buffer while the ePXP continues filling the now-empty buffer.

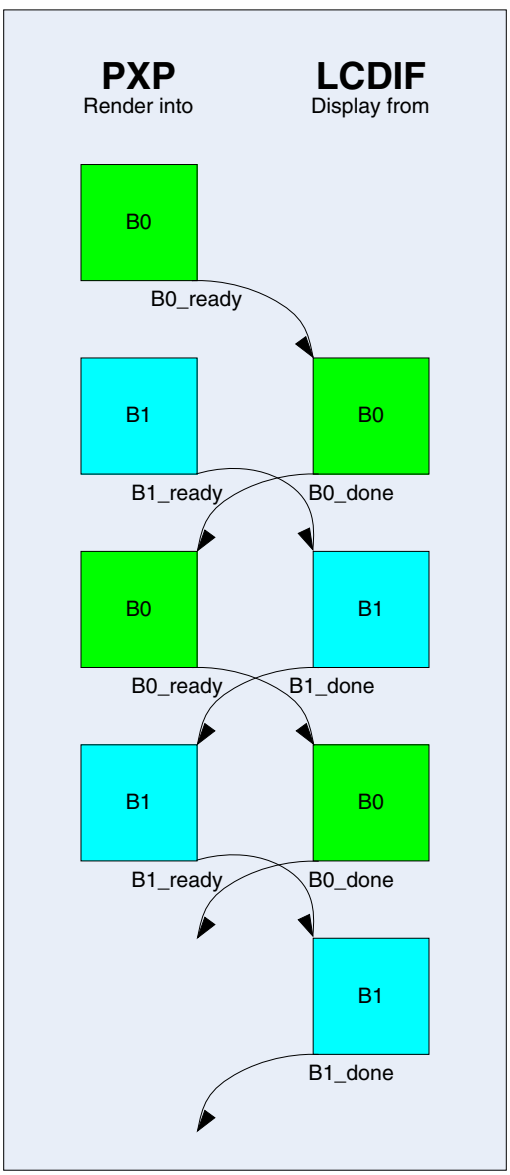


Figure 43-16. ePXP and eLCDIF Buffer Sharing

When the memory subsystem is not loaded, the ePXP should be able to render the buffers faster than the eLCDIF can drain the buffers. It is possible under some scenarios (high eLCDIF output rates with high memory latency) that the ePXP may not be able to keep up with the eLCDIF, even in the SRAM mode of operation. When this happens, the eLCDIF will signal that it has completed one of the buffers before the other has been rendered by the ePXP. This condition will be detected by the ePXP's control logic as an eLCDIF Abort, that causes the ePXP to abort processing in the current row and proceed to the following row. While an abort will create artifacts in the video display, it does minimize the artifacts by limiting them to the remaining pixels blocks in the current row versus ruining the entire frame buffer.

The memory requirements in SRAM to accommodate the partial frame buffer depends on the pixel resolution (16bpp, 24bpp, or 32bpp), the horizontal line size of the output image, and the block size (16 pixels or 8 pixels square).

Table 43-5. SRAM Memory Requirements

Image Size	Buffer Size (16bpp)	Buffer Size (24bpp)	Buffer Size (32bpp)
320x240 (QVGA) 0/180 rotation	10 Kbyte	15 Kbyte	2 0 Kbyte
320x240 (QVGA) 90/270 rotation	7.5 Kbyte	11.5 Kbyte	15 Kbyte
640x480 (VGA) 0/180 rotation	20 Kbyte	30 Kbyte	40 Kbyte
640x480 (VGA) 90/270 rotation	15 Kbyte	22.5 Kbyte	30 Kbyte

To enable the ePXP/eLCDIF Interlock mode, software must set the PXP_CTRL_ENABLE_LCD_HANDSHAKE bit in the ePXP and the LCDIF_ENABLE_PXP_HANDSHAKE bit in the eLCDIF. Using this mode requires the ePXP to begin processing the next frame immediately in order to have data ready for the eLCDIF at the beginning of the next frame. See Queueing Frame Operations to see how this can be accomplished.

43.2.15 Queueing Frame Operations

The ePXP supports a primitive ability to queue up one operation while the current operation is running. This is enabled through the use of the ePXP_NEXT register. When this register is written, it enables the ePXP to reload its current register contents with the data found at the location pointed to by this address (when it completes processing of the current frame (note that if virtual memory is used, this will be a virtual memory address).

This feature may be useful in helping to reduce the interrupt latency in servicing the ePXP, especially in cases where the ePXP and eLCDIF are using the on-chip SRAM buffer handshake (since the ePXP must begin generating next frame data immediately).

If the ePXP is idle when the ePXP_NEXT register is written, the ePXP treats this as an indication that it should immediately load the values at the pointer and begin processing the frame. This ability should allow software to use the same routines when programming the ePXP (so that the first frame does not differ from subsequent frames).

When loading values from the ePXP_NEXT register, nearly all registers in the ePXP are reloaded, including the interrupt enable bit in the control register. It is recommended that the interrupt enable value not be changed when using queued operations to ensure that interrupts are not spuriously lost or generated. The following table indicates the registers that are affected and the offset into the block address in memory.

Table 43-6. Registers and Offsets

Offset	Register	Offset	Register
0x00	CTRL	0x60	OL2
0x04	RGBBUF	0x64	OL2SIZE
0x08	RGBBUF2	0x68	OL2PARAM
0x0C	RGBSIZE	0x6C	OL2PARAM2
0x10	S0BUF	0x70	OL3
0x14	S0UBUF	0x74	OL3SIZE
0x18	S0VBUF	0x78	OL3PARAM
0x1C	S0PARAM	0x7C	OL3PARAM2
0x20	S0BACKGROUND	0x80	OL4
0x24	S0CROP	0x84	OL4SIZE
0x28	S0SCALE	0x88	OL4PARAM
0x2C	S0OFFSET	0x8C	OL4PARAM2
0x30	S0COLORKEYLOW	0x90	OL5
0x34	S0COLORKEYHIGH	0x94	OL5SIZE
0x38	OLCOLORKEYLOW	0x98	OL5PARAM
0x3C	OLCOLORKEYHIGH	0x9C	OL5PARAM2
0x40	OL0	0xA0	OL6
0x44	OL0SIZE	0xA4	OL6SIZE
0x48	OL0PARAM	0xA8	OL6PARAM
0x4C	OL0PARAM2	0xAC	OL6PARAM2
0x50	OL1	0xB0	OL7
0x54	OL1SIZE	0xB4	OL7SIZE

Table continues on the next page...

Table 43-6. Registers and Offsets (continued)

Offset	Register	Offset	Register
0x58	OL1PARAM	0xB8	OL7PARAM
0x5C	OL1PARAM2	0xBC	OL7PARAM2

43.3 Examples

This section includes several examples of programming the ePXP to render an output image. The image could be either a still image or one frame of a sequence of video images. For each case, the input and output images will be shown along with a table of ePXP register settings. In all examples, pointers to the data structures with image data will be referred to in the following notation: *imagename_type, where imagename indicates which image is being used and type indicates either luma (y), chroma (u, v) or RGB data (rgb). All register names are assumed to have the ePXP_ register prefix. The registers can be written in any order except the ePXP_CTRL register, which must be written last since it enables the ePXP's operation.

43.3.1 Basic QVGA Example

This example shows how to perform basic color space conversion of a 3-plane YUV image into an RGB image suitable for an LCD device.

Table 43-7. Register Use for Conversion

Register	Value	Description
RGBBUF	*example1_rgb	Pointer to the output buffer.
RGBSIZE	0xFF1400F0	ALPHA=0xFF WIDTH=0x140=320 HEIGHT=0x0F0=240
S0BUF	*morraine_y	Pointer to input Y buffer
S0UBUF	*morraine_u	Pointer to input U buffer
SOVBUF	*morraine_v	Pointer to input V buffer
S0PARAM	0x0000281E	WIDTH=0x28=40 (40*8=320 pixels) HEIGHT=0x1E=30 (30*8=240 pixels)
S0BACKGROUND	0x00000000	Black background region
S0CROP	0x00000000	No Cropping

Table continues on the next page...

Table 43-7. Register Use for Conversion (continued)

Register	Value	Description
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0PARAM	0x00000000	Overlay 0 disabled
OL1PARAM	0x00000000	Overlay 1 disabled
OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled
CTRL	0x00009003	S0_FORMAT=9 (YUV420) IRQ_ENABLE=1 ENABLE=1

The resulting image is simply the RGB equivalent of the YUV image:

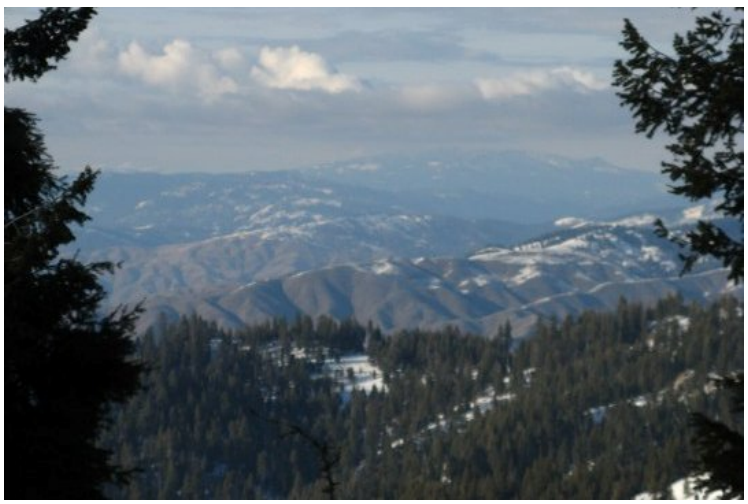


Figure 43-17. 3-Plane YUV to RGB Image

43.3.2 Basic QVGA with Overlays

This example is similar to the last, but adds two overlay images, one for a logo and the other as a time counter/control bar. The two overlay images are shown below. (Note that the black background is actually transparent in the real image).

Figure 43-18. Overlay Images

Table 43-8. Register Use for Conversion

Register	Value	Description
RGBBUF	*example1_rgb	Pointer to the output buffer.
RGBSIZE	0xFF1400F0	ALPHA=0xFF WIDTH=0x140=320 HEIGHT=0x0F0=240
S0BUF	*morraine_y	Pointer to input Y buffer
S0UBUF	*morraine_u	Pointer to input U buffer
SOVBUF	*morraine_v	Pointer to input V buffer
S0PARAM	0x0000281E	WIDTH=0x28=40 (40*8=320 pixels) HEIGHT=0x1E=30 (30*8= 240 pixels)
S0BACKGROUND	0x00000000	Black background region
S0CROP	0x00000000	No Cropping
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0	*overlay1_rgb	Pointer to control graphic
OL0SIZE	0x00000A02	WIDTH=0x0A=80 pixels HEIGHT=0x02=16pixels
OL0PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1

Table continues on the next page...

Table 43-8. Register Use for Conversion (continued)

Register	Value	Description
OL1	*logo_rgb	Pointer to logo graphic
OL1SIZE	0x0A181D06	XBASE=0x0A=80pixels YBASE=0x18=192pixels WIDTH=0x1D=232pixels HEIGHT=0x06=48pixels
OL1PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled
CTRL	0x00009003	S0_FORMAT=9 (YUV420) IRQ_ENABLE=1 ENABLE=1

The resulting image is shown below. Note the presence of the overlays in the upper left and lower right corners of the image.

Figure 43-19. Resulting Image

43.3.3 Cropped QVGA Example

This example displays the same image as the first example, but does so on a portrait-oriented display (240x320) without the overlays. Changes from the first example are shown in bold.

Table 43-9. Register Use for Conversion

Register	Value	Description
RGBBUF	*example1_rgb	Pointer to the output buffer.
RGBSIZE	0xFF0F0140	ALPHA=0xFF WIDTH=0x0F0=240 pixels HEIGHT=0x140=320 pixels
S0BUF	*morraine_y	Pointer to input Y buffer
S0UBUF	*morraine_u	Pointer to input U buffer
SOVBUF	*morraine_v	Pointer to input V buffer

Table continues on the next page...

Table 43-9. Register Use for Conversion (continued)

Register	Value	Description
S0PARAM	0x00 05 281E	YBASE=0x05=40pixels WIDTH=0x28=40 (40*8=320 pixels) HEIGHT=0x1E=30 (30*8= 240 pixels)
S0BACKGROUND	0x00000000	Black background region
S0CROP	0x05001E1E	XBASE=0x05=40 pixels YBASE=00=0pixels WIDTH=0x1E=240pixels HEIGHT=0x1E=240 pixels
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0PARAM	0x00000000	Overlay 0 disabled
OL1PARAM	0x00000000	Overlay 1 disabled
OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled
CTRL	0x00089003	CROP=1 S0_FORMAT=9 (YUV420) IRQ_ENABLE=1 ENABLE=1

In this case, we have now changed the RGB size to reflect the portrait nature of the display. The S0PARAM_YBASE has been changed to 0x05 (40 pixels) to place the S0 plane down 40 pixels from the top of the screen. The cropping register is now also used to control the cropping extents. The CROP_XBASE is set to 0x05 (40 pixels) to move the origin of the S0 buffer to the (40,0) location within the buffer. The CROP_WIDTH/ CROP_HEIGHT are also programmed to ensure that the resulting image in the output buffer is cropped to 240x240 pixels. Since the image no longer covers the entire output buffer, the S0BACKGROUND register is used to letterbox the image in black. The resulting image is shown below.

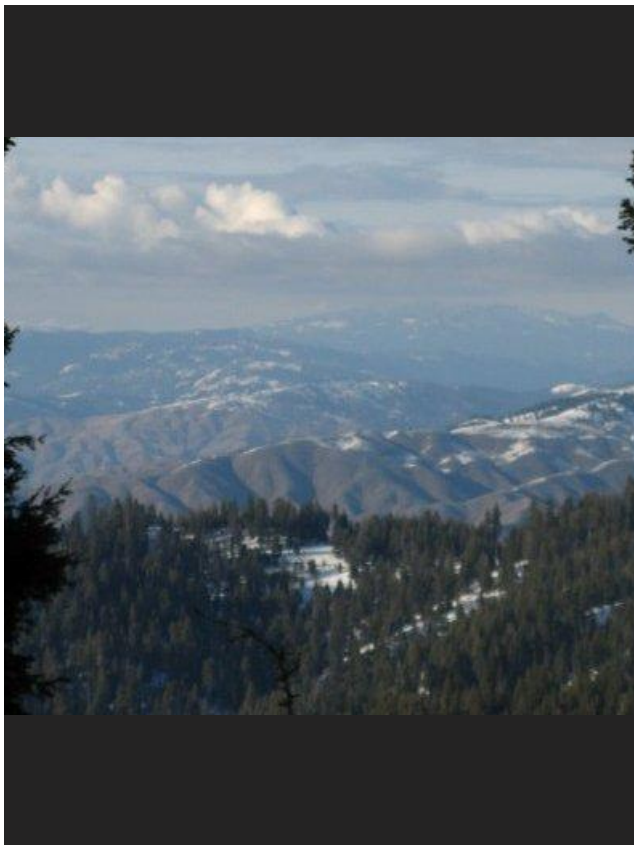


Figure 43-20. Cropped QVGA Example

43.3.4 Upscale QVGA to VGA with Overlays

In this example, the image will be upscaled from QVGA to VGA resolution and displayed with the two overlays from the second example. Changes from the second example are shown in bold.

Table 43-10. Register Use for Conversion

Register	Value	Description
RGBBUF	*example1_rgb	Pointer to the output buffer.
RGBSIZE	0xFF2801E0	ALPHA=0xFF WIDTH=0x280=640 HEIGHT=0x1E0=480
S0BUF	*morraine_y	Pointer to input Y buffer
S0UBUF	*morraine_u	Pointer to input U buffer
SOVBUF	*morraine_v	Pointer to input V buffer

Table continues on the next page...

Table 43-10. Register Use for Conversion (continued)

Register	Value	Description
S0PARAM	0x0000281E	WIDTH=0x28=40 (40*8=320 pixels) HEIGHT=0x1E=30 (30*8= 240 pixels)
S0BACKGROUND	0x00000000	Black background region
S0CROP	0x0000503C	WIDTH=0x50=640pixels HEIGHT=0x3C=320pixels
S0SCALE	0x08000800	XSCALE=0x0800=2x scale YSCALE=0x0800=2x scale
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0	*overlay1_rgb	Pointer to control graphic
OL0SIZE	0x 23 000A02	XBASE=0x23=280pixels WIDTH=0x0A=80 pixels HEIGHT=0x02=16pixels
OL0PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL1	*logo_rgb	Pointer to logo graphic
OL1SIZE	0x 1936 1D06	XBASE=0x19=200pixels YBASE=0x36=432pixels WIDTH=0x1D=232pixels HEIGHT=0x06=48pixels
OL1PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled

Table continues on the next page...

Table 43-10. Register Use for Conversion (continued)

Register	Value	Description
CTRL	0x000c9003	SCALE=1 CROP=1 S0_FORMAT=9 (YUV420) IRQ_ENABLE=1 ENABLE=1

The resulting image is shown in the figure below. The overlays have moved in this image and that the overall image size is now larger than before.

Figure 43-21. Upscale QVGA to VGA with Overlays

43.3.5 Downscale VGA to WQVGA (480x272) to fill screen

In this example, a VGA image will be downscaled to fix the extents of a 480x272 WQVGA display. This means that the aspect ratio of the resulting image will not match that of the source image, therefore the scaling factors in the horizontal and vertical directions will differ from each other.

Table 43-11. Register Use for Conversion

Register	Value	Description
RGBBUF	*example_rgb	Pointer to the output buffer.
RGBSIZE	0xFFf1E0110	ALPHA=0xFF WIDTH=0x1E0=480 HEIGHT=0x110=272
S0BUF	*garden_y	Pointer to input Y buffer
S0UBUF	*garden_u	Pointer to input U buffer
SOVBUF	*garden_v	Pointer to input V buffer
S0PARAM	0x0000503C	WIDTH=0x50=80=640 pixels HEIGHT=0x3C=60=480 pixels
S0BACKGROUND	0x00000000	Black background region
S0CROP	0x00003C22	WIDTH=0x3C=480 pixels HEIGHT=0x22=272 pixels
S0SCALE	0x1C3C1555	YSCALE=0x1C3C=1/1.765x XSCALE=0x1555=1/1.333x
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0PARAM	0x00000000	Overlay 0 disabled
OL1PARAM	0x00000000	Overlay 1 disabled
OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled
CTRL	0x000C9003	SCALE=1 CROP=1 S0_FORMAT=9 (YUV420) IRQ_ENABLE=1 ENABLE=1

Note that the scaling factors are computed as (source/dest)*4096, thus in the horizontal direction 640/480 x 4096 = 5461 = 0 x 1555. In the vertical direction, the scaling factor is computed as 480/272 x 4096=7228 = 0x1C3C. The original source image and resulting scaled images are shown below:

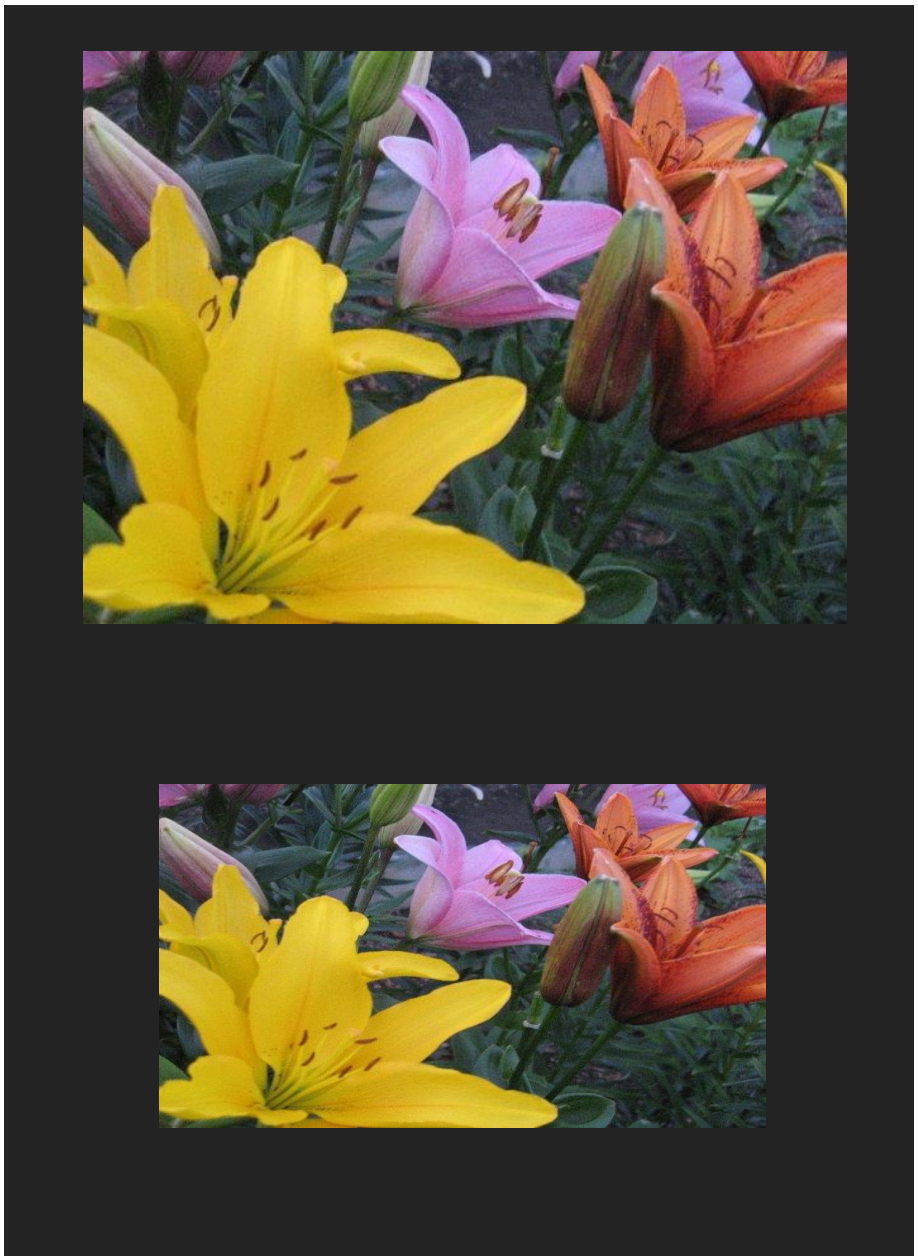


Figure 43-22. Downscale VGA to WQVGA (480x272)

43.3.6 Downscale VGA to QVGA with Overlapping Overlays

The final example will perform a 1/2x scaling of a VGA image to QVGA to maintain the aspect ratio. It will also add four overlays to present the image as if it were a photo album application.

Table 43-12. Register Use for Conversion

Register	Value	Description
RGBBUF	*example_rgb	Pointer to the output buffer.
RGBSIZE	0xFFf1E0110	ALPHA=0xFF WIDTH=0x1E0=480 HEIGHT=0x110=272
S0BUF	*garden_y	Pointer to input Y buffer
S0UBUF	garden_u	Pointer to input U buffer
SOVBUF	garden_v	Pointer to input V buffer
S0PARAM	0x0000503C	WIDTH=0x50=80=640 pixels HEIGHT=0x3C=60=480 pixels
S0BACKGROUND	0x00000040	Dark Blue background region
S0CROP	0x0000281E	WIDTH=0x28=320 pixels HEIGHT=0x1E=240 pixels
S0SCALE	0x20002000	YSCALE=0x2000=1/2x XSCALE=0x1555=1/2x
S0OFFSET	0x08000800	XOFFSET=0x0800 (1/2 pixel) YOFFSET=0x0800 (1/2 pixel)
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0	*prev_rgb	Pointer to previous graphic
OL0SIZE	0x0B1B0402	XBASE=0x0B=88pixels YBASE=0x1B=216pixels WIDTH=0x04=32 pixels HEIGHT=0x02=16pixels
OL0PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL1	*next_rgb	Pointer to next graphic
OL1SIZE	0x2D1B0402	XBASE=0x2D=360pixels YBASE=0x1B=216pixels WIDTH=0x04=32 pixels HEIGHT=0x02=16pixels

Table continues on the next page...

Table 43-12. Register Use for Conversion (continued)

Register	Value	Description
OL1PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL2	*text_overlay	Pointer to text graphic
OL2SIZE	0x00000A1E	XBASE=0x00=0pixels YBASE=0x00=0pixels WIDTH=0x0A=80pixels HEIGHT=0x1E=240pixels
OL2PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL3	*border_rgb	Pointer to rectangular border graphic
OL3SIZE	0x0A00281E	XBASE=0x0A=80pixels YBASE=0x00=0pixels WIDTH=0x28=320pixels HEIGHT=0x1E=240pixels
OL3PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888) ALPHA_CTRL=0 (embedded alpha) ENABLE=1
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled
CTRL	0x000C9003	SCALE=1 CROP=1 S0_FORMAT=9 (YUV420) IRQ_ENABLE=1 ENABLE=1

The resulting image is shown below. The text is rendered in a transparent overlay (overlay #2) on the right side of the screen. The background color (#000040) is dark blue and shows through the overlay as the background color. Overlay #3 applies a thin white alpha-blended border around the image to frame it. Overlays #0 and #1 generate the

Next> and <Prev images alpha blended onto the image. Because these overlays are higher priority (lower numbered) than the border, they are used at these locations instead of overlay #3.

Figure 43-23. Downscale VGA to QVGA with Overlapping Overlays

43.4 Programmable Registers

ePXP Hardware Register Format Summary

ePXP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_C000	ePXP Control Register 0 (ePXP_CTRL)	32	R/W	C000_0000h	43.4.1/ 2444
4100_C004	ePXP Control Register 0 (ePXP_CTRL_SET)	32	R/W	C000_0000h	43.4.1/ 2444
4100_C008	ePXP Control Register 0 (ePXP_CTRL_CLR)	32	R/W	C000_0000h	43.4.1/ 2444
4100_C00C	ePXP Control Register 0 (ePXP_CTRL_TOG)	32	R/W	C000_0000h	43.4.1/ 2444

Table continues on the next page...

ePXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_C010	ePXP Status Register (ePXP_STAT)	32	R/W	0000_0000h	43.4.2/ 2448
4100_C014	ePXP Status Register (ePXP_STAT_SET)	32	R/W	0000_0000h	43.4.2/ 2448
4100_C018	ePXP Status Register (ePXP_STAT_CLR)	32	R/W	0000_0000h	43.4.2/ 2448
4100_C01C	ePXP Status Register (ePXP_STAT_TOG)	32	R/W	0000_0000h	43.4.2/ 2448
4100_C020	Output Frame Buffer Pointer (ePXP_OUTBUF)	32	R/W	0000_0000h	43.4.3/ 2449
4100_C030	Output Frame Buffer Pointer #2 (ePXP_OUTBUF2)	32	R/W	0000_0000h	43.4.4/ 2449
4100_C040	ePXP Output Buffer Size (ePXP_OUTSIZE)	32	R/W	0000_0000h	43.4.5/ 2450
4100_C050	ePXP Source 0 (video) Input Buffer Pointer (ePXP_S0BUF)	32	R/W	0000_0000h	43.4.6/ 2451
4100_C060	Source 0 U/Cb or 2 Plane UV Input Buffer Pointer (ePXP_S0UBUF)	32	R/W	0000_0000h	43.4.7/ 2451
4100_C070	Source 0 V/Cr Input Buffer Pointer (ePXP_S0VBUF)	32	R/W	0000_0000h	43.4.8/ 2452
4100_C080	ePXP Source 0 (video) Buffer Parameters (ePXP_S0PARAM)	32	R/W	0000_0000h	43.4.9/ 2452
4100_C090	Source 0 Background Color (ePXP_S0BACKGROUND)	32	R/W	0000_0000h	43.4.10/ 2453
4100_C0A0	Source 0 Cropping Register (ePXP_S0CROP)	32	R/W	0000_0000h	43.4.11/ 2454
4100_C0B0	Source 0 Scale Factor Register (ePXP_S0SCALE)	32	R/W	1000_1000h	43.4.12/ 2455
4100_C0C0	Source 0 Scale Offset Register (ePXP_S0OFFSET)	32	R/W	0000_0000h	43.4.13/ 2456
4100_C0D0	Color Space Conversion Coefficient Register 0 (ePXP_CSCCOEF0)	32	R/W	0400_0000h	43.4.14/ 2457
4100_C0E0	Color Space Conversion Coefficient Register 1 (ePXP_CSCCOEF1)	32	R/W	0123_0208h	43.4.15/ 2458
4100_C0F0	Color Space Conversion Coefficient Register 2 (ePXP_CSCCOEF2)	32	R/W	079B_076Ch	43.4.16/ 2459
4100_C100	ePXP Next Frame Pointer (ePXP_NEXT)	32	R/W	0000_0000h	43.4.17/ 2460
4100_C104	ePXP Next Frame Pointer (ePXP_NEXT_SET)	32	R/W	0000_0000h	43.4.17/ 2460

Table continues on the next page...

ePXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_C108	ePXP Next Frame Pointer (ePXP_NEXT_CLR)	32	R/W	0000_0000h	43.4.17/ 2460
4100_C10C	ePXP Next Frame Pointer (ePXP_NEXT_TOG)	32	R/W	0000_0000h	43.4.17/ 2460
4100_C180	ePXP S0 Color Key Low (ePXP_S0COLORKEYLOW)	32	R/W	00FF_FFFFh	43.4.18/ 2462
4100_C190	ePXP S0 Color Key High (ePXP_S0COLORKEYHIGH)	32	R/W	0000_0000h	43.4.19/ 2462
4100_C1A0	ePXP Overlay Color Key Low (ePXP_OLCOLORKEYLOW)	32	R/W	00FF_FFFFh	43.4.20/ 2463
4100_C1B0	ePXP Overlay Color Key High (ePXP_OLCOLORKEYHIGH)	32	R/W	0000_0000h	43.4.21/ 2464
4100_C1E0	ePXP Debug Register (ePXP_DEBUG)	32	R	0000_0000h	43.4.22/ 2464
4100_C200	ePXP Overlay n Buffer Pointer (ePXP_OL0)	32	R/W	0000_0000h	43.4.23/ 2465
4100_C210	ePXP Overlay n Size (ePXP_OL__SIZE0)	32	R/W	0000_0000h	43.4.24/ 2465
4100_C220	ePXP Overlay n Parameters (ePXP_OL__PARAM0)	32	R/W	0000_0000h	43.4.25/ 2466
4100_C240	ePXP Overlay n Buffer Pointer (ePXP_OL1)	32	R/W	0000_0000h	43.4.23/ 2465
4100_C250	ePXP Overlay n Size (ePXP_OL__SIZE1)	32	R/W	0000_0000h	43.4.24/ 2465
4100_C260	ePXP Overlay n Parameters (ePXP_OL__PARAM1)	32	R/W	0000_0000h	43.4.25/ 2466
4100_C280	ePXP Overlay n Buffer Pointer (ePXP_OL2)	32	R/W	0000_0000h	43.4.23/ 2465
4100_C290	ePXP Overlay n Size (ePXP_OL__SIZE2)	32	R/W	0000_0000h	43.4.24/ 2465
4100_C2A0	ePXP Overlay n Parameters (ePXP_OL__PARAM2)	32	R/W	0000_0000h	43.4.25/ 2466
4100_C2C0	ePXP Overlay n Buffer Pointer (ePXP_OL3)	32	R/W	0000_0000h	43.4.23/ 2465
4100_C2D0	ePXP Overlay n Size (ePXP_OL__SIZE3)	32	R/W	0000_0000h	43.4.24/ 2465
4100_C2E0	ePXP Overlay n Parameters (ePXP_OL__PARAM3)	32	R/W	0000_0000h	43.4.25/ 2466
4100_C300	ePXP Overlay n Buffer Pointer (ePXP_OL4)	32	R/W	0000_0000h	43.4.23/ 2465

Table continues on the next page...

ePXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_C310	ePXP Overlay n Size (ePXP_OL__SIZE4)	32	R/W	0000_0000h	43.4.24/ 2465
4100_C320	ePXP Overlay n Parameters (ePXP_OL__PARAM4)	32	R/W	0000_0000h	43.4.25/ 2466
4100_C340	ePXP Overlay n Buffer Pointer (ePXP_OL5)	32	R/W	0000_0000h	43.4.23/ 2465
4100_C350	ePXP Overlay n Size (ePXP_OL__SIZE5)	32	R/W	0000_0000h	43.4.24/ 2465
4100_C360	ePXP Overlay n Parameters (ePXP_OL__PARAM5)	32	R/W	0000_0000h	43.4.25/ 2466
4100_C380	ePXP Overlay n Buffer Pointer (ePXP_OL6)	32	R/W	0000_0000h	43.4.23/ 2465
4100_C390	ePXP Overlay n Size (ePXP_OL__SIZE6)	32	R/W	0000_0000h	43.4.24/ 2465
4100_C3A0	ePXP Overlay n Parameters (ePXP_OL__PARAM6)	32	R/W	0000_0000h	43.4.25/ 2466
4100_C3C0	ePXP Overlay n Buffer Pointer (ePXP_OL7)	32	R/W	0000_0000h	43.4.23/ 2465
4100_C3D0	ePXP Overlay n Size (ePXP_OL__SIZE7)	32	R/W	0000_0000h	43.4.24/ 2465
4100_C3E0	ePXP Overlay n Parameters (ePXP_OL__PARAM7)	32	R/W	0000_0000h	43.4.25/ 2466
4100_C400	Color Space Conversion Control Register. (ePXP_CSC2CTRL)	32	R/W	0000_0001h	43.4.26/ 2468
4100_C410	Color Space Conversion Coefficient Register 0 (ePXP_CSC2COEF0)	32	R/W	0000_0000h	43.4.27/ 2469
4100_C420	Color Space Conversion Coefficient Register 1 (ePXP_CSC2COEF1)	32	R/W	0000_0000h	43.4.28/ 2470
4100_C430	Color Space Conversion Coefficient Register 2 (ePXP_CSC2COEF2)	32	R/W	0000_0000h	43.4.29/ 2470
4100_C440	Color Space Conversion Coefficient Register 3 (ePXP_CSC2COEF3)	32	R/W	0000_0000h	43.4.30/ 2471
4100_C450	Color Space Conversion Coefficient Register 4 (ePXP_CSC2COEF4)	32	R/W	0000_0000h	43.4.31/ 2471
4100_C460	Color Space Conversion Coefficient Register 5 (ePXP_CSC2COEF5)	32	R/W	0000_0000h	43.4.32/ 2472
4100_C470	Lookup Table Control Register. (ePXP_LUT_CTRL)	32	R/W	8000_0000h	43.4.33/ 2473
4100_C480	Lookup Table Data Register. (ePXP_LUT)	32	R/W	0000_0000h	43.4.34/ 2473

Table continues on the next page...

ePXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4100_C490	Histogram Control Register. (ePXP_HIST_CTRL)	32	R/W	0000_0020h	43.4.35/ 2474
4100_C4A0	2-level Histogram Parameter Register. (ePXP_HIST2_PARAM)	32	R/W	0000_0F00h	43.4.36/ 2475
4100_C4B0	4-level Histogram Parameter Register. (ePXP_HIST4_PARAM)	32	R/W	0F0A_0500h	43.4.37/ 2476
4100_C4C0	8-level Histogram Parameter 0 Register. (ePXP_HIST8_PARAM0)	32	R/W	0604_4000h	43.4.38/ 2477
4100_C4D0	8-level Histogram Parameter 1 Register. (ePXP_HIST8_PARAM1)	32	R/W	0F0D_0B09h	43.4.39/ 2478
4100_C4E0	16-level Histogram Parameter 0 Register. (ePXP_HIST16_PARAM0)	32	R/W	0302_0100h	43.4.40/ 2479
4100_C4F0	16-level Histogram Parameter Register. (ePXP_HIST16_PARAM1)	32	R/W	0706_0504h	43.4.41/ 2480
4100_C500	16-level Histogram Parameter Register. (ePXP_HIST16_PARAM2)	32	R/W	0B0A_0908h	43.4.42/ 2481
4100_C510	16-level Histogram Parameter Register. (ePXP_HIST16_PARAM3)	32	R/W	0F0E_ 0D0Ch	43.4.43/ 2482

43.4.1 ePXP Control Register 0 (ePXP_CTRL*n*)

The Control register contains the primary controls for the ePXP block. The present bits indicate which of the sub-features of the block are present in the hardware.

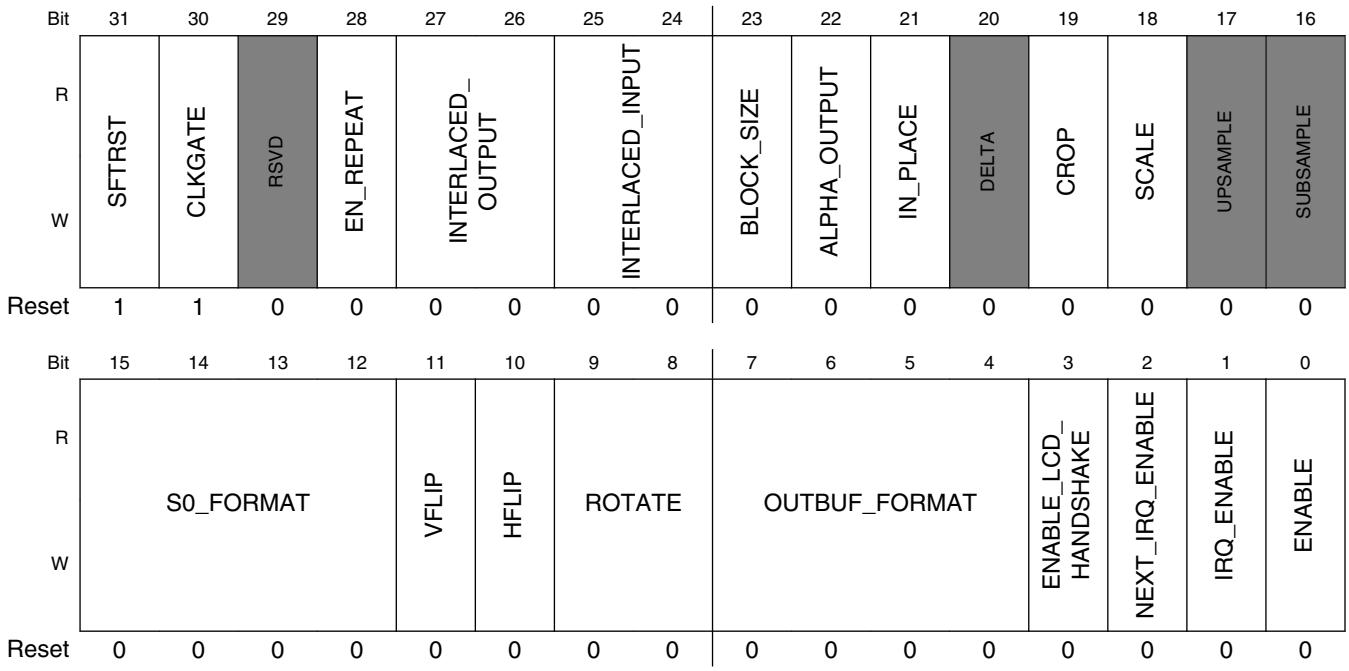
```
ePXP_CTRL_SET(BM_PXP_CTRL_SFTRST);
    ePXP_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Addresses: ePXP_CTRL is 4100_C000h base + 0h offset = 4100_C000h

ePXP_CTRL_SET is 4100_C000h base + 4h offset = 4100_C004h

ePXP_CTRL_CLR is 4100_C000h base + 8h offset = 4100_C008h

ePXP_CTRL_TOG is 4100_C000h base + Ch offset = 4100_C00Ch



ePXP_CTRL*n* field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal ePXP operation. Set this bit to one (default) to disable clocking with the ePXP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the ePXP block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 RSVD	Reserved, always set to zero.
28 EN_REPEAT	Enables the ePXP to run continuously. When this bit is set, the ePXP will repeat based on the current configuration register settings. If this bit is not set, the ePXP will complete the process and enter the idle state ready to accept the next frame to be processed. This bit should be set when the LCDIF handshake mode is enabled so that the next frame is automatically generated for the next screen refresh cycle. If it not set and the handshake mode is enabled, the CPU will have to initiate the ePXP for the next refresh cycle. When the ePXP NEXT feature is used, it has priority over the REPEAT mode, in that the new register settings are fetched first, and then the next ePXP operation will continue.
27–26 INTERLACED_OUTPUT	Determines how the ePXP writes its output data. Output interlacing should not be used in conjunction with input interlacing. Splitting frames into fields is most efficient using output interlacing. two-plane output formats AND interlaced output is not supported. 0x0 **PROGRESSIVE** — All data written in progressive format to the OUTBUF Pointer. 0x1 **FIELD0** — Interlaced output: only data for field 0 is written to the OUTBUF Pointer. 0x2 **FIELD1** — Interlaced output: only data for field 1 is written to the OUTBUF2 Pointer. 0x3 **INTERLACED** — Interlaced output: data for field 0 is written to OUTBUF and data for field 1 is written to OUTBUF2.

Table continues on the next page...

ePXP_CTRL*n* field descriptions (continued)

Field	Description
25–24 INTERLACED_ INPUT	When set, causes the fetch side of the ePXP to fetch every other line from the source buffers. This effectively produces one field of interlaced output data. Scaling should NOT be enabled for interlaced operation and only overlays with boundaries on 8x16 multiples are supported. 0x0 **PROGRESSIVE** — All data will be read and processed in progressive format. 0x2 **FIELD0** — Interlaced, Field 0: only data for field 0 (even lines) is read/processed. 0x3 **FIELD1** — Interlaced, Field 1: only data for field 1 (odd lines) is read/processed.
23 BLOCK_SIZE	Select the block size to process. 0x0 **8X8** — Process 8x8 pixel blocks. 0x1 **16X16** — Process 16x16 pixel blocks.
22 ALPHA_ OUTPUT	Indicates that alpha component in output buffer pixels should be overridden by PXP_OUTSIZE[ALPHA] register. If 0, retain their alpha value from the computed alpha for that pixel.
21 IN_PLACE	When set, this enables the ePXP to perform an alpha blend operation on an existing buffer (output buffer is set to S0 buffer). In this case, the ePXP will perform the alpha blending of the overlays into the source buffer. Since only pixels containing an overlay are processed, the ePXP does this very efficiently.
20 DELTA	Reserved for future use.
19 CROP	Indicates that the S0 plane should use the cropping register to provide the extents for the output S0 buffer cropping. If not set, the input video cropping extents will be inferred from the S0 WIDTH and HEIGHT fields. When scaling, the CROP bit and controls should be used to specify the scaled image size in the output buffer.
18 SCALE	This bit indicates that the output image should be scaled (only YUV/YCbCr images may be scaled -- RGB scaling is not supported). The XSCALE and YSCALE registers should be programmed accordingly. In addition, the CROP bit and the S0CROP registers should be programmed to ensure that the scaled image is properly cropped in the output buffer. When this bit is zero, the contents of the scaling registers are ignored.
17 UPSAMPLE	Reserved for future use.
16 SUBSAMPLE	Reserved for future use.
15–12 S0_FORMAT	Source 0 buffer format. To select between YUV and YCbCr formats, see bit 31 of the CSCCOEF0 register. 0x1 **RGB888** — 32-bit pixels (unpacked 24-bit format) 0x4 **RGB565** — 16-bit pixels 0x5 **RGB555** — 16-bit pixels 0x8 **YUV422** — 16-bit pixels (3-plane format) 0x9 **YUV420** — 16-bit pixels (3-plane format) 0xA **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0xB **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0xC **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0xD **YUV2P420** — 16-bit pixels 0xE **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0xF **YVU2P420** — 16-bit pixels

Table continues on the next page...

ePXP_CTRL*n* field descriptions (continued)

Field	Description
11 VFLIP	Indicates that the output buffer should be flipped vertically (effect applied before rotation).
10 HFLIP	Indicates that the output buffer should be flipped horizontally (effect applied before rotation).
9–8 ROTATE	Indicates the clockwise rotation to be applied at the output buffer. The rotation effect is defined as occurring after the FLIP_X and FLIP_Y permutation. 0x0 **ROT_0** — 0x1 **ROT_90** — 0x2 **ROT_180** — 0x3 **ROT_270** —
7–4 OUTBUF_ FORMAT	Output framebuffer format. The UV byte lanes are synonymous with CbCr byte lanes for YUV output pixel formats. For example, the YUV2P420 format should be selected when the output is YCbCr 2-plane 420 output format. 0x0 **ARGB8888** — 32-bit pixels 0x1 **RGB888** — 32-bit pixels (unpacked 24-bit pixel in 32 bit DWORD.) 0x2 **RGB888P** — 24-bit pixels (packed 24-bit format) 0x3 **ARGB1555** — 16-bit pixels 0x4 **RGB565** — 16-bit pixels 0x5 **RGB555** — 16-bit pixels 0x7 **YUV444** — 32-bit pixels (1-plane XYUV unpacked) 0x8 **MONOC8** — 8-bit monochrome pixels (1-plane Y luma output) 0x9 **MONOC4** — 4-bit monochrome pixels (1-plane Y luma, 4 bit truncation) 0xA **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0xB **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0xC **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0xD **YUV2P420** — 16-bit pixels (2-plane UV) 0xE **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0xF **YVU2P420** — 16-bit pixels (2-plane VU)
3 ENABLE_LCD_ HANDSHAKE	Enable handshake with LCD controller. When this is set, the ePXP will not process an entire framebuffer, but will instead process rows of NxN blocks in a double-buffer handshake with the LCDIF. This enables the use of the onboard SRAM for a partial frame buffer.
2 NEXT_IRQ_ ENABLE	Next command interrupt enable. When set, the ePXP will issue an interrupt when a queued command initiated by a write to the PXP_NEXT register has been loaded into the ePXP's registers. This interrupt also indicates that a new command may now be queued.
1 IRQ_ENABLE	Interrupt enable. When set, the ePXP will issue an interrupt when PXP processing is complete. Note: When using the PXP_NEXT functionality to reprogram the ePXP, the new value of this bit will be used and may therefore enable or disable an interrupt unintentionally.
0 ENABLE	Enables ePXP operation with specified parameters. The ENABLE bit will remain set while the ePXP is active and will be cleared once the current operation completes. Software should use the IRQ bit in the PXP_STAT when polling for ePXP completion.

43.4.2 ePXP Status Register (ePXP_STAT*n*)

The ePXP Interrupt Status register provides ePXP interrupt status and the current X/Y block coordinate that is being processed.

EXAMPLE

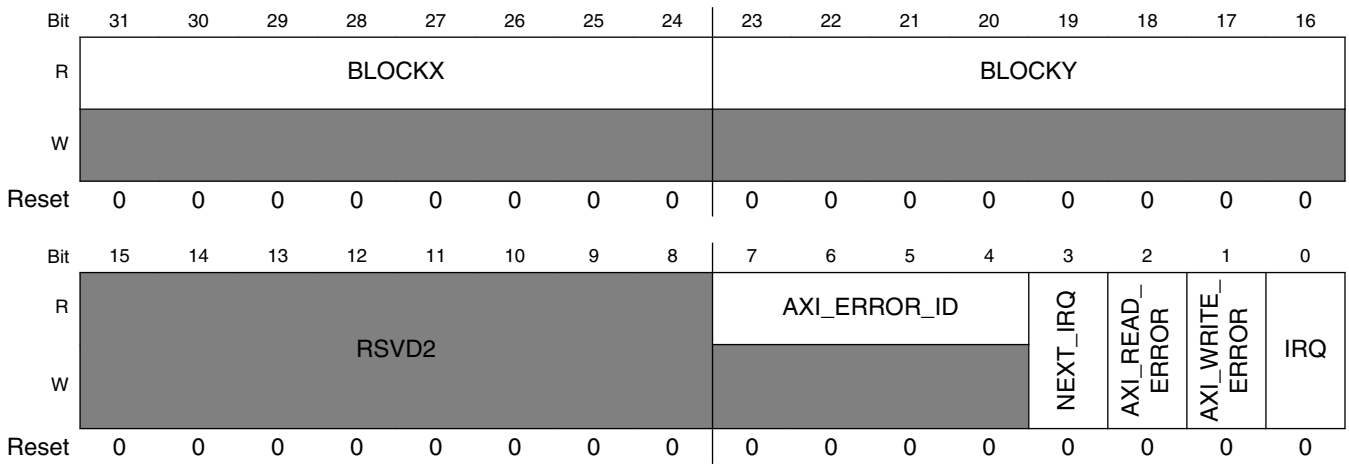
```
ePXP_STAT_CLR(BM_PXP_STAT_IRQ);  // clear CSC interrupt
```

Addresses: ePXP_STAT is 4100_C000h base + 10h offset = 4100_C010h

ePXP_STAT_SET is 4100_C000h base + 14h offset = 4100_C014h

ePXP_STAT_CLR is 4100_C000h base + 18h offset = 4100_C018h

ePXP_STAT_TOG is 4100_C000h base + 1Ch offset = 4100_C01Ch



ePXP_STAT*n* field descriptions

Field	Description
31–24 BLOCKX	Indicates the X coordinate of the block currently being rendered.
23–16 BLOCKY	Indicates the X coordinate of the block currently being rendered.
15–8 RSVD2	Reserved, always set to zero.
7–4 AXI_ERROR_ID	Indicates the AXI ID of the failing bus operation.
3 NEXT_IRQ	Indicates that a command issued with the "Next Command" functionality has been issued and that a new command may be initiated with a write to the PXP_NEXT register.
2 AXI_READ_ ERROR	Indicates PXP encountered an AXI read error and processing has been terminated.
1 AXI_WRITE_ ERROR	Indicates ePXP encountered an AXI write error and processing has been terminated.

Table continues on the next page...

ePXP_STAT*n* field descriptions (continued)

Field	Description
0 IRQ	Indicates current ePXP interrupt status.

43.4.3 Output Frame Buffer Pointer (ePXP_OUTBUF)

Output Framebuffer Pointer. This register points to the beginning of the output frame buffer. This pointer is used for progressive format and field 0 when generating interlaced output.

This register is used by the logic to point to the current output location for the output frame buffer.

EXAMPLE

```
PXP_OUTBUF_WR( buffer );
```

Address: ePXP_OUTBUF is 4100_C000h base + 20h offset = 4100_C020h

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

ePXP_OUTBUF field descriptions

Field	Description
31–0 ADDR	Current address pointer for the output frame buffer. The address must be word-aligned for proper ePXP operation.

43.4.4 Output Frame Buffer Pointer #2 (ePXP_OUTBUF2)

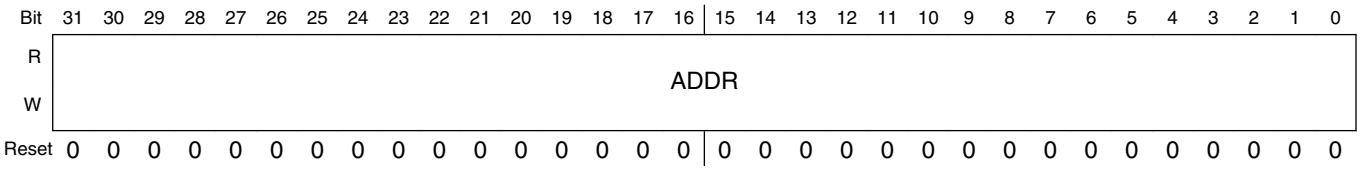
Output Framebuffer Pointer #2. This register points to the beginning of the output frame buffer for eitherfield 1 when generating interlaced output or for the UV buffer when in YUV 2-plane output modes. Both interlaced output AND 2-plane output modes are not supported. This register is NOT used as the pointer to the 2nd buffer when in LCDIF_HANDSHAKE mode.

This register is used by the logic to point to the current output location for the field 1 or UV output frame buffer.

EXAMPLE

```
PXP_OUTBUF_WR( field0 );  // buffer for interlaced field 0
PXP_OUTBUF2_WR( field1 ); // buffer for interlaced field 1
```

Address: ePXP_OUTBUF2 is 4100_C000h base + 30h offset = 4100_C030h



ePXP_OUTBUF2 field descriptions

Field	Description
31–0 ADDR	Current address pointer for the output frame buffer. The address must be word-aligned for proper ePXP operation.

43.4.5 ePXP Output Buffer Size (ePXP_OUTSIZE)

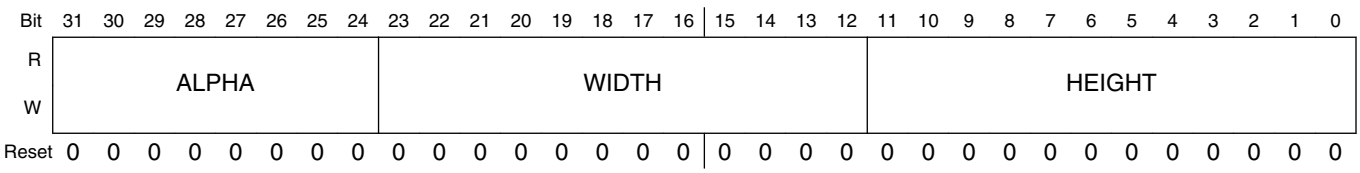
This register contains framebuffer size information for the output buffer (independent of the rotation). When rotating the framebuffer, the ePXP will automatically modify the output WIDTH/HEIGHT to accomodate the rotated size.

This register sets the size of the output frame buffer in pixels, not blocks. The frame buffer need not be a multiple of NxN pixels. Partial blocks will be written for output frame buffer sizes that are not divisable by N pixels in either dimension.

EXAMPLE

```
PXP_OUTSIZE.U.WIDTH=320;   // set width
PXP_OUTSIZE.U.HEIGHT=240;  // set height

PX_OUTSIZE_WR( BF_PXP_OUTSIZE_WIDTH(320) |  BF_PXP_OUTSIZE_HEIGHT(240) );
```

Address: ePXP_OUTSIZE is 4100_C000h base + 40h offset = 4100_C040h



ePXP_OUTSIZE field descriptions

Field	Description
31–24 ALPHA	When generating an output buffer with an alpha component, the value in this field will be used.
23–12 WIDTH	Indicates number of horizontal PIXELS in the image (non-rotated). The image size is not required to be a multiple of 8 pixels. The ePXP will handle clipping the pixel output at this boundary.
11–0 HEIGHT	Indicates the number of vertical PIXELS in the image (non-rotated). The image size is not required to be a multiple of 8 pixels. The ePXP will handle clipping the pixel output at this boundary.

43.4.6 ePXP Source 0 (video) Input Buffer Pointer (ePXP_S0BUF)

S0 Input Buffer Pointer. This should be programmed to the starting address of the RGB data or Y (luma) data for the S0 plane.

This register contains the pointer to the Luma/RGB buffer.

EXAMPLE

```
PXP_S0BUF_WR(image_rgb); // RGB image

PXP_S0BUF_WR(image_y);   // Y (luma) image data
PXP_S0UBUF_WR(image_u);  // U (Cb) image data
PXP_S0VBUF_WR(image_v);  // V (Cr) image data
```

Address: ePXP_S0BUF is 4100_C000h base + 50h offset = 4100_C050h

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

ePXP_S0BUF field descriptions

Field	Description
31–0 ADDR	Address pointer for the S0 RGB or Y (luma) input buffer. The address must be word-aligned for proper ePXP operation.

43.4.7 Source 0 U/Cb or 2 Plane UV Input Buffer Pointer (ePXP_S0UBUF)

S0 Chroma (U/Cb/UV) Input Buffer Pointer. This register points to the beginning of the Source 0 U/Cb input buffer. In two plane operation, this register points to the beginning of the Source 0 UV chroma input buffer.

This register contains the pointer to the Chroma U/Cb or 2 plane UV buffer when performing colorspace conversion. This register is unused when processing RGB data.

EXAMPLE

```
PXP_S0BUF_WR(image_y);   // Y (luma) image data
PXP_S0UBUF_WR(image_u);  // U (Cb) image data
PXP_S0VBUF_WR(image_v);  // V (Cr) image data
```

Address: ePXP_S0UBUF is 4100_C000h base + 60h offset = 4100_C060h

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

ePXP_S0UBUF field descriptions

Field	Description
31–0 ADDR	Address pointer for the S0 (video) U/Cb or two-plane UV Chroma input buffer. The address must be word-aligned for proper ePXP operation.

43.4.8 Source 0 V/Cr Input Buffer Pointer (ePXP_S0VBUF)

S0 Chroma (V/Cr) Input Buffer Pointer. This register points to the beginning of the Source 0 V/Cr input buffer. In two-plane operation, this register is not used.

This register contains the pointer to the Chroma V/Cr buffer when performing colorspace conversion. This register is unused when processing RGB data.

EXAMPLE

```
PXP_S0BUF_WR(image_y);   // Y (luma) image data
PXP_S0UBUF_WR(image_u);  // U (Cb) image data
PXP_S0VBUF_WR(image_v);  // V (Cr) image data
```

Address: ePXP_S0VBUF is 4100_C000h base + 70h offset = 4100_C070h

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

ePXP_S0VBUF field descriptions

Field	Description
31–0 ADDR	Address pointer for the S0 (video) V/Cr Chroma input buffer. The address MUST be word-aligned for proper ePXP operation.

43.4.9 ePXP Source 0 (video) Buffer Parameters (ePXP_S0PARAM)

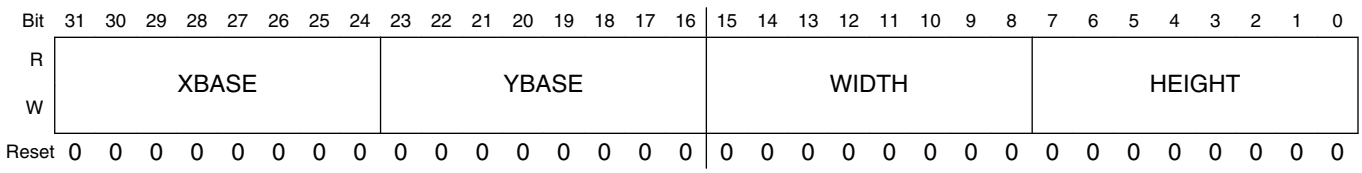
This register contains buffer information for the S0 input RGB/YUV buffer.

The S0 Parameter register contains the size of the S0 input buffer (WIDTH, HEIGHT) as well as provides an offset for the display of this buffer within the output frame buffer (XBASE,YBASE). All four values are in terms of NxN pixel blocks. In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
PXP_S0PARAM_WR(0x0101281E); // S0 buffer will appear at offset (8,8) in the output buffer.
                            // the size is 0x28 (40*8=320 pixels) by 0x1E (30*8=240
pixels)
```

Address: ePXP_S0PARAM is 4100_C000h base + 80h offset = 4100_C080h



ePXP_S0PARAM field descriptions

Field	Description
31–24 XBASE	This field indicates the horizontal offset location (in NxN block) of the S0 buffer within the output frame buffer.
23–16 YBASE	This field indicates the vertical offset location (in NxN block) of the S0 buffer within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

43.4.10 Source 0 Background Color (ePXP_S0BACKGROUND)

S0 Background Pixel Color. This register provides a pixel value used when processing blocks outside of the region specified by the S0SIZE register. This value can effectively be used to set the color of the letterboxing region around a video image.

This register contains a pixel value to be used for any S0 blocks that fall outside the S0 extents. This is effectively a background or letterbox color.

EXAMPLE

```
PXP_S0BACKGROUND_WR(0x00000000);  // letterbox is black
PXP_S0BACKGROUND_WR(0x00800000);  // letterbox is dark red
PXP_S0BACKGROUND_WR(0x00008000);  // letterbox is dark green
PXP_S0BACKGROUND_WR(0x00000080);  // letterbox is dark blue
```

Address: ePXP_S0BACKGROUND is 4100_C000h base + 90h offset = 4100_C090h

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

ePXP_S0BACKGROUND field descriptions

Field	Description
31–0 COLOR	Background color (in 32bpp format) for any pixels not in the S0 buffer range specified in the S0SIZE register.

43.4.11 Source 0 Cropping Register (ePXP_S0CROP)

This register contains controls for image/video cropping. XBASE and YBASE select the origin of the S0 buffer for ePXP operations. The WIDTH and HEIGHT determine the visible size of the selected region in the output frame buffer. Software should program the input framebuffer cropped width/height values into these fields. Cropping is applied in the output buffer, therefore after any scaling operations. Scaled regions may need to be cropped to avoid artifacts at the edge of a scaled region.

The cropping register can be used to specify cropping extents for S0 plane in the output buffer. It is only used if the CROP bit is set in the ePXP_CTRL register is set. When this bit is not set, no cropping of the input image will be performed and the ePXP will default to using the S0 WIDTH and HEIGHT parameters. Cropping should always be used when scaling images since the PXP cannot determine the scaled image size. In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
PXP_S0CROP_WR(0x02021810);  // S0 origin is at (16,16) -- 0x0202
                               // output width is 192 (0x18->24*8=192 pixels)
                               // output height is 128 (0x10->16*8=128 pixels)
```

Address: ePXP_S0CROP is 4100_C000h base + A0h offset = 4100_C0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	XBASE								YBASE								WIDTH								HEIGHT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ePXP_S0CROP field descriptions

Field	Description
31–24 XBASE	This field indicates the horizontal offset (in terms of N-pixel blocks) into the S0 buffer which is considered the origin of the image. This allows selection of a subset of a source image for processing.
23–16 YBASE	This field indicates the vertical offset (in terms of N-pixel blocks) into the S0 buffer which is considered the origin of the image. This allows selection of a subset of a source image for processing.
15–8 WIDTH	Output buffer cropped video width (in terms of N pixel blocks). This field should be programmed to the desired cropped width of the S0 plane in the output buffer. When scaling is not used, this value is effectively the width of the input buffer that should appear in the output buffer. For scaling operations, it's important that this field be programmed to the width of the scaled size of the S0 output image.
7–0 HEIGHT	Input buffer cropped video height (in terms of N pixel blocks). This field should be programmed to the desired cropped height of the S0 plan in the output buffer. When scaling is not used, this value is effectively the height of the input buffer that should appear in the output buffer. For scaling operations, it's important that this field be programmed to the height of the scaled size of the S0 output image.

43.4.12 Source 0 Scale Factor Register (ePXP_S0SCALE)

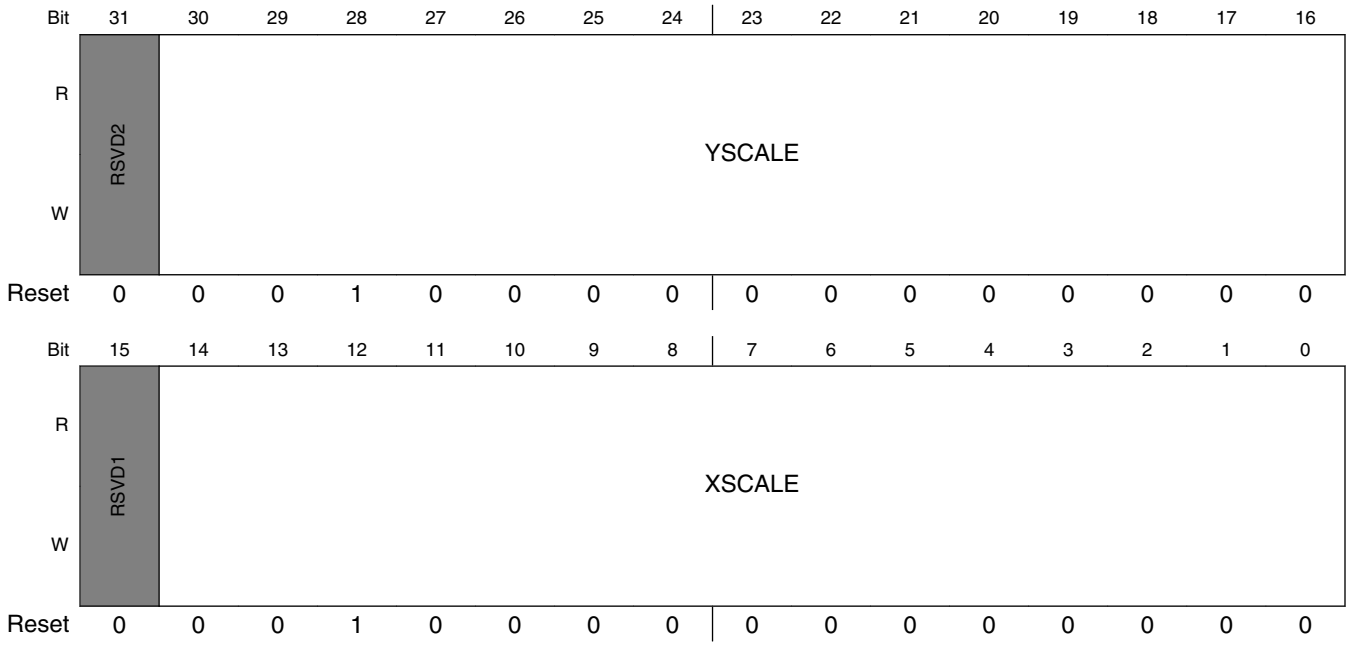
S0 Scale Factor. This register provides the scale factor for the S0 (video) buffer.

The maximum down scaling factor is 1/4 such that the output image in either axis is 1/4th the size of the source. The maximum up scaling factor is 2^12 for either axis. The reciprocal of the scale factor should be loaded into this register. To reduce the S0 buffer by a factor of two in the output frame buffer, a value of 10.0000_0000_0000 should be loaded into this register. The scale up by a factor of 4, the value of 1/4, or 00.0100_0000_0000, should be loaded into this register. To scale up by 8/5, the value of 00.1010_0000_0000 should be loaded.

EXAMPLE

```
PXP_S0SCALE_WR(0x10001000);  // 1:1  scaling (0x1.000)
PXP_S0SCALE_WR(0x08000800);  // 2x   scaling (0x0.800)
PXP_S0SCALE_WR(0x20002000);  // 1/2x scaling (0x2.000)
```

Address: ePXP_S0SCALE is 4100_C000h base + B0h offset = 4100_C0B0h



ePXP_S0SCALE field descriptions

Field	Description
31 RSVD2	Reserved, always set to zero.
30–16 YSCALE	This is a three bit integer and 12 bit fractional representation (###.####_####_####) of the Y scaling factor for the S0 source buffer. The maximum value programmed should be 4 since scaling down by a factor greater than 4 is not supported.
15 RSVD1	Reserved, always set to zero.
14–0 XSCALE	This is a three bit integer and 12 bit fractional representation (###.####_####_####) of the X scaling factor for the S0 source buffer. The maximum value programmed should be 4 since scaling down by a factor greater than 4 is not supported.

43.4.13 Source 0 Scale Offset Register (ePXP_S0OFFSET)

S0 Scale Offset. This register provides the initial scale offset for the S0 (video) buffer.

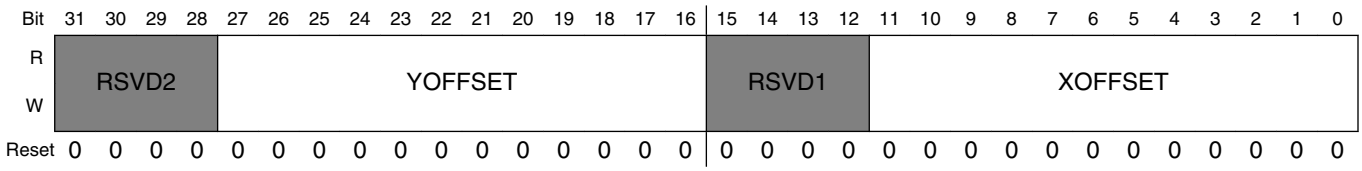
The X and Y offset provides the ability to access the source image with a per pixel or per sub-pixel granularity. To shift the source input image by a single pixel, for example, a value of 0x200 (for 8x8 block size) would be loaded into this offset field. For a 8x8 block size, 0x200 (or 1/8), will provide a fixed offset of 1 pixel for the entire ePXP operation. With this setting for 16x16 block size, the value of 0x200 will provide a fixed offset of 2 pixels since 1/8 of a 16 pixel block is 2. The fixed offset values can also be used for sub-pixel adjustments in the bilinear scaling filter. For example, when scaling an image down

by a factor of 2, an initial offset of 0x0 would result in sub-sampling every other pixel. If a fixed offset of 0x100 (1/16) with 8x8 block size selected is programmed, all pixels are used in scaling the final output pixel value.

EXAMPLE

```
Empty Example.
PXP_S0SCALE_WR(0x20002000);   // 1/2x scaling (0x2.000)
PXP_S0OFFSET_WR(0x01000100);  // half-pixel offset for 8x8 block size in both X and Y to
ensure averaging versus pixel replication
```

Address: ePXP_S0OFFSET is 4100_C000h base + C0h offset = 4100_C0C0h



ePXP_S0OFFSET field descriptions

Field	Description
31–28 RSVD2	Reserved, always set to zero.
27–16 YOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the Y scaling offset. This represents a fixed block offset which gets added to the scaled block address to determine source data for the scaling engine.
15–12 RSVD1	Reserved, always set to zero.
11–0 XOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the X scaling offset. This represents a fixed block offset which gets added to the scaled block address to determine source data for the scaling engine.

43.4.14 Color Space Conversion Coefficient Register 0 (ePXP_CSCCOEF0)

This register contains color space conversion coefficients in two's complement notation.

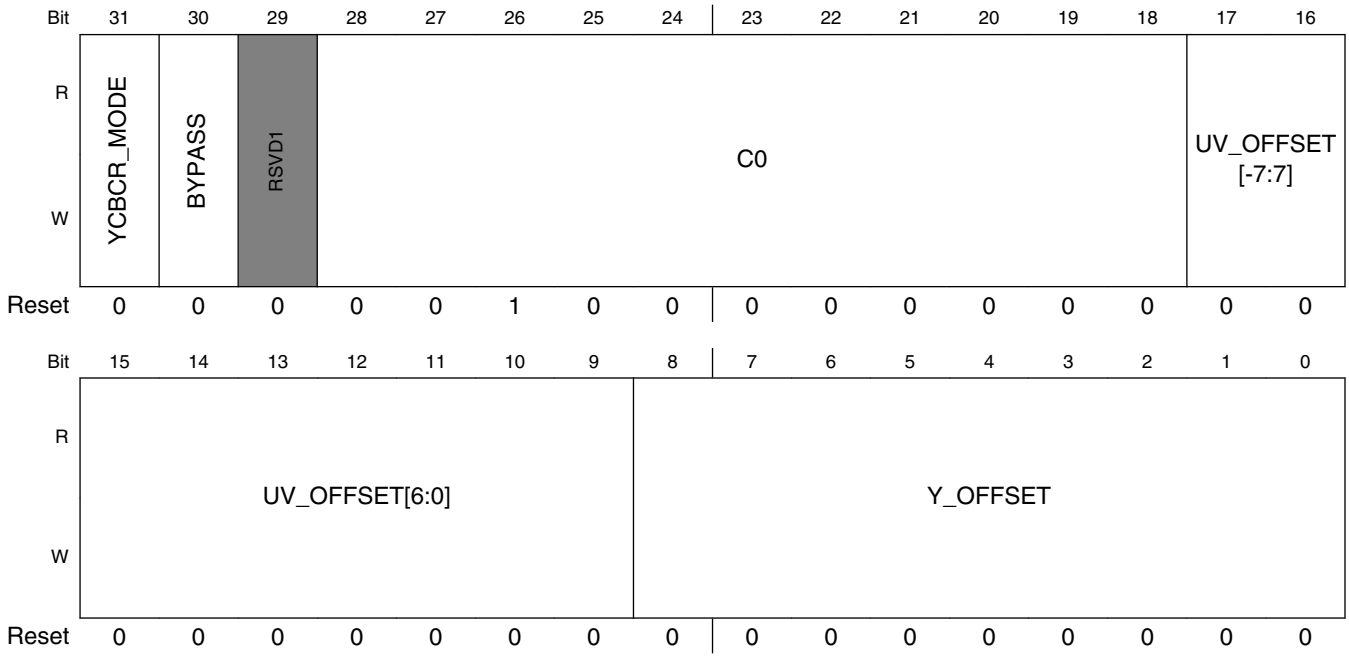
The Coefficient 0 register contains coefficients used in the color space conversion algorithm. The Y and UV offsets are added to the source buffer to normalize them before the conversion. C0 is the coefficient that is used to multiply the luma component of the data for all three RGB components.

EXAMPLE

```
//  The equations used for Colorspace conversion are:
//     R = C0*(Y+YOFFSET)                    + C1(V+UV_OFFSET)
//     G = C0*(Y+YOFFSET) + C3(U+UV_OFFSET) + C2(V+UV_OFFSET)
//     R = C0*(Y+YOFFSET) + C4(U+UV_OFFSET)

   PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
```

```
PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: ePXP_CSCCOEF0 is 4100_C000h base + D0h offset = 4100_C0D0h



ePXP_CSCCOEF0 field descriptions

Field	Description
31 YCBCR_MODE	Set to 1 when performing YCbCr conversion to RGB. Set to 0 when converting YUV to RGB data. This bit changes the behavior of the scaler when performing U/V scaling.
30 BYPASS	Bypass the CSC unit in the scaling engine. When set to logic 1, bypass is enabled and the output pixels will be in the YUV/YCbCr color space. When set to logic 0, the CSC unit is enabled and the pixels will be converted based on the programmed coefficients.
29 RSVD1	Reserved, always set to zero.
28–18 C0	Two's complement Y multiplier coefficient. YUV=0x100 (1.000) YCbCr=0x12A (1.164)
17–9 UV_OFFSET	Two's complement phase offset implicit for CbCr data. Generally used for YCbCr to RGB conversion. YCbCr=0x180, YUV=0x000 (typically -128 or 0x180 to indicate normalized -0.5 to 0.5 range)
8–0 Y_OFFSET	Two's complement amplitude offset implicit in the Y data. For YUV, this is typically 0 and for YCbCr, this is typically -16 (0x1F0)

43.4.15 Color Space Conversion Coefficient Register 1 (ePXP_CSCCOEF1)

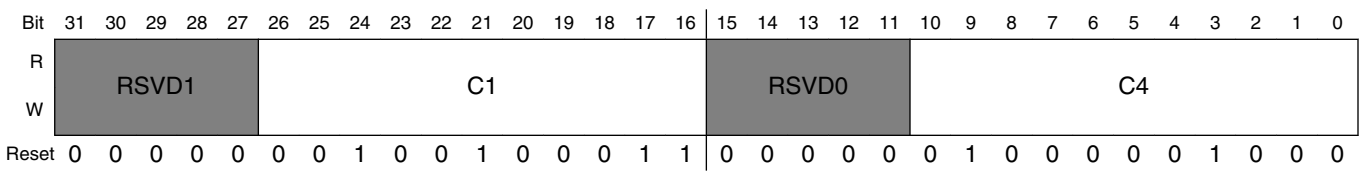
This register contains color space conversion coefficients in two's complement notation.

The Coefficient 1 register contains coefficients used in the color space conversion algorithm. C1 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the red component. C4 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the blue component. Both values should be coded as a two's complement fixed point number with 8 bits right of the decimal.

EXAMPLE

```
PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
   PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: ePXP_CSCCOEF1 is 4100_C000h base + E0h offset = 4100_C0E0h



ePXP_CSCCOEF1 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C1	Two's complement Red V/Cr multiplier coefficient. YUV=0x123 (1.140) YCbCr=0x198 (1.596)
15–11 RSVD0	Reserved, always set to zero.
10–0 C4	Two's complement Blue U/Cb multiplier coefficient. YUV=0x208 (2.032) YCbCr=0x204 (2.017)

43.4.16 Color Space Conversion Coefficient Register 2 (ePXP_CSCCOEF2)

This register contains color space conversion coefficients in two's complement notation.

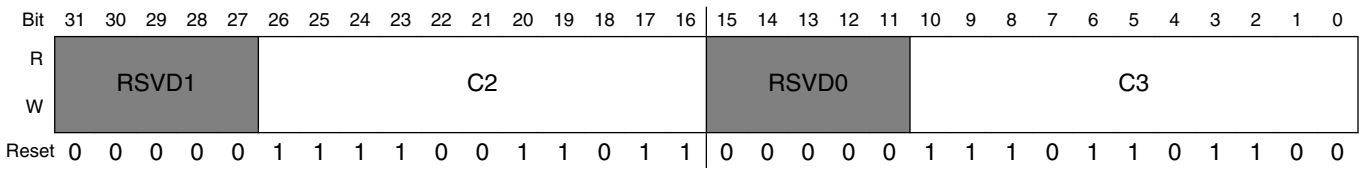
The Coefficient 2 register contains coefficients used in the color space conversion algorithm. C2 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the green component. C3 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the green component. Both values should be coded as a two's complement fixed point number with 8 bits right of the decimal.

EXAMPLE

```
// NOTE: The default values for the CSCCOEF2 register are incorrect.  C2 should be 0x76B and
C3 should be 0x79C for proper operation.

   PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
```

```
PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: ePXP_CSCCOEF2 is 4100_C000h base + F0h offset = 4100_C0F0h



ePXP_CSCCOEF2 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C2	Two's complement Green V/Cr multiplier coefficient. YUV=0x76B (-0.581) YCbCr=0x730 (-0.813)
15–11 RSVD0	Reserved, always set to zero.
10–0 C3	Two's complement Green U/Cb multiplier coefficient. YUV=0x79C (-0.394) YCbCr=0x79C (-0.392)

43.4.17 ePXP Next Frame Pointer (ePXP_NEXT*n*)

This register contains a pointer to a data structure used to reload the ePXP registers at the end of the current frame.

To enable this functionality, software must write this register while the ePXP is processing the current data frame (if the ePXP is currently idle, this will also initiate an immediate load of registers from the pointer). The process of writing this register (WRITE operation) will set a semaphore in hardware to notify the control logic that a register reload operation must be performed when the current frame processing is complete. At the end of a frame, the ePXP will fetch the register settings from this location, signal an interrupt to software, then proceed with rendering the next frame of data. Software may cancel the reload operation by issuing a CLEAR operation to this register. SET and TOGGLE operations should not be used when addressing this register. All registers will be reloaded with the exception of the following: STAT, CSCCOEFn, NEXT, VERSION. All other registers will be loaded in the order they appear in the register map. Once the pointer's contents have been loaded into the ePXP's registers, the NEXT_IRQ interrupt will be issued (see the ePXP_STATUS register).

EXAMPLE

```
// create register command structure in memory
  u32* pxp_commands0[48], pxp_commands1;
  u32  rc;
```

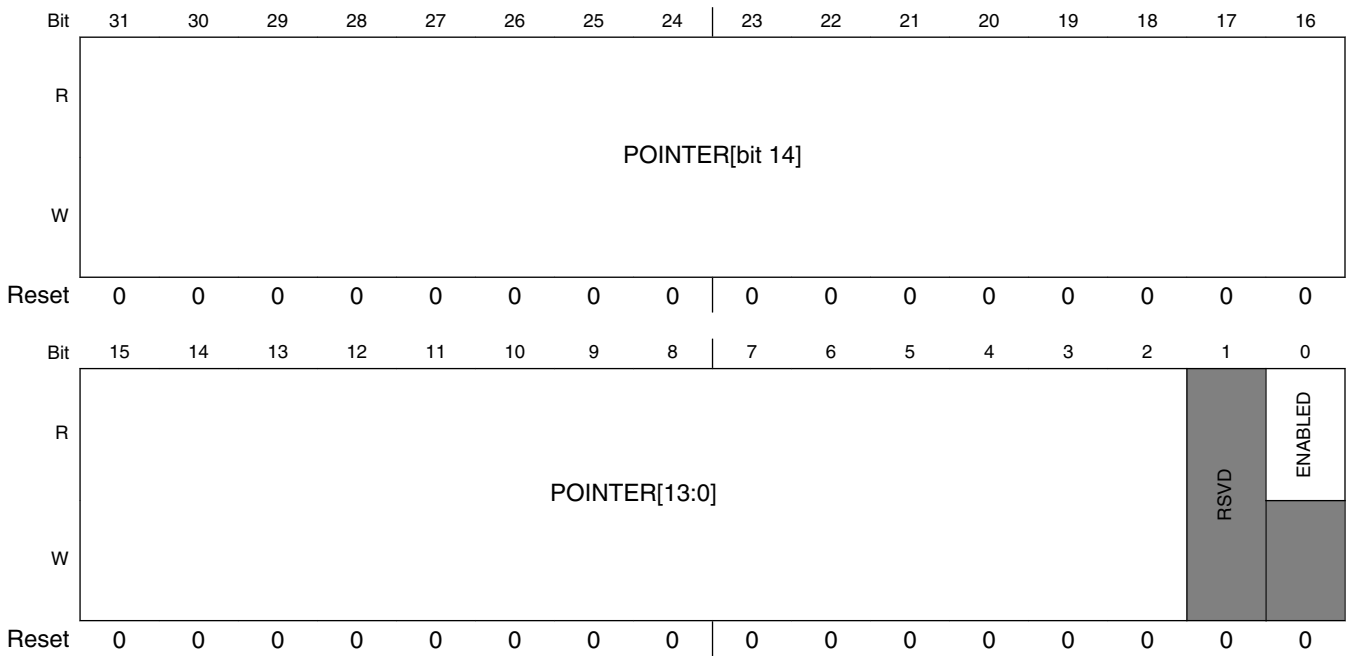
```
// initialize control structure for frame 0
  pxp_commands0[0] = ...; // CTRL
  pxp_commands0[1] = ...; // OUT Buffer
  ...
  pxp_commands0[47] = ..; // Overlay7 param2

// initialize control structure for frame 1
  pxp_commands1[0] = ...; // CTRL
  pxp_commands1[1] = ...; // OUT Buffer
  ...
  pxp_commands1[47] = ..; // Overlay7 param2

// poll until a command isn't queued
  while (rc=PXP_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  PXP_NEXT_WR(pxp_commands0);  // enable PXP operation 0 via command pointer

// poll until first command clears
  while (rc=PXP_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  PXP_NEXT_WR(pxp_commands1);  // enable PXP operation 1 via command pointer
```

Addresses: ePXP_NEXT is 4100_C000h base + 100h offset = 4100_C100h

ePXP_NEXT_SET is 4100_C000h base + 104h offset = 4100_C104h

ePXP_NEXT_CLR is 4100_C000h base + 108h offset = 4100_C108h

ePXP_NEXT_TOG is 4100_C000h base + 10Ch offset = 4100_C10Ch



ePXP_NEXT*n* field descriptions

Field	Description
31–2 POINTER	A pointer to a data structure containing register values to be used when processing the next frame. The pointer must be 32-bit aligned and should reside in on-chip or off-chip memory.
1 RSVD	Reserved, always set to zero.
0 ENABLED	Indicates that the "next frame" functionality has been enabled. This bit reflects the status of the hardware semaphore indicating that a reload operation is pending at the end of the current frame.

43.4.18 ePXP S0 Color Key Low (ePXP_S0COLORKEYLOW)

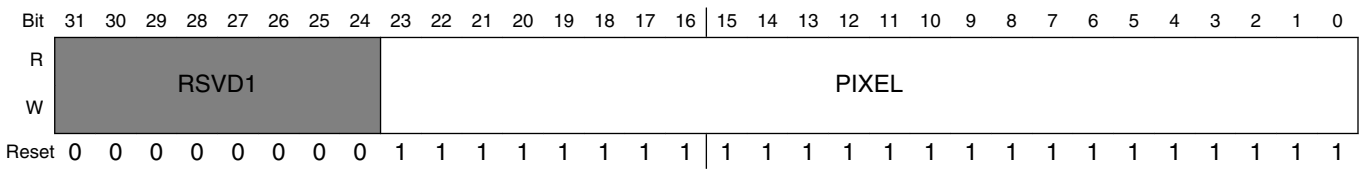
This register contains the color key low value for the S0 buffer.

When processing an image, the if the ePXP finds a pixel in the background image with a color that falls in the range from the S0COLORKEYLOW to S0COLORKEYHIGH range, it will substitute the color found in the matching overlay. If no overlay is present or if the overlay also matches its colorkey range, the s0background color is used.

EXAMPLE

```
// colorkey values between
PXP_S0COLORKEYLOW_WR (0x008000); // medium green and
PXP_S0COLORKEYHIGH_WR(0x00FF00); // light green
```

Address: ePXP_S0COLORKEYLOW is 4100_C000h base + 180h offset = 4100_C180h



ePXP_S0COLORKEYLOW field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
23–0 PIXEL	Low range of RGB color key applied to S0 buffer. To disable S0 colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

43.4.19 ePXP S0 Color Key High (ePXP_S0COLORKEYHIGH)

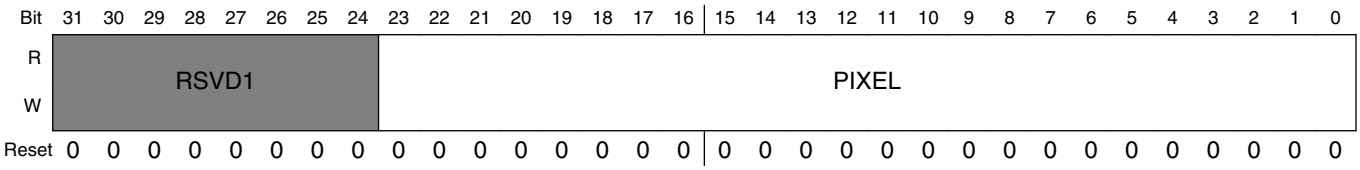
This register contains the color key high value for the S0 buffer.

When processing an image, if the ePXP finds a pixel in the background image with a color that falls in the range from the S0COLORKEYLOW to S0COLORKEYHIGH range, it will substitute the color found in the matching overlay. If no overlay is present or if the overlay also matches its colorkey range, the s0background color is used.

EXAMPLE

```
// colorkey values between
PXP_S0COLORKEYLOW_WR (0x008000); // medium green and
PXP_S0COLORKEYHIGH_WR(0x00FF00); // light green
```

Address: ePXP_S0COLORKEYHIGH is 4100_C000h base + 190h offset = 4100_C190h



ePXP_S0COLORKEYHIGH field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
23–0 PIXEL	High range of RGB color key applied to S0 buffer. To disable S0 colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

43.4.20 ePXP Overlay Color Key Low (ePXP_OLCOLORKEYLOW)

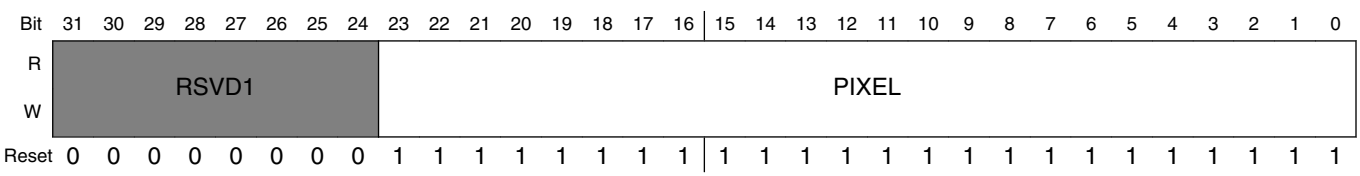
This register contains the color key low value for the OL buffer.

When processing an image, the if the ePXP finds a pixel in the current overlay image with a color that falls in the range from the OLCOLORKEYLOW to OLCOLORKEYHIGH range, it will use the S0 pixel value for that location. If no S0 image is present or if the S0 image also matches its colorkey range, the s0background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
// colorkey values between
PXP_OLCOLORKEYLOW_WR (0x000000); // black and
PXP_OLCOLORKEYHIGH_WR(0x800000); // medium red
```

Address: ePXP_OLCOLORKEYLOW is 4100_C000h base + 1A0h offset = 4100_C1A0h



ePXP_OLCOLORKEYLOW field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
23–0 PIXEL	Low range of RGB color key applied to OL buffer. Each overlay has an independent colorkey enable.

43.4.21 ePXP Overlay Color Key High (ePXP_OLCOLORKEYHIGH)

This register contains the color key high value for the OL buffer.

When processing an image, the if the ePXP finds a pixel in the current overlay image with a color that falls in the range from the OLCOLORKEYLOW to OLCOLORKEYHIGH range, it will use the S0 pixel value for that location. If no S0 image is present or if the S0 image also matches its colorkey range, the s0background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
// colorkey values between
PXP_OLCOLORKEYLOW_WR (0x000000); // black and
PXP_OLCOLORKEYHIGH_WR(0x800000); // medium red
```

Address: ePXP_OLCOLORKEYHIGH is 4100_C000h base + 1B0h offset = 4100_C1B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD1								PIXEL																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ePXP_OLCOLORKEYHIGH field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
23–0 PIXEL	High range of RGB color key applied to OL buffer. Each overlay has an independent colorkey enable.

43.4.22 ePXP Debug Register (ePXP_DEBUG)

This register returns selected debug register values.

Debug register. Select the appropriate register in debug control and the values are returned here.

EXAMPLE

Address: ePXP_DEBUG is 4100_C000h base + 1E0h offset = 4100_C1E0h

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

ePXP_DEBUG field descriptions

Field	Description
31–0 DATA	Debug data

43.4.23 ePXP Overlay n Buffer Pointer (ePXP_OL*n*)

Overlay n Buffer Address Pointer. This register points to the beginning of the RGB Overlay n input buffer.

This register is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
PXP_OLn_WR(n,overlay_ptr);
```

Addresses: ePXP_OL0 is 4100_C000h base + 200h offset = 4100_C200h
ePXP_OL1 is 4100_C000h base + 240h offset = 4100_C240h
ePXP_OL2 is 4100_C000h base + 280h offset = 4100_C280h
ePXP_OL3 is 4100_C000h base + 2C0h offset = 4100_C2C0h
ePXP_OL4 is 4100_C000h base + 300h offset = 4100_C300h
ePXP_OL5 is 4100_C000h base + 340h offset = 4100_C340h
ePXP_OL6 is 4100_C000h base + 380h offset = 4100_C380h
ePXP_OL7 is 4100_C000h base + 3C0h offset = 4100_C3C0h

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

ePXP_OL*n* field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay n buffer. The address MUST be word-aligned for proper ePXP operation.

43.4.24 ePXP Overlay n Size (ePXP_OL__SIZE)

This register contains buffer size/location information for the Overlay n input buffer.

This register contains information about Overlay n indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
PXP_OLnSIZE_WR(n,0x10000401); // 32x8 overlay at offset +128+0
```

Addresses: ePXP_OL__SIZE0 is 4100_C000h base + 210h offset = 4100_C210h
ePXP_OL__SIZE1 is 4100_C000h base + 250h offset = 4100_C250h
ePXP_OL__SIZE2 is 4100_C000h base + 290h offset = 4100_C290h
ePXP_OL__SIZE3 is 4100_C000h base + 2D0h offset = 4100_C2D0h
ePXP_OL__SIZE4 is 4100_C000h base + 310h offset = 4100_C310h
ePXP_OL__SIZE5 is 4100_C000h base + 350h offset = 4100_C350h
ePXP_OL__SIZE6 is 4100_C000h base + 390h offset = 4100_C390h
ePXP_OL__SIZE7 is 4100_C000h base + 3D0h offset = 4100_C3D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ePXP_OL__SIZE*n* field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

43.4.25 ePXP Overlay n Parameters (ePXP_OL__PARAM)

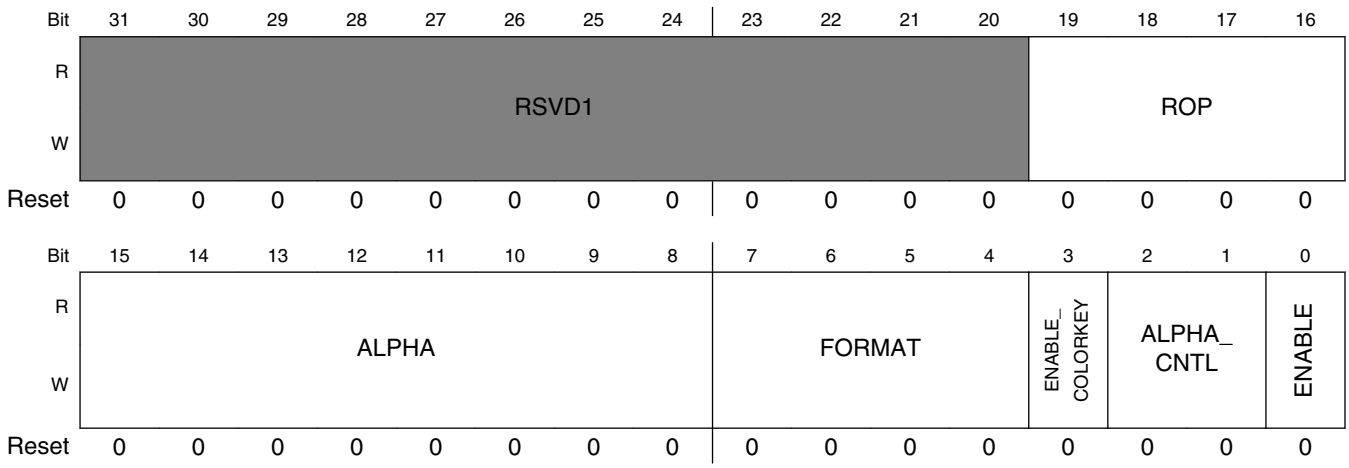
This register contains buffer parameters for the Overlay n input buffer.

The S1 Overlay n Parameter register provides additional controls for Overlay n.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
PXP_OLnPARAM_WR(n,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Addresses: ePXP_OL__PARAM0 is 4100_C000h base + 220h offset = 4100_C220h
ePXP_OL__PARAM1 is 4100_C000h base + 260h offset = 4100_C260h
ePXP_OL__PARAM2 is 4100_C000h base + 2A0h offset = 4100_C2A0h
ePXP_OL__PARAM3 is 4100_C000h base + 2E0h offset = 4100_C2E0h
ePXP_OL__PARAM4 is 4100_C000h base + 320h offset = 4100_C320h
ePXP_OL__PARAM5 is 4100_C000h base + 360h offset = 4100_C360h
ePXP_OL__PARAM6 is 4100_C000h base + 3A0h offset = 4100_C3A0h
ePXP_OL__PARAM7 is 4100_C000h base + 3E0h offset = 4100_C3E0h



ePXP_OL__PARAM*n* field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field. 0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0 0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay n. 0x0 **ARGB8888** — 32-bit pixels with alpha

Table continues on the next page...

ePXP_OL__PARAM*n* field descriptions (continued)

Field	Description
	0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.
0 ENABLE	Indicates that the overlay is active for this operation.

43.4.26 Color Space Conversion Control Register. (ePXP_CSC2CTRL)

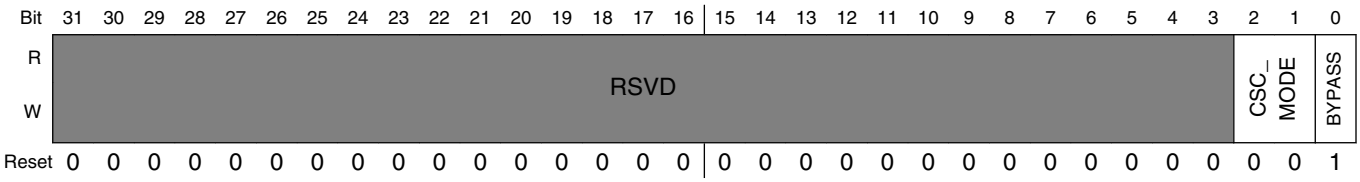
This register contains the control registers to configure the CSC module.

The CSC control register will configure the CSC module to perform color space conversion between the RGB/YUV/YCbCr color spaces.

EXAMPLE

```
//Converting from YUV/YCbCr color spaces to the RGB color space uses the
   //following equation structure:
   //
   // R = A1(Y-D1) + A2(U-D2) + A3(V-D3)
   // G = B1(Y-D1) + B2(U-D2) + B3(V-D3)
   // B = C1(Y-D1) + C2(U-D2) + C3(V-D3)
   //
   //Converting from the RGB color space to YUV/YCbCr color spaces uses the
   //following equation structure:
   //
   // Y = A1*R + A2*G + A3*B + D1
   // U = B1*R + B2*G + B3*B + D2
   // V = C1*R + C2*G + C3*B + D3
   //
   //All math is signed, so all coefficients come in as two's comp numbers
   //
```

Address: ePXP_CSC2CTRL is 4100_C000h base + 400h offset = 4100_C400h



ePXP_CSC2CTRL field descriptions

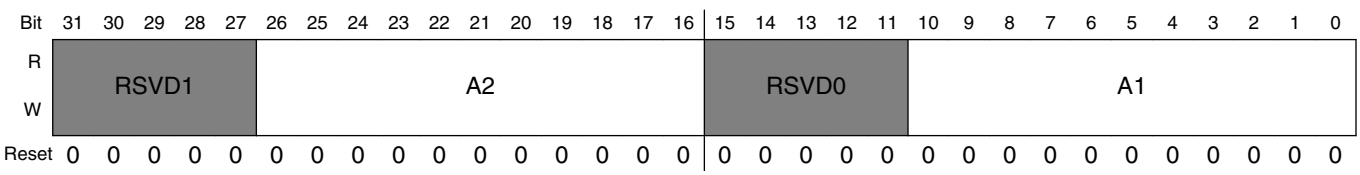
Field	Description
31–3 RSVD	Reserved, always set to zero.
2–1 CSC_MODE	This field controls how the CSC unit operates on pixels when the CSC is not bypassed. 0x0 **YUV2RGB** — Convert from YUV to RGB. 0x1 **YCbCr2RGB** — Convert from YCbCr to RGB. 0x2 **RGB2YUV** — Convert from RGB to YUV. 0x3 **RGB2YCbCr** — Convert from RGB to YCbCr.
0 BYPASS	This bit controls whether the pixels entering the CSC2 unit get converted or not. When BYPASS is set, no operations occur on the pixels. When BYPASS is cleared, the selected CSC operation takes place.

43.4.27 Color Space Conversion Coefficient Register 0 (ePXP_CSC2COEF0)

This register contains color space conversion coefficients in two's complement notation.

EXAMPLE

Address: ePXP_CSC2COEF0 is 4100_C000h base + 410h offset = 4100_C410h



ePXP_CSC2COEF0 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 A2	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
10–0 A1	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

43.4.28 Color Space Conversion Coefficient Register 1 (ePXP_CSC2COEF1)

This register contains color space conversion coefficients in two's complement notation.

EXAMPLE

Address: ePXP_CSC2COEF1 is 4100_C000h base + 420h offset = 4100_C420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD1					B1											RSVD0					A3											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ePXP_CSC2COEF1 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 B1	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
10–0 A3	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

43.4.29 Color Space Conversion Coefficient Register 2 (ePXP_CSC2COEF2)

This register contains color space conversion coefficients in two's complement notation.

EXAMPLE

Address: ePXP_CSC2COEF2 is 4100_C000h base + 430h offset = 4100_C430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD1					B3											RSVD0					B2											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ePXP_CSC2COEF2 field descriptions

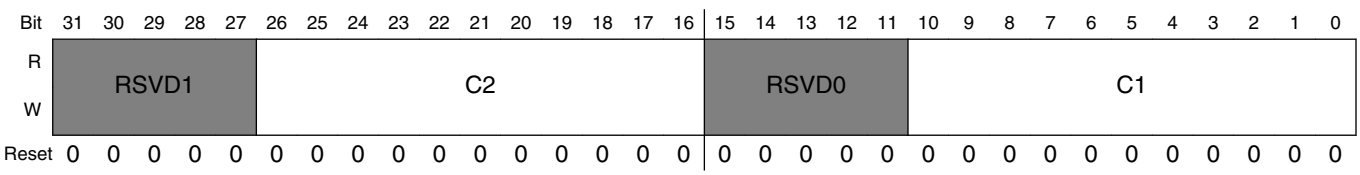
Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 B3	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
10–0 B2	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

43.4.30 Color Space Conversion Coefficient Register 3 (ePXP_CSC2COEF3)

This register contains color space conversion coefficients in two's complement notation.

EXAMPLE

Address: ePXP_CSC2COEF3 is 4100_C000h base + 440h offset = 4100_C440h



ePXP_CSC2COEF3 field descriptions

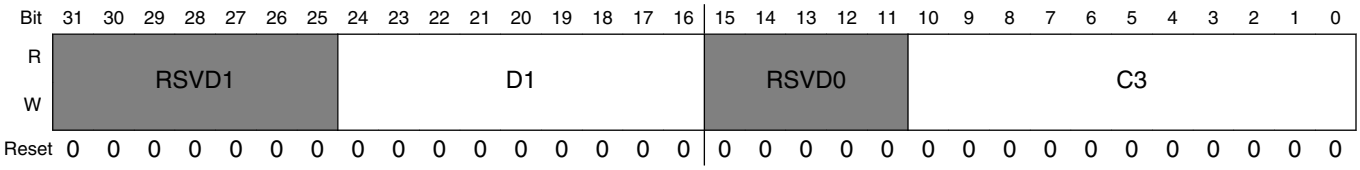
Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C2	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.
15–11 RSVD0	Reserved, always set to zero.
10–0 C1	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

43.4.31 Color Space Conversion Coefficient Register 4 (ePXP_CSC2COEF4)

This register contains color space conversion coefficients in two's complement notation.

EXAMPLE

Address: ePXP_CSC2COEF4 is 4100_C000h base + 450h offset = 4100_C450h



ePXP_CSC2COEF4 field descriptions

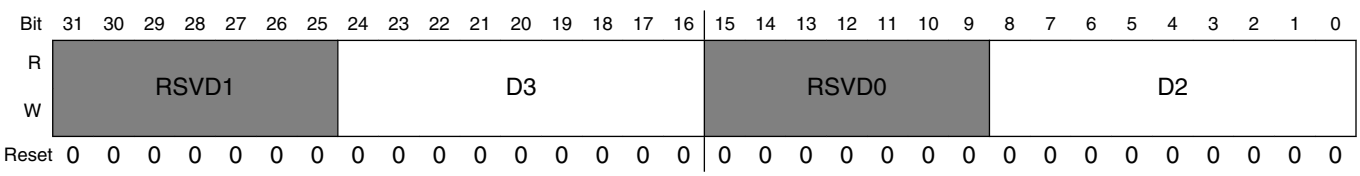
Field	Description
31–25 RSVD1	Reserved, always set to zero.
24–16 D1	Two's complement coefficient integer offset to be added.
15–11 RSVD0	Reserved, always set to zero.
10–0 C3	Two's complement coefficient offset. This coefficient has a sign bit, 2 bits integer, and 8 bits of fraction as ###.####_####.

43.4.32 Color Space Conversion Coefficient Register 5 (ePXP_CSC2COEF5)

This register contains color space conversion coefficients in two's complement notation.

EXAMPLE

Address: ePXP_CSC2COEF5 is 4100_C000h base + 460h offset = 4100_C460h



ePXP_CSC2COEF5 field descriptions

Field	Description
31–25 RSVD1	Reserved, always set to zero.
24–16 D3	Two's complement coefficient integer offset to be added.
15–9 RSVD0	Reserved, always set to zero.

Table continues on the next page...

ePXP_CSC2COEF5 field descriptions (continued)

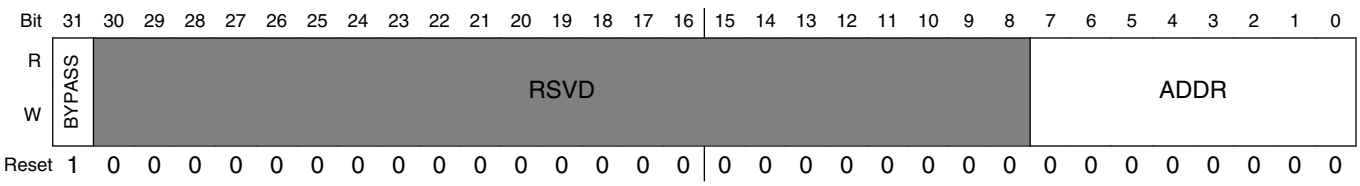
Field	Description
8–0 D2	Two's complement coefficient integer offset to be added.

43.4.33 Lookup Table Control Register. (ePXP_LUT_CTRL)

This register is used to access/control the Monochrome Lookup table.

EXAMPLE

Address: ePXP_LUT_CTRL is 4100_C000h base + 470h offset = 4100_C470h



ePXP_LUT_CTRL field descriptions

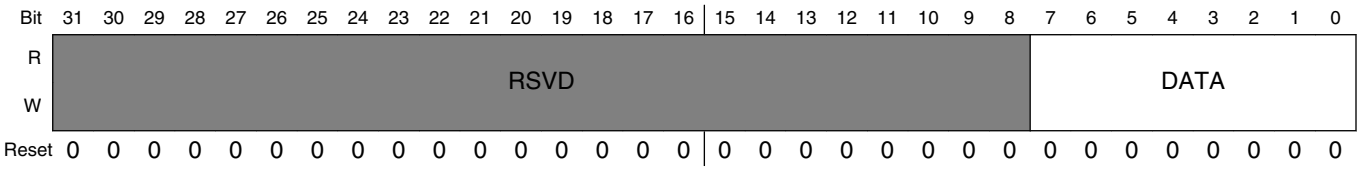
Field	Description
31 BYPASS	Set this bit to bypass the Y (Luma) lookup table. If Gamma correction is required, as for monochrome output, then clear this bit, program the LUT with valid data, and clear the bypass bit to enable the LUT resources.
30–8 RSVD	Reserved, always set to zero.
7–0 ADDR	LUT indexed address pointer. When the LUT data register is written, the contents of the LUT at the address specified by this address filed will be loaded. This address pointer will be incremented after the LUT data is written. This will provide recursive writes to the LUT data regeister to initialize the entire LUT array.

43.4.34 Lookup Table Data Register. (ePXP_LUT)

This register is used to load data into the lookup table.

EXAMPLE

Address: ePXP_LUT is 4100_C000h base + 480h offset = 4100_C480h



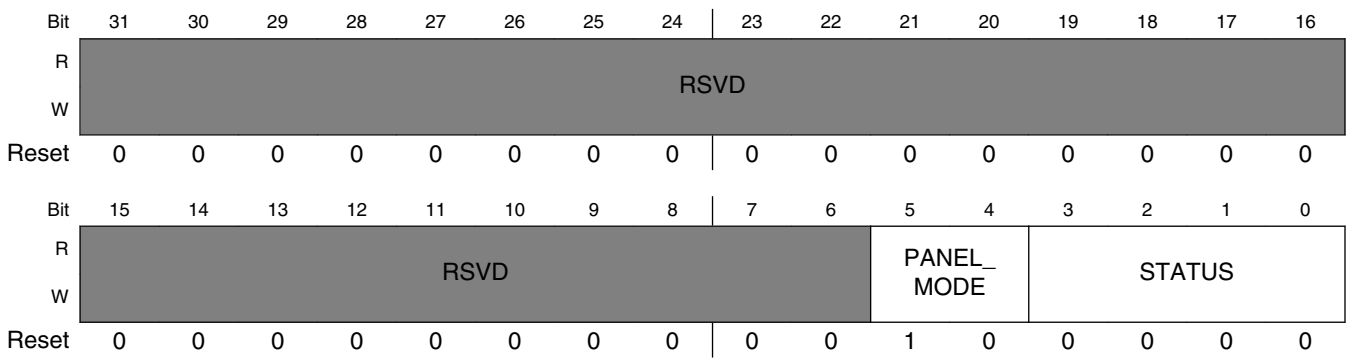
ePXP_LUT field descriptions

Field	Description
31–8 RSVD	Reserved, always set to zero.
7–0 DATA	Writting this field will write the data indexed by the ADDR field of the ePXP_LUT_CTRL register.

43.4.35 Histogram Control Register. (ePXP_HIST_CTRL)

Provides control and status registers for the ePXP's histogram classification algorithm.

Address: ePXP_HIST_CTRL is 4100_C000h base + 490h offset = 4100_C490h



ePXP_HIST_CTRL field descriptions

Field	Description
31–6 RSVD	Reserved, always set to zero.
5–4 PANEL_MODE	Specifies the EPDC panel grayscale depth. This value is used to specify the number of bits used in comparisons when matching pixels to histogram bins. All comparator values MUST be programmed such that their bit width is consistent with the value of this register field. For instance, if GRAY16 is selected, comparator values must be in the range of 0x0-0xF. 0x0 **GRAY4** — 4-bit grayscale 0x1 **GRAY8** — 8-bit grayscale 0x2 **GRAY16** — 16-bit grayscale 0x3 **GRAY32** — 32-bit grayscale

Table continues on the next page...

ePXP_HIST_CTRL field descriptions (continued)

Field	Description
3–0 STATUS	Indicates which histogram matched the processed bitmap. Bit[0] indicates that the bitmap pixels were fully contained within the HIST2 (black / white) histogram. Bit[1] indicates that the bitmap pixels were fully contained within the HIST4 (2-bit grayscale) histogram. Bit[2] indicates that the bitmap pixels were fully contained within the HIST8 (3-bit grayscale) histogram. Bit[3] indicates that the bitmap pixels were fully contained within the HIST16 (4-bit grayscale) histogram.

43.4.36 2-level Histogram Parameter Register. (ePXP_HIST2_PARAM)

This register specifies the valid values for a 2-level histogram. If all pixels in a bitmap match these two values, ePXP_HIST_CTRL[STATUS[0]] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with ePXP_HIST_CTRL[PANEL_MODE].

EXAMPLE

Address: ePXP_HIST2_PARAM is 4100_C000h base + 4A0h offset = 4100_C4A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD																RSVD1			VALUE1					RSVD0			VALUE0					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

ePXP_HIST2_PARAM field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–13 RSVD1	Reserved, always set to zero.
12–8 VALUE1	White value for 2-level histogram
7–5 RSVD0	Reserved, always set to zero.
4–0 VALUE0	Black value for 2-level histogram

43.4.37 4-level Histogram Parameter Register. (ePXP_HIST4_PARAM)

This register specifies the valid values for a 4-level histogram. If all pixels in a bitmap match these two values, ePXP_HIST_CTRL[STATUS[1]] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with ePXP_HIST_CTRL[PANEL_MODE].

EXAMPLE

Address: ePXP_HIST4_PARAM is 4100_C000h base + 4B0h offset = 4100_C4B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD3			VALUE3					RSVD2			VALUE2					RSVD1			VALUE1					RSVD0			VALUE0				
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

ePXP_HIST4_PARAM field descriptions

Field	Description
31–29 RSVD3	Reserved, always set to zero.
28–24 VALUE3	GRAY3 (White) value for 4-level histogram
23–21 RSVD2	Reserved, always set to zero.
20–16 VALUE2	GRAY2 value for 4-level histogram
15–13 RSVD1	Reserved, always set to zero.
12–8 VALUE1	GRAY1 value for 4-level histogram
7–5 RSVD0	Reserved, always set to zero.
4–0 VALUE0	GRAY0 (Black) value for 4-level histogram

43.4.38 8-level Histogram Parameter 0 Register. (ePXP_HIST8_PARAM0)

This register specifies four of the valid values for an 8-level histogram. If all pixels in a bitmap match these two values, ePXP_HIST_CTRL[STATUS[2]] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with ePXP_HIST_CTRL[PANEL_MODE].

EXAMPLE

Address: ePXP_HIST8_PARAM0 is 4100_C000h base + 4C0h offset = 4100_C4C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD3			VALUE3					RSVD2			VALUE2					RSVD1			VALUE1					RSVD0			VALUE0				
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ePXP_HIST8_PARAM0 field descriptions

Field	Description
31–29 RSVD3	Reserved, always set to zero.
28–24 VALUE3	GRAY3 value for 8-level histogram
23–21 RSVD2	Reserved, always set to zero.
20–16 VALUE2	GRAY2 value for 8-level histogram
15–13 RSVD1	Reserved, always set to zero.
12–8 VALUE1	GRAY1 value for 8-level histogram
7–5 RSVD0	Reserved, always set to zero.
4–0 VALUE0	GRAY0 (Black) value for 8-level histogram

43.4.39 8-level Histogram Parameter 1 Register. (ePXP_HIST8_PARAM1)

This register specifies four of the valid values for an 8-level histogram. If all pixels in a bitmap match these two values, ePXP_HIST_CTRL[STATUS[2]] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with ePXP_HIST_CTRL[PANEL_MODE].

Address: ePXP_HIST8_PARAM1 is 4100_C000h base + 4D0h offset = 4100_C4D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD7			VALUE7					RSVD6			VALUE6					RSVD5			VALUE5					RSVD4			VALUE4				
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	1

ePXP_HIST8_PARAM1 field descriptions

Field	Description
31–29 RSVD7	Reserved, always set to zero.
28–24 VALUE7	GRAY7 (White) value for 8-level histogram
23–21 RSVD6	Reserved, always set to zero.
20–16 VALUE6	GRAY6 value for 8-level histogram
15–13 RSVD5	Reserved, always set to zero.
12–8 VALUE5	GRAY5 value for 8-level histogram
7–5 RSVD4	Reserved, always set to zero.
4–0 VALUE4	GRAY4 value for 8-level histogram

43.4.40 16-level Histogram Parameter 0 Register. (ePXP_HIST16_PARAM0)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match these two values, ePXP_HIST_CTRL[STATUS[3]] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with ePXP_HIST_CTRL[PANEL_MODE].

Address: ePXP_HIST16_PARAM0 is 4100_C000h base + 4E0h offset = 4100_C4E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD3			VALUE3					RSVD2			VALUE2					RSVD1			VALUE1					RSVD0			VALUE0				
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

ePXP_HIST16_PARAM0 field descriptions

Field	Description
31–29 RSVD3	Reserved, always set to zero.
28–24 VALUE3	GRAY3 value for 16-level histogram
23–21 RSVD2	Reserved, always set to zero.
20–16 VALUE2	GRAY2 value for 16-level histogram
15–13 RSVD1	Reserved, always set to zero.
12–8 VALUE1	GRAY1 value for 16-level histogram
7–5 RSVD0	Reserved, always set to zero.
4–0 VALUE0	GRAY0 (Black) value for 16-level histogram

43.4.41 16-level Histogram Parameter Register. (ePXP_HIST16_PARAM1)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match these two values, ePXP_HIST_CTRL[STATUS[3]] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with ePXP_HIST_CTRL[PANEL_MODE].

Address: ePXP_HIST16_PARAM1 is 4100_C000h base + 4F0h offset = 4100_C4F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD7			VALUE7					RSVD6			VALUE6					RSVD5			VALUE5					RSVD4			VALUE4				
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0

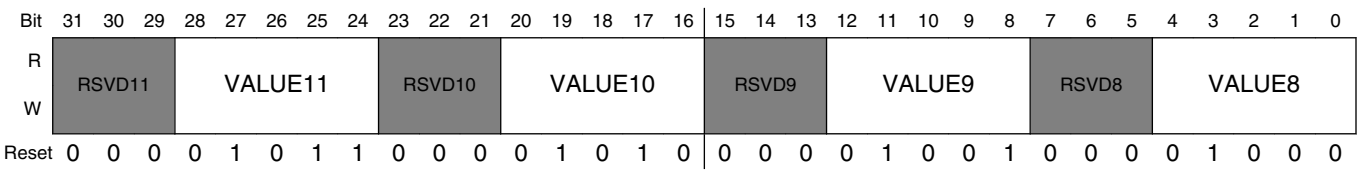
ePXP_HIST16_PARAM1 field descriptions

Field	Description
31–29 RSVD7	Reserved, always set to zero.
28–24 VALUE7	GRAY7 value for 16-level histogram
23–21 RSVD6	Reserved, always set to zero.
20–16 VALUE6	GRAY6 value for 16-level histogram
15–13 RSVD5	Reserved, always set to zero.
12–8 VALUE5	GRAY5 value for 16-level histogram
7–5 RSVD4	Reserved, always set to zero.
4–0 VALUE4	GRAY4 value for 16-level histogram

43.4.42 16-level Histogram Parameter Register. (ePXP_HIST16_PARAM2)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match these two values, ePXP_HIST_CTRL[STATUS[3]] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with ePXP_HIST_CTRL[PANEL_MODE].

Address: ePXP_HIST16_PARAM2 is 4100_C000h base + 500h offset = 4100_C500h



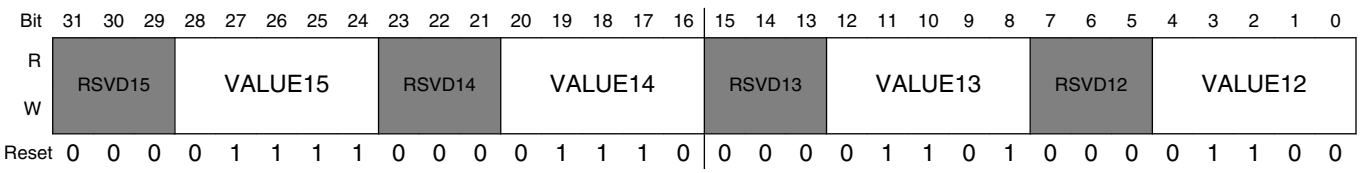
ePXP_HIST16_PARAM2 field descriptions

Field	Description
31–29 RSVD11	Reserved, always set to zero.
28–24 VALUE11	GRAY11 value for 16-level histogram
23–21 RSVD10	Reserved, always set to zero.
20–16 VALUE10	GRAY10 value for 16-level histogram
15–13 RSVD9	Reserved, always set to zero.
12–8 VALUE9	GRAY9 value for 16-level histogram
7–5 RSVD8	Reserved, always set to zero.
4–0 VALUE8	GRAY8 value for 16-level histogram

43.4.43 16-level Histogram Parameter Register. (ePXP_HIST16_PARAM3)

This register specifies four of the valid values for a 16-level histogram. If all pixels in a bitmap match these two values, ePXP_HIST_CTRL[STATUS[3]] will be set at the end of frame processing. All comparator values should be programmed such that they are consistent with ePXP_HIST_CTRL[PANEL_MODE].

Address: ePXP_HIST16_PARAM3 is 4100_C000h base + 510h offset = 4100_C510h



ePXP_HIST16_PARAM3 field descriptions

Field	Description
31–29 RSVD15	Reserved, always set to zero.
28–24 VALUE15	GRAY15 (White) value for 16-level histogram
23–21 RSVD14	Reserved, always set to zero.
20–16 VALUE14	GRAY14 value for 16-level histogram
15–13 RSVD13	Reserved, always set to zero.
12–8 VALUE13	GRAY13 value for 16-level histogram
7–5 RSVD12	Reserved, always set to zero.
4–0 VALUE12	GRAY12 value for 16-level histogram

Chapter 44
Quality of Service Controller (QoSC)

44.1 Introduction

This chapter describes the requirements for the Quality of Service Controller (herein referred to as QoSC). It describes the controller functionality and implementation in detail.

44.2 Overview of Quality of Service Controller APB Slave (QoSC)

The Quality of Service Controller APB Slave provides the following features:

- Provide software programmability of the AWQOS and ARQOS sideband signals of the master ports of the AXI fabric.
- Provide translation and passthrough of AXI QOS values to the AWPRIORITY and ARPRIORITY sideband signals of the DRAM Memory Controller (DRAM MC). Alternatively allow independent programmability of the DRAM MC priority sideband signals.
- Automatic qos/priority boost for EPDC and eLCDIF masters.
- Per port disabling of AXI master port traffic.

44.3 Top-Level Symbol and Functional Overview

Figure 44-1 shows the basic functional connections of the QoSC with other logic internal to the i.MX50.

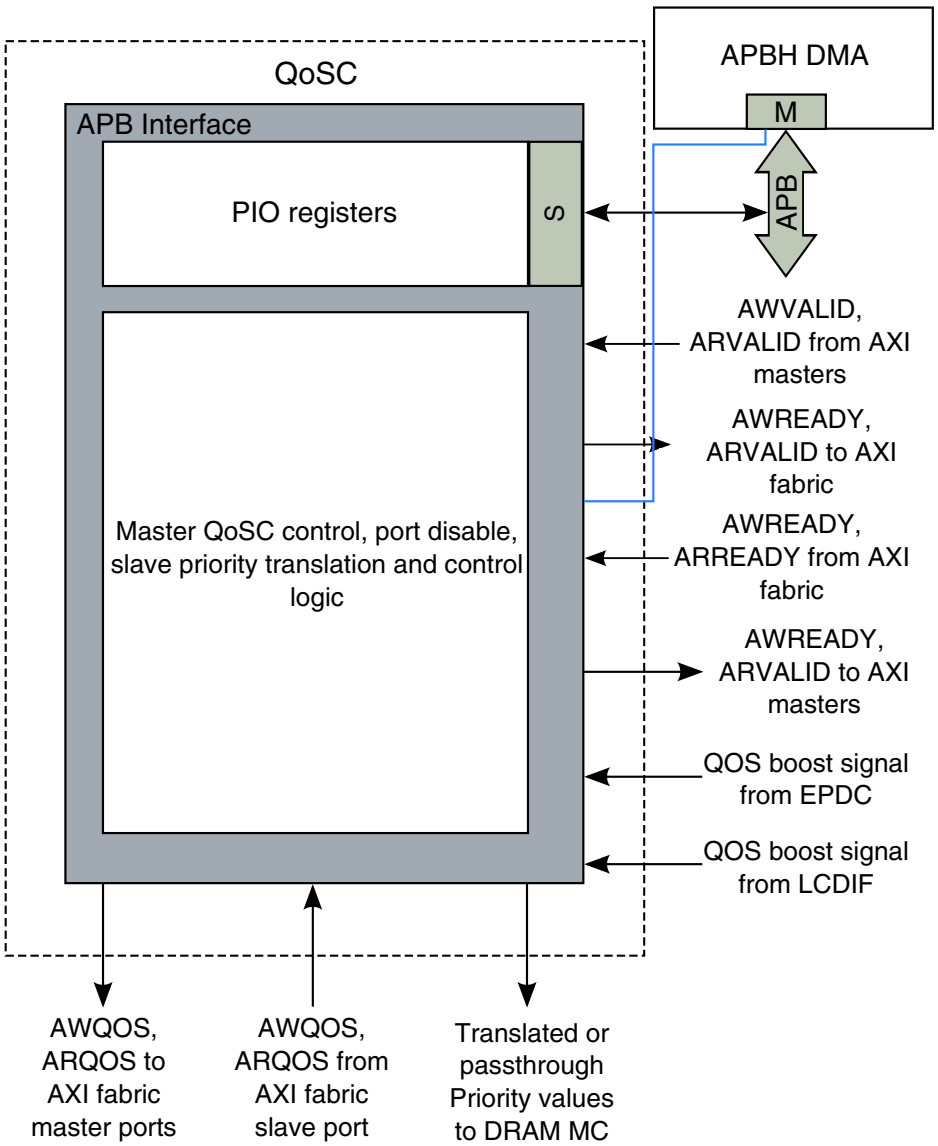


Figure 44-1. QoSC System Level Diagram

Figure 44-2 shows more detail of how the QoSC is connected to the pertinent blocks that surround it. It shows the connections for the AW (Address Write) channel for one AXI master and one AXI slave (the DRAM MC) in the system. This is done for simplicity. The reader should keep in mind that these channel signals are replicated for the AR (Address Read) channel of the master shown. And in turn, these two channels are replicated for each of the 12 i.MX50 masters in the system.

The following sections will reference this figure and explain the interactions between the blocks in the system.

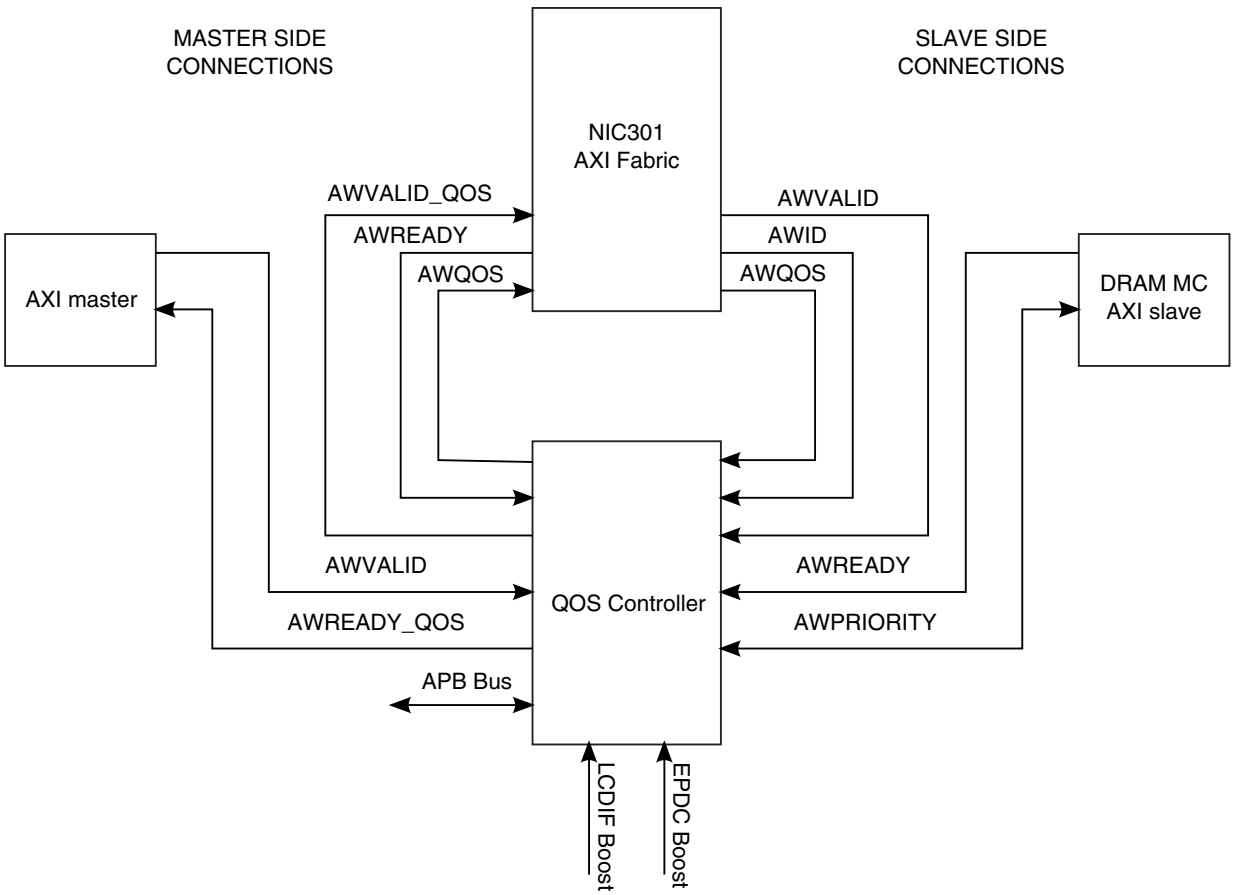


Figure 44-2. QOS Controller

The following table shows the AXI fabric port connection number and the functional block connected to it.

Table 44-1. AXI Fabric port to Functional Block Mapping

Master/Slave Port	Function
M0	ARM
M1_0	SDMA
M1_1	FEC
M1_2	MSHC
M2	DCP
M3	ePXP
M4	USBOH1
M5	GPU2D
M6	BCH
M7	AHB
M8	EPDC

Table continues on the next page...

Table 44-1. AXI Fabric port to Functional Block Mapping (continued)

Master/Slave Port	Function
M9	eLCDIF
S0	TZIC
S1	ROMCP
S2	OCRAM
S3	DRAM MC
S4	EIM
S5	APBH DMA

44.4 APB Slave

The APB interface of the QoSC provides access to programmable registers. Block control and AXI QOS and DRAM MC priority values are set through the interface.

The primary functional register fields are:

- QOS_CTRL[EMI_PRIORITY_MODE] selects whether the AxPRIORITY signals going to the DRAM MC are passed through from the AXI Fabric or the values come from the fields of the QOS_EMI_PRIORITYx registers.
- QOS_CTRL[XLATE_AXI_MODE] selects translate options when in DRAM MC Priority Passthrough mode.
- QOS_CTRL[EPDC_PRIORITY_BOOST] and QOS_CTRL[LCDIF_PRIORITY_BOOST] enable automatic boost of AxQOS and AxPRIORITY values of the EPDC and eLCDIF blocks based on the dynamic bandwidth requirements of those blocks.
- QOS_DISABLE[Mx_DIS] disables a master port from issuing any new transactions to the AXI Fabric.
- QOS_AXI_QOSx and QOS_PRIORITYx registers hold the manually programmed AxQOS and AxPRIORITY values for each master.

NOTE

The naming convention of the AXI Fabric calls the sideband signals that specify transaction priority through the fabric as "qos" signals. The DRAM MC specifies its similar inputs as "priority" signals. For consistency, and to differentiate between the master side and slave side signals, the same convention is followed in this document and in the naming of registers and register fields.

44.5 Master Side Signal Control

As is seen in Figure 44-2, the QoSC supplies the AxQOS sideband signal values. These values come from PIO registers. The register fields are 3 bits and define a range of values from 0-7. Seven being the highest priority and 0 being the lowest priority. For the i.MX50 only 8 relative AxQOS values are implemented even though these sideband signals are 4 bits and can decode priorities of 0-15. It was a design decision that 3 bits would be implemented instead of 4 as this would still provided enough flexibility in the system and would match the bit width of the AxPRIORITY signals on the slave side. Software is responsible for monitoring the system masters performance and setting and adjusting their relative priority based on their individual requirements and system loading. The AR and AW channels are controlled independently on both the master and slave sides of the fabric.

the QoSC is also connected to the AxVALID and AxREADY signals of each AXI master and the AXI fabric instead of the masters and fabric being connected directly. The QoSC needs to monitor these signals so it can safely change the AxQOS signals if software writes a new value to the register. The AXI protocol requires that the AxQOS signals not change during a bus transaction. So to ensure compliance, the QoSC will only allow an AxQOS change under the following conditions:

- On a transaction boundary. The condition of both AxVALID from the master and AxREADY from the fabric are both asserted. This indicates that a transaction has been completed by the fabric and a new transaction can be initiated on the next clock edge.
- When the bus is idle. The AXI channel is considered idle when the AxVALID signal from the master is deasserted. See **[AXI]** for AXI protocol details.

44.6 Slave Side Signal Control

The QoSC monitors all AW and AR channel transactions from the AXI fabric to the DRAM MC slave. For each transaction, the fabric passes the AxQOS bus directly to the QoSC instead of to the DRAM MC. In addition, the QoSC continuously snoops the AxID (master ID), AxVALID and AxREADY signals. (The slave side differs from the master side in that the AxVALID and AxREADY are also directly connected between the fabric and DRAM MC in addition to being connected to the QoSC.) The QoSC uses this information to set the AxPRIORITY signals to the slave for each address channel transaction. It does this according to the HW_QOS_CTRL[EMI_PRIORITY_MODE] register setting. The two modes are are explained below.

44.6.1 Passthrough Priority Mode

In Passthrough Priority mode (QOS_CTRL[EMI_PRIORITY_MODE] = 0) the AxQOS values coming from the AXI fabric are translated and passed on directly to the DRAM MC. They need to be translated first because the AXI fabric and the DRAM MC interpret transaction priorities differently. In the fabric, larger values have higher priority while in the DRAM MC, smaller values signify higher priority. They also differ in the number of bits used. As explained above, although the fabric uses 4 bits for AxQOS values this implementation of QoSC effectively uses 3 bits which matches the DRAM MC. Table 44-2 shows the straightforward translation of the AxQOS bits based on the setting of the QOS_CTRL[XLATE_AXI_MODE] bit.

Table 44-2. QoSC AXI QOS to DRAM MC Priority Translation

AXI Fabrix AxQOS Value	DRAM MC Translated AxPRIORITY value. XLATE_AXI_MODE = 0.	DRAM MC Translated AxPRIORITY value. XLATE_AXI_MODE = 1.
0	7	7
1	6	6
2	5	5
3	4	4
4	3	3
5	2	2
6	1	1
7	1	0

The DRAM MC documentation recommends that the highest priority value, 0, not be used so the system can use it exclusively to dynamically elevate priority to avoid starvation of low priority transactions. The QOS_CTRL[XLATE_AXI_MODE] bit exists, therefore, to provide this flexibility to the programmer.

44.6.2 Manual Priority Mode

In Manual Priority mode (QOS_CTRL[EMI_PRIORITY_MODE] = 1) the AxQOS values coming from the AXI fabric are not used at all. The AxPRIORITY values come from the fields of the QOS_EMI_PRIORITYx registers. The value given for each transaction is the programmed value for that master and transaction type.

44.7 Boost Functionality

QoSC functionality exists to automatically and dynamically boost the qos/priority values for AXI masters 8 (EPDC) and 9 (eLCDIF). This functionality can be enabled by programming the QOS_CTRL[EPDC_PRIORITY_BOOST] and/or QOS_CTRL[LCDIF_PRIORITY_BOOST] register fields.

This function is implemented by monitoring a single level-sensitive boost signal from the EPDC as shown in Figure 2 (identical functionality exists for the LCDIF master). When the boost signal is asserted the QoSC will raise the values of the AxQOS signals on the master side of the fabric. These elevated values will be translated and hence reflected on the slave side in the AxPRIORITY values in Passthrough Priority mode. (Note that AxPRIORITY values will not be boosted in Manual Priority mode as they come from register values directly.) How these values are boosted depends on the programming of the QOS_CTRL[EPDC_PRIORITY_BOOST] field. The qos/priority values can be incremented to the next higher priority or they can be boosted to the highest priority. As soon as the EPDC boost signal is deasserted the values return to their normal programmed values.

44.8 AXI Fabric Master Port Disable

The master ports connected to the AXI fabric can be disabled under software control in the QOS. The QOS_DISABLE register contains disable bits and disable status bits for each master port in the system. When a Mx_DIS disable bit is set, the QOS waits until an AXI transaction boundary or the bus to be idle and then it disables the port. This is accomplished by essentially breaking the AxVALID and AxREADY connections between the master and fabric. Keeping these signals deasserted at the receive end keeps the bus idle by not allowing the initiation of any new transactions. When a port is disabled the corresponding Mx_DIS_STAT bit is set telling software when the bus is actually disabled. It should be noted that this bit does not indicate when any in-flight transactions have finished. System software will have to discover by other means when all pending transaction have finished. The QOS will only hold off the initiation of new ones. Only 11 of the 12 master channels can be disabled. Master 0, which is the ARM processor connection, cannot be disabled through QOS control.

44.9 Programmable Registers

QOS Hardware Register Format Summary

QOS memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_2000	QOS Control Register (QOS_CTRL)	32	R/W	C000_0000h	44.9.1/ 2491
4101_2004	QOS Control Register (QOS_CTRL_SET)	32	R/W	C000_0000h	44.9.1/ 2491
4101_2008	QOS Control Register (QOS_CTRL_CLR)	32	R/W	C000_0000h	44.9.1/ 2491
4101_200C	QOS Control Register (QOS_CTRL_TOG)	32	R/W	C000_0000h	44.9.1/ 2491
4101_2010	AXI QOS Register (QOS_AXI_QOS0)	32	R/W	5555_5555h	44.9.2/ 2493
4101_2014	AXI QOS Register (QOS_AXI_QOS0_SET)	32	R/W	5555_5555h	44.9.2/ 2493
4101_2018	AXI QOS Register (QOS_AXI_QOS0_CLR)	32	R/W	5555_5555h	44.9.2/ 2493
4101_201C	AXI QOS Register (QOS_AXI_QOS0_TOG)	32	R/W	5555_5555h	44.9.2/ 2493
4101_2020	AXI QOS Register (QOS_AXI_QOS1)	32	R/W	5555_5555h	44.9.3/ 2495
4101_2024	AXI QOS Register (QOS_AXI_QOS1_SET)	32	R/W	5555_5555h	44.9.3/ 2495
4101_2028	AXI QOS Register (QOS_AXI_QOS1_CLR)	32	R/W	5555_5555h	44.9.3/ 2495
4101_202C	AXI QOS Register (QOS_AXI_QOS1_TOG)	32	R/W	5555_5555h	44.9.3/ 2495
4101_2030	AXI QOS Register (QOS_AXI_QOS2)	32	R/W	6666_5555h	44.9.4/ 2496
4101_2034	AXI QOS Register (QOS_AXI_QOS2_SET)	32	R/W	6666_5555h	44.9.4/ 2496
4101_2038	AXI QOS Register (QOS_AXI_QOS2_CLR)	32	R/W	6666_5555h	44.9.4/ 2496
4101_203C	AXI QOS Register (QOS_AXI_QOS2_TOG)	32	R/W	6666_5555h	44.9.4/ 2496
4101_2040	EMI priority Registers (QOS_EMI_PRIORITY0)	32	R/W	2222_2222h	44.9.5/ 2498

Table continues on the next page...

QOS memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_2044	EMI priority Registers (QOS_EMI_PRIORITY0_SET)	32	R/W	2222_2222h	44.9.5/ 2498
4101_2048	EMI priority Registers (QOS_EMI_PRIORITY0_CLR)	32	R/W	2222_2222h	44.9.5/ 2498
4101_204C	EMI priority Registers (QOS_EMI_PRIORITY0_TOG)	32	R/W	2222_2222h	44.9.5/ 2498
4101_2050	EMI priority Registers (QOS_EMI_PRIORITY1)	32	R/W	2222_2222h	44.9.6/ 2500
4101_2054	EMI priority Registers (QOS_EMI_PRIORITY1_SET)	32	R/W	2222_2222h	44.9.6/ 2500
4101_2058	EMI priority Registers (QOS_EMI_PRIORITY1_CLR)	32	R/W	2222_2222h	44.9.6/ 2500
4101_205C	EMI priority Registers (QOS_EMI_PRIORITY1_TOG)	32	R/W	2222_2222h	44.9.6/ 2500
4101_2060	EMI priority Registers (QOS_EMI_PRIORITY2)	32	R/W	1111_2222h	44.9.7/ 2502
4101_2064	EMI priority Registers (QOS_EMI_PRIORITY2_SET)	32	R/W	1111_2222h	44.9.7/ 2502
4101_2068	EMI priority Registers (QOS_EMI_PRIORITY2_CLR)	32	R/W	1111_2222h	44.9.7/ 2502
4101_206C	EMI priority Registers (QOS_EMI_PRIORITY2_TOG)	32	R/W	1111_2222h	44.9.7/ 2502
4101_2070	AXI Master Disble Register (QOS_DISABLE)	32	R/W	0000_0000h	44.9.8/ 2504
4101_2074	AXI Master Disble Register (QOS_DISABLE_SET)	32	R/W	0000_0000h	44.9.8/ 2504
4101_2078	AXI Master Disble Register (QOS_DISABLE_CLR)	32	R/W	0000_0000h	44.9.8/ 2504
4101_207C	AXI Master Disble Register (QOS_DISABLE_TOG)	32	R/W	0000_0000h	44.9.8/ 2504
4101_2080	QOS Version Register (QOS_VERSION)	32	R	0100_0000h	44.9.9/ 2506

44.9.1 QOS Control Register (QOS_CTRL*n*)

The Control Register specifies the reset state and the interrupt controls for the Performance Monitor.

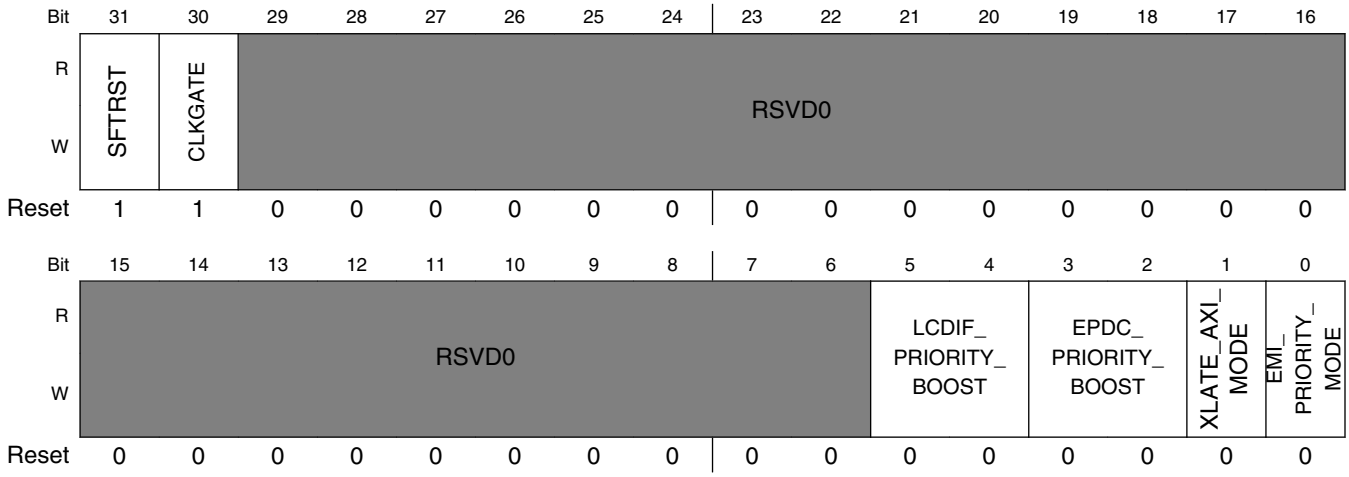
This register controls functions of the QoS module.

Addresses: QOS_CTRL is 4101_2000h base + 0h offset = 4101_2000h

QOS_CTRL_SET is 4101_2000h base + 4h offset = 4101_2004h

QOS_CTRL_CLR is 4101_2000h base + 8h offset = 4101_2008h

QOS_CTRL_TOG is 4101_2000h base + Ch offset = 4101_200Ch



QOS_CTRL*n* field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. 0x0 **RUN** — Allow QoS to operate normally. 0x1 **RESET** — Hold QoS in reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. 0x0 **RUN** — Allow QoS to operate normally. 0x1 **NO_CLKS** — Do not clock QoS gates in order to minimize power consumption.
29–6 RSVD0	Always set this bit field to zero.
5–4 LCDIF_ PRIORITY_ BOOST	Set this field to enable a feedback path that allows the eLCDIF to request higher priority for its accesses when needed. This functionality works in both passthrough and manual priority modes. 0x0 **DISABLE** — Disable priority boost mode for the eLCDIF accesses. 0x1 **INC_PRIORITY** — This setting will raise the access priority level by 1 for the eLCDIF when it requests a priority boost. 0x2 **HIGHEST_PRIORITY** — This setting will raise the access priority to the highest level for the eLCDIF when it requests a priority boost. 0x3 **RESERVED** — Reserved.
3–2 EPDC_ PRIORITY_ BOOST	Set this field to enable a feedback path that allows the EPDC to request higher priority for its accesses when needed. This functionality works in both passthrough and manual priority modes. 0x0 **DISABLE** — Disable priority boost mode for the EPDC accesses. 0x1 **INC_PRIORITY** — This setting will raise the access priority level by 1 for the EPDC when it requests a priority boost. 0x2 **HIGHEST_PRIORITY** — This setting will raise the access priority to the highest level for the EPDC when it requests a priority boost. 0x3 **RESERVED** — Reserved.

Table continues on the next page...

QOS_CTRL*n* field descriptions (continued)

Field	Description
1 XLATE_AXI_ MODE	This bit selects the way in which AXI QoS information is translated into EMI priority values. This bit is only used when the EMI_PRIORITY_MODE bit in this register is set to PASSTHROUGH_PRIORITY. 0x0 **DISALLOW_PRIORITY0** — Never map transaction priorities to level 0 (highest priority). Allow the EMI itself to exclusively use this level to elevate transactions to through aging. 0x1 **ALLOW_PRIORITY0** — Map transactions to all EMI priority levels including level 0 (highest).
0 EMI_PRIORITY_ MODE	Set this bit to one to enable QoS and priority values to be set separately for the AXI fabric and EMI. 0x0 **PASSTHROUGH_PRIORITY** — Translate and pass the AXI fabric QoS settings for each master and command type through to the EMI with each command. 0x1 **MANUAL_PRIORITY** — Pass the register priority settings for each master and command type through to the EMI with each command.

44.9.2 AXI QOS Register (QOS_AXI_QOS0*n*)

This register specifies the AWQOS and ARQOS parameters for masters 0-3.

EXAMPLE

Addresses: QOS_AXI_QOS0 is 4101_2000h base + 10h offset = 4101_2010h

QOS_AXI_QOS0_SET is 4101_2000h base + 14h offset = 4101_2014h

QOS_AXI_QOS0_CLR is 4101_2000h base + 18h offset = 4101_2018h

QOS_AXI_QOS0_TOG is 4101_2000h base + 1Ch offset = 4101_201Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD7	M1_2_ARQOS			RSVD6	M1_2_AWQOS			RSVD5	M1_1_ARQOS			RSVD4	M1_1_AWQOS		
W																
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3	M1_0_ARQOS			RSVD2	M1_0_AWQOS			RSVD1	M0_ARQOS			RSVD0	M0_AWQOS		
W																
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

QOS_AXI_QOS0*n* field descriptions

Field	Description
31 RSVD7	Always set this bit field to zero.
30–28 M1_2_ARQOS	Specifies the QOS level for the read commands on MasterID 3. 15 = highest QOS. 0 = lowest QOS.
27 RSVD6	Always set this bit field to zero.
26–24 M1_2_AWQOS	Specifies the QOS level for the write commands on MasterID 3. 15 = highest QOS. 0 = lowest QOS.
23 RSVD5	Always set this bit field to zero.
22–20 M1_1_ARQOS	Specifies the QOS level for the read commands on MasterID 2. 15 = highest QOS. 0 = lowest QOS.
19 RSVD4	Always set this bit field to zero.
18–16 M1_1_AWQOS	Specifies the QOS level for the write commands on MasterID 2. 15 = highest QOS. 0 = lowest QOS.
15 RSVD3	Always set this bit field to zero.
14–12 M1_0_ARQOS	Specifies the QOS level for the read commands on MasterID 1. 15 = highest QOS. 0 = lowest QOS.
11 RSVD2	Always set this bit field to zero.
10–8 M1_0_AWQOS	Specifies the QOS level for the write commands on MasterID 1. 15 = highest QOS. 0 = lowest QOS.
7 RSVD1	Always set this bit field to zero.
6–4 M0_ARQOS	Specifies the QOS level for the read commands on MasterID 0. 15 = highest QOS. 0 = lowest QOS.
3 RSVD0	Always set this bit field to zero.
2–0 M0_AWQOS	Specifies the QOS level for the write commands on MasterID 0. 15 = highest QOS. 0 = lowest QOS.

44.9.3 AXI QOS Register (QOS_AXI_QOS1*n*)

This register specifies the AWQOS and ARQOS parameters for masters 4-7.

Addresses: QOS_AXI_QOS1 is 4101_2000h base + 20h offset = 4101_2020h

QOS_AXI_QOS1_SET is 4101_2000h base + 24h offset = 4101_2024h

QOS_AXI_QOS1_CLR is 4101_2000h base + 28h offset = 4101_2028h

QOS_AXI_QOS1_TOG is 4101_2000h base + 2Ch offset = 4101_202Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSVD7	M5_ARQOS			RSVD6	M5_AWQOS			RSVD5	M4_ARQOS			RSVD4	M4_AWQOS		
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD3	M3_ARQOS			RSVD2	M3_AWQOS			RSVD1	M2_ARQOS			RSVD0	M2_AWQOS		
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

QOS_AXI_QOS1*n* field descriptions

Field	Description
31 RSVD7	Always set this bit field to zero.
30–28 M5_ARQOS	Specifies the QOS level for the read commands on MasterID 7. 15 = highest QOS. 0 = lowest QOS.
27 RSVD6	Always set this bit field to zero.
26–24 M5_AWQOS	Specifies the QOS level for the write commands on MasterID 7. 15 = highest QOS. 0 = lowest QOS.
23 RSVD5	Always set this bit field to zero.
22–20 M4_ARQOS	Specifies the QOS level for the read commands on MasterID 6. 15 = highest QOS. 0 = lowest QOS.
19 RSVD4	Always set this bit field to zero.
18–16 M4_AWQOS	Specifies the QOS level for the write commands on MasterID 6. 15 = highest QOS. 0 = lowest QOS.

Table continues on the next page...

QOS_AXI_QOS1*n* field descriptions (continued)

Field	Description
15 RSVD3	Always set this bit field to zero.
14–12 M3_ARQOS	Specifies the QOS level for the read commands on MasterID 5. 15 = highest QOS. 0 = lowest QOS.
11 RSVD2	Always set this bit field to zero.
10–8 M3_AWQOS	Specifies the QOS level for the write commands on MasterID 5. 15 = highest QOS. 0 = lowest QOS.
7 RSVD1	Always set this bit field to zero.
6–4 M2_ARQOS	Specifies the QOS level for the read commands on MasterID 4. 15 = highest QOS. 0 = lowest QOS.
3 RSVD0	Always set this bit field to zero.
2–0 M2_AWQOS	Specifies the QOS level for the write commands on MasterID 4. 15 = highest QOS. 0 = lowest QOS.

44.9.4 AXI QOS Register (QOS_AXI_QOS2*n*)

This register specifies the AWQOS and ARQOS parameters for masters 8-11.

Addresses: QOS_AXI_QOS2 is 4101_2000h base + 30h offset = 4101_2030h

QOS_AXI_QOS2_SET is 4101_2000h base + 34h offset = 4101_2034h

QOS_AXI_QOS2_CLR is 4101_2000h base + 38h offset = 4101_2038h

QOS_AXI_QOS2_TOG is 4101_2000h base + 3Ch offset = 4101_203Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RSVD7	M9_ARQOS			RSVD6	M9_AWQOS			RSVD5	M8_ARQOS			RSVD4	M8_AWQOS		
Reset	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RSVD3	M7_ARQOS			RSVD2	M7_AWQOS			RSVD1	M6_ARQOS			RSVD0	M6_AWQOS		
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

QOS_AXI_QOS2*n* field descriptions

Field	Description
31 RSVD7	Always set this bit field to zero.
30–28 M9_ARQOS	Specifies the QOS level for the read commands on MasterID 11. 15 = highest QOS. 0 = lowest QOS.
27 RSVD6	Always set this bit field to zero.
26–24 M9_AWQOS	Specifies the QOS level for the write commands on MasterID 10. 15 = highest QOS. 0 = lowest QOS.
23 RSVD5	Always set this bit field to zero.
22–20 M8_ARQOS	Specifies the QOS level for the read commands on MasterID 10. 15 = highest QOS. 0 = lowest QOS.
19 RSVD4	Always set this bit field to zero.
18–16 M8_AWQOS	Specifies the QOS level for the write commands on MasterID 10. 15 = highest QOS. 0 = lowest QOS.
15 RSVD3	Always set this bit field to zero.
14–12 M7_ARQOS	Specifies the QOS level for the read commands on MasterID 9. 15 = highest QOS. 0 = lowest QOS.
11 RSVD2	Always set this bit field to zero.
10–8 M7_AWQOS	Specifies the QOS level for the write commands on MasterID 9. 15 = highest QOS. 0 = lowest QOS.
7 RSVD1	Always set this bit field to zero.
6–4 M6_ARQOS	Specifies the QOS level for the read commands on MasterID 8. 15 = highest QOS. 0 = lowest QOS.
3 RSVD0	Always set this bit field to zero.
2–0 M6_AWQOS	Specifies the QOS level for the write commands on MasterID 8. 15 = highest QOS. 0 = lowest QOS.

44.9.5 EMI priority Registers (QOS_EMI_PRIORITY0*n*)

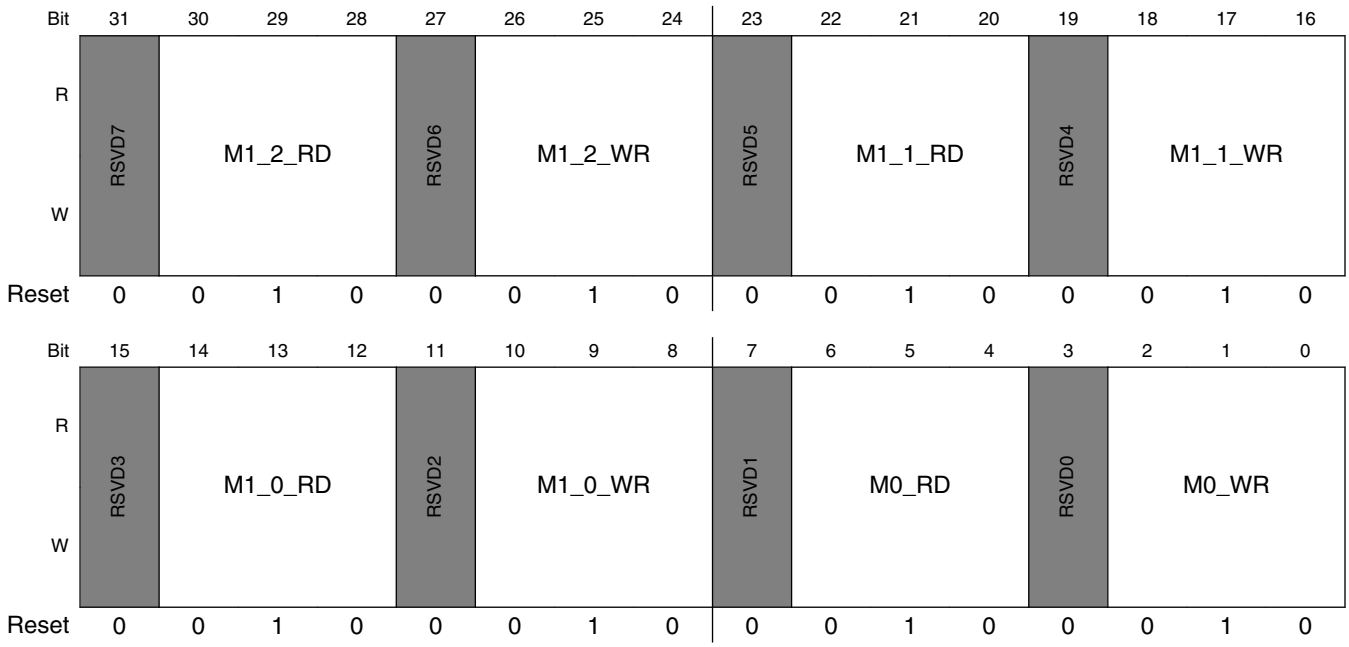
For EMI manual priority mode, this register specifies the AWPRIORITY and ARPRIORITY parameters for masters 0-3.

Addresses: QOS_EMI_PRIORITY0 is 4101_2000h base + 40h offset = 4101_2040h

QOS_EMI_PRIORITY0_SET is 4101_2000h base + 44h offset = 4101_2044h

QOS_EMI_PRIORITY0_CLR is 4101_2000h base + 48h offset = 4101_2048h

QOS_EMI_PRIORITY0_TOG is 4101_2000h base + 4Ch offset = 4101_204Ch



QOS_EMI_PRIORITY0*n* field descriptions

Field	Description
31 RSVD7	Always set this bit field to zero.
30–28 M1_2_RD	Specifies the priority level for the read commands on MasterID 3. 0 = highest priority. 7 = lowest priority.
27 RSVD6	Always set this bit field to zero.
26–24 M1_2_WR	Specifies the priority level for the write commands on MasterID 3. 0 = highest priority. 7 = lowest priority.
23 RSVD5	Always set this bit field to zero.
22–20 M1_1_RD	Specifies the priority level for the read commands on MasterID 2. 0 = highest priority. 7 = lowest priority.
19 RSVD4	Always set this bit field to zero.

Table continues on the next page...

QOS_EMI_PRIORITY0*n* field descriptions (continued)

Field	Description
18–16 M1_1_WR	Specifies the priority level for the write commands on MasterID 2. 0 = highest priority. 7 = lowest priority.
15 RSVD3	Always set this bit field to zero.
14–12 M1_0_RD	Specifies the priority level for the read commands on MasterID 1. 0 = highest priority. 7 = lowest priority.
11 RSVD2	Always set this bit field to zero.
10–8 M1_0_WR	Specifies the priority level for the write commands on MasterID 1. 0 = highest priority. 7 = lowest priority.
7 RSVD1	Always set this bit field to zero.
6–4 M0_RD	Specifies the priority level for the read commands on MasterID 0. 0 = highest priority. 7 = lowest priority.
3 RSVD0	Always set this bit field to zero.
2–0 M0_WR	Specifies the priority level for the write commands on MasterID 0. 0 = highest priority. 7 = lowest priority.

44.9.6 EMI priority Registers (QOS_EMI_PRIORITY1*n*)

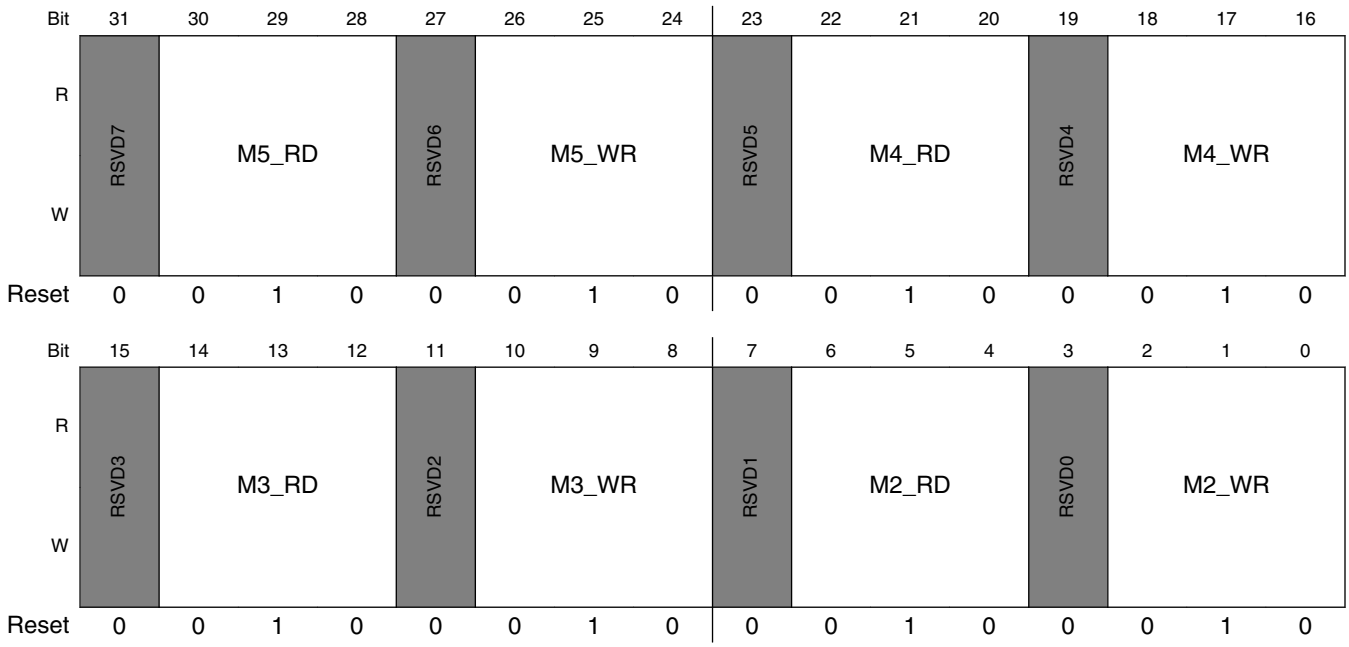
For EMI manual priority mode, this register specifies the AWPRIORITY and ARPRIORITY parameters for masters 4-7.

Addresses: QOS_EMI_PRIORITY1 is 4101_2000h base + 50h offset = 4101_2050h

QOS_EMI_PRIORITY1_SET is 4101_2000h base + 54h offset = 4101_2054h

QOS_EMI_PRIORITY1_CLR is 4101_2000h base + 58h offset = 4101_2058h

QOS_EMI_PRIORITY1_TOG is 4101_2000h base + 5Ch offset = 4101_205Ch



QOS_EMI_PRIORITY1*n* field descriptions

Field	Description
31 RSVD7	Always set this bit field to zero.
30–28 M5_RD	Specifies the priority level for the read commands on MasterID 7. 0 = highest priority. 7 = lowest priority.
27 RSVD6	Always set this bit field to zero.
26–24 M5_WR	Specifies the priority level for the write commands on MasterID 7. 0 = highest priority. 7 = lowest priority.
23 RSVD5	Always set this bit field to zero.
22–20 M4_RD	Specifies the priority level for the read commands on MasterID 6. 0 = highest priority. 7 = lowest priority.
19 RSVD4	Always set this bit field to zero.

Table continues on the next page...

QOS_EMI_PRIORITY1*n* field descriptions (continued)

Field	Description
18–16 M4_WR	Specifies the priority level for the write commands on MasterID 6. 0 = highest priority. 7 = lowest priority.
15 RSVD3	Always set this bit field to zero.
14–12 M3_RD	Specifies the priority level for the read commands on MasterID 5. 0 = highest priority. 7 = lowest priority.
11 RSVD2	Always set this bit field to zero.
10–8 M3_WR	Specifies the priority level for the write commands on MasterID 5. 0 = highest priority. 7 = lowest priority.
7 RSVD1	Always set this bit field to zero.
6–4 M2_RD	Specifies the priority level for the read commands on MasterID 4. 0 = highest priority. 7 = lowest priority.
3 RSVD0	Always set this bit field to zero.
2–0 M2_WR	Specifies the priority level for the write commands on MasterID 4. 0 = highest priority. 7 = lowest priority.

44.9.7 EMI priority Registers (QOS_EMI_PRIORITY2*n*)

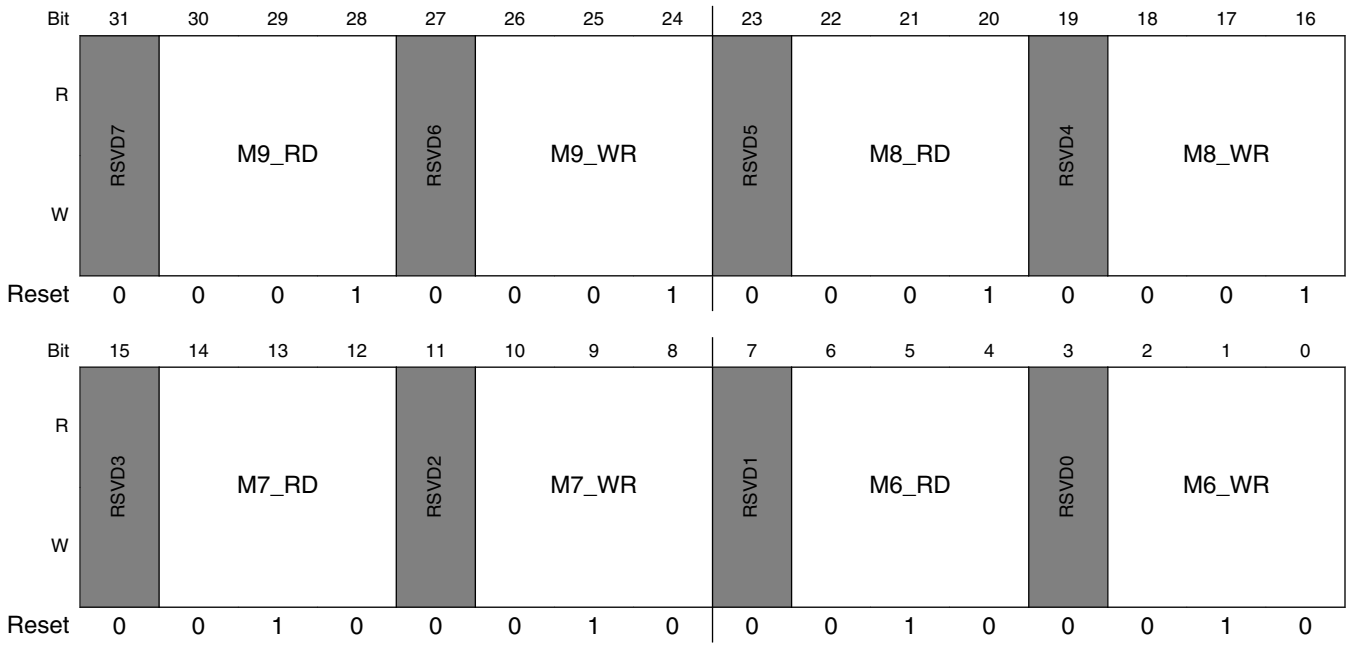
For EMI manual priority mode, this register specifies the AWPRIORITY and ARPRIORITY parameters for masters 8-11.

Addresses: QOS_EMI_PRIORITY2 is 4101_2000h base + 60h offset = 4101_2060h

QOS_EMI_PRIORITY2_SET is 4101_2000h base + 64h offset = 4101_2064h

QOS_EMI_PRIORITY2_CLR is 4101_2000h base + 68h offset = 4101_2068h

QOS_EMI_PRIORITY2_TOG is 4101_2000h base + 6Ch offset = 4101_206Ch



QOS_EMI_PRIORITY2*n* field descriptions

Field	Description
31 RSVD7	Always set this bit field to zero.
30–28 M9_RD	Specifies the priority level for the read commands on MasterID 11. 0 = highest priority. 7 = lowest priority.
27 RSVD6	Always set this bit field to zero.
26–24 M9_WR	Specifies the priority level for the write commands on MasterID 10. 0 = highest priority. 7 = lowest priority.
23 RSVD5	Always set this bit field to zero.
22–20 M8_RD	Specifies the priority level for the read commands on MasterID 10. 0 = highest priority. 7 = lowest priority.
19 RSVD4	Always set this bit field to zero.

Table continues on the next page...

QOS_EMI_PRIORITY2*n* field descriptions (continued)

Field	Description
18–16 M8_WR	Specifies the priority level for the write commands on MasterID 10. 0 = highest priority. 7 = lowest priority.
15 RSVD3	Always set this bit field to zero.
14–12 M7_RD	Specifies the priority level for the read commands on MasterID 9. 0 = highest priority. 7 = lowest priority.
11 RSVD2	Always set this bit field to zero.
10–8 M7_WR	Specifies the priority level for the write commands on MasterID 9. 0 = highest priority. 7 = lowest priority.
7 RSVD1	Always set this bit field to zero.
6–4 M6_RD	Specifies the priority level for the read commands on MasterID 8. 0 = highest priority. 7 = lowest priority.
3 RSVD0	Always set this bit field to zero.
2–0 M6_WR	Specifies the priority level for the write commands on MasterID 8. 0 = highest priority. 7 = lowest priority.

44.9.8 AXI Master Disble Register (QOS_DISABLE*n*)

This register allows disabling of the individual master ports of the AXI fabric. This can be useful during system clock frequency programming.

Addresses: QOS_DISABLE is 4101_2000h base + 70h offset = 4101_2070h

QOS_DISABLE_SET is 4101_2000h base + 74h offset = 4101_2074h

QOS_DISABLE_CLR is 4101_2000h base + 78h offset = 4101_2078h

QOS_DISABLE_TOG is 4101_2000h base + 7Ch offset = 4101_207Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2				M9_DIS_STAT	M8_DIS_STAT	M7_DIS_STAT	M6_DIS_STAT	M5_DIS_STAT	M4_DIS_STAT	M3_DIS_STAT	M2_DIS_STAT	M1_2_DIS_STAT	M1_1_DIS_STAT	M1_0_DIS_STAT	RSVD1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1				M9_DIS	M8_DIS	M7_DIS	M6_DIS	M5_DIS	M4_DIS	M3_DIS	M2_DIS	M1_2_DIS	M1_1_DIS	M1_0_DIS	RSVD0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

QOS_DISABLE*n* field descriptions

Field	Description
31–28 RSVD2	Always set this bit field to zero.
27 M9_DIS_STAT	This bit indicates the enable/disable status for master port0 on the AXI fabric. 0 = enabled; 1 = disabled.
26 M8_DIS_STAT	This bit indicates the enable/disable status for master port0 on the AXI fabric. 0 = enabled; 1 = disabled.
25 M7_DIS_STAT	This bit indicates the enable/disable status for master port0 on the AXI fabric. 0 = enabled; 1 = disabled.
24 M6_DIS_STAT	This bit indicates the enable/disable status for master port0 on the AXI fabric. 0 = enabled; 1 = disabled.
23 M5_DIS_STAT	This bit indicates the enable/disable status for master port0 on the AXI fabric. 0 = enabled; 1 = disabled.
22 M4_DIS_STAT	This bit indicates the enable/disable status for master port0 on the AXI fabric. 0 = enabled; 1 = disabled.

Table continues on the next page...

QOS_DISABLE*n* field descriptions (continued)

Field	Description
21 M3_DIS_STAT	This bit indicates the enable/disable status for master port0 on the AXI fabric. 0 = enabled; 1 = disabled.
20 M2_DIS_STAT	This bit indicates the enable/disable status for master port0 on the AXI fabric. 0 = enabled; 1 = disabled.
19 M1_2_DIS_ STAT	This bit indicates the enable/disable status for master port1_2 on the AXI fabric. 0 = enabled; 1 = disabled.
18 M1_1_DIS_ STAT	This bit indicates the enable/disable status for master port1_1 on the AXI fabric. 0 = enabled; 1 = disabled.
17 M1_0_DIS_ STAT	This bit indicates the enable/disable status for master port1_0 on the AXI fabric. 0 = enabled; 1 = disabled.
16–12 RSVD1	Always set this bit field to zero.
11 M9_DIS	Setting this bit disables the interface between master9 and the AXI fabric.
10 M8_DIS	Setting this bit disables the interface between master8 and the AXI fabric.
9 M7_DIS	Setting this bit disables the interface between master7 and the AXI fabric.
8 M6_DIS	Setting this bit disables the interface between master6 and the AXI fabric.
7 M5_DIS	Setting this bit disables the interface between master5 and the AXI fabric.
6 M4_DIS	Setting this bit disables the interface between master4 and the AXI fabric.
5 M3_DIS	Setting this bit disables the interface between master3 and the AXI fabric.
4 M2_DIS	Setting this bit disables the interface between master2 and the AXI fabric.
3 M1_2_DIS	Setting this bit disables the interface between master1_2 and the AXI fabric.
2 M1_1_DIS	Setting this bit disables the interface between master1_1 and the AXI fabric.
1 M1_0_DIS	Setting this bit disables the interface between master1_0 and the AXI fabric.
0 RSVD0	Always set this bit field to zero.

44.9.9 QOS Version Register (QOS_VERSION)

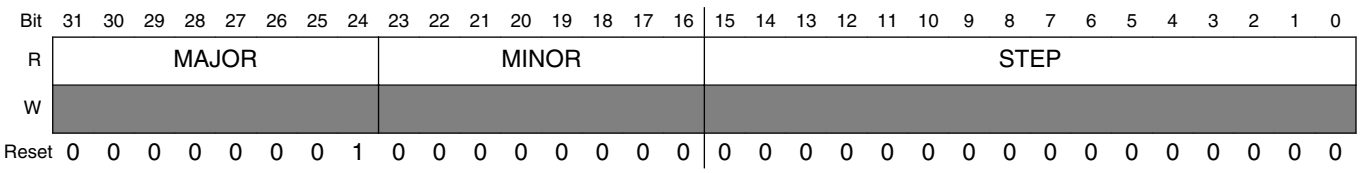
This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
if (QOS_VERSION.B.MAJOR != 1) Error();
```

Address: QOS_VERSION is 4101_2000h base + 80h offset = 4101_2080h



QOS_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 45
Read Only Memory Controller Patch (ROMCP)

45.1 Introduction

45.1.1 Overview

The Read Only Memory Controller module with ROM Patch (ROMCP) acts as an interface between the ARM advanced high-performance bus (AHB - Lite) and the Read Only Memory. The ROMCP module consists of a ROM Controller and a ROM Patch. The ROM Patch module is used to either patch code routines or fix data tables in the ROM area. Figure 45-1 depicts the main functional blocks of the ROMCP.

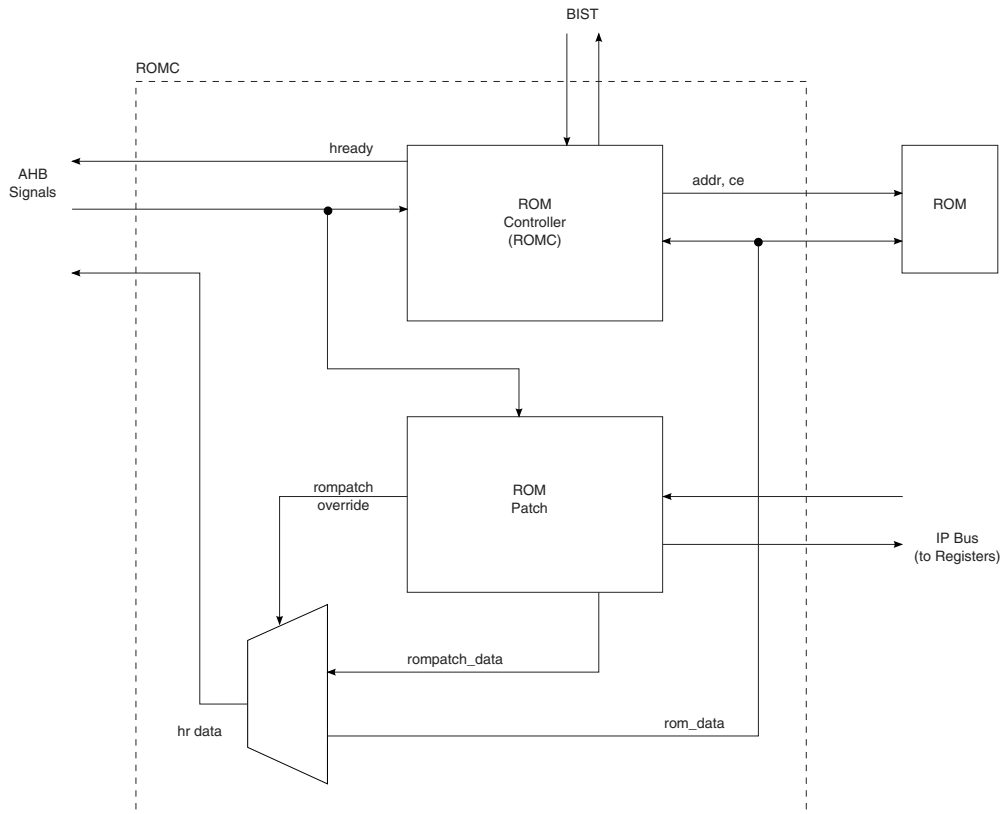


Figure 45-1. ROMCP Block Diagram

45.1.2 Features

The features are as follows:

- Supports ROM size ranges from 16 Kbyte up to 4 Mbyte with increments of 1 Kbyte
- Supports opcode patching for a maximum of 16 different addresses in 4 Mbytes of ROM space
- Supports one-word data fixes for a max of 8 memory locations in 4 Mbytes of ROM space
- Supports patching of the Reset Vector (at 0x0000_0000) to allow external booting

45.1.3 Modes of Operation

There are two modes of operation: normal mode and BIST mode. In normal mode the ROMCP ensures correct reads from the ROM, assuming the memory complies with the characteristics and requirements for which the ROMCP was designed. The ROMCP also performs hole decoding, aborting accesses into memory holes and converting the non-continuous ROM memory map into a continuous one for physical memory accesses.

45.1.3.1 Normal Operating Modes

This subsection should describe all the functional operation modes of the IP block.

45.1.3.2 Low Power Modes

There are two clock enables that are used to switch off parts of the ROMCP logic when inactive. The first clock enable is used to disable the ROM Controller when the master connected to the AHB interface is not initiating a read to the ROM. The second clock enable is used to disable the registers used to program the ROM patch feature when the registers are not being accessed.

45.2 Functional Description

This section is divided up into the ROM Controller Functional Description and the ROMPATCH functional description.

45.2.1 ROM Controller (ROMC) Functional Description

45.2.1.1 Functionality Overview

The ROMC serves two main functions. First, as an interface between the AHB-Lite bus on an ARM platform and the ROM. Second, it drives and receives several signals for the BIST engine. In normal mode of operation, the ROMC monitors the AHB-Lite for memory access requests and performs the memory operation to the ROM.

The ROMC includes the option to wait state all accesses from either the ARM or non-ARM masters to ROM in the event that timing requirements will not allow single hclk clock cycle reads. If a wait state is required, the static inputs rom_wait_arm11 or

rom_wait_alt_mstr can be set to 1 and accesses will take two hclk clock cycles. If wait states are not required, rom_wait_arm11 or rom_wait_alt_mstr can be set to 0 and accesses will take one hclk clock cycle to complete.

45.2.1.2 ROMC Architecture Diagram

A simple block diagram illustrating the internal architecture of the ROMC is shown in Figure 45-2. The basic logic blocks of the ROMC include an AHB-Lite interface, a ROM memory interface, a BIST block and a block for muxing between BIST mode and normal mode.

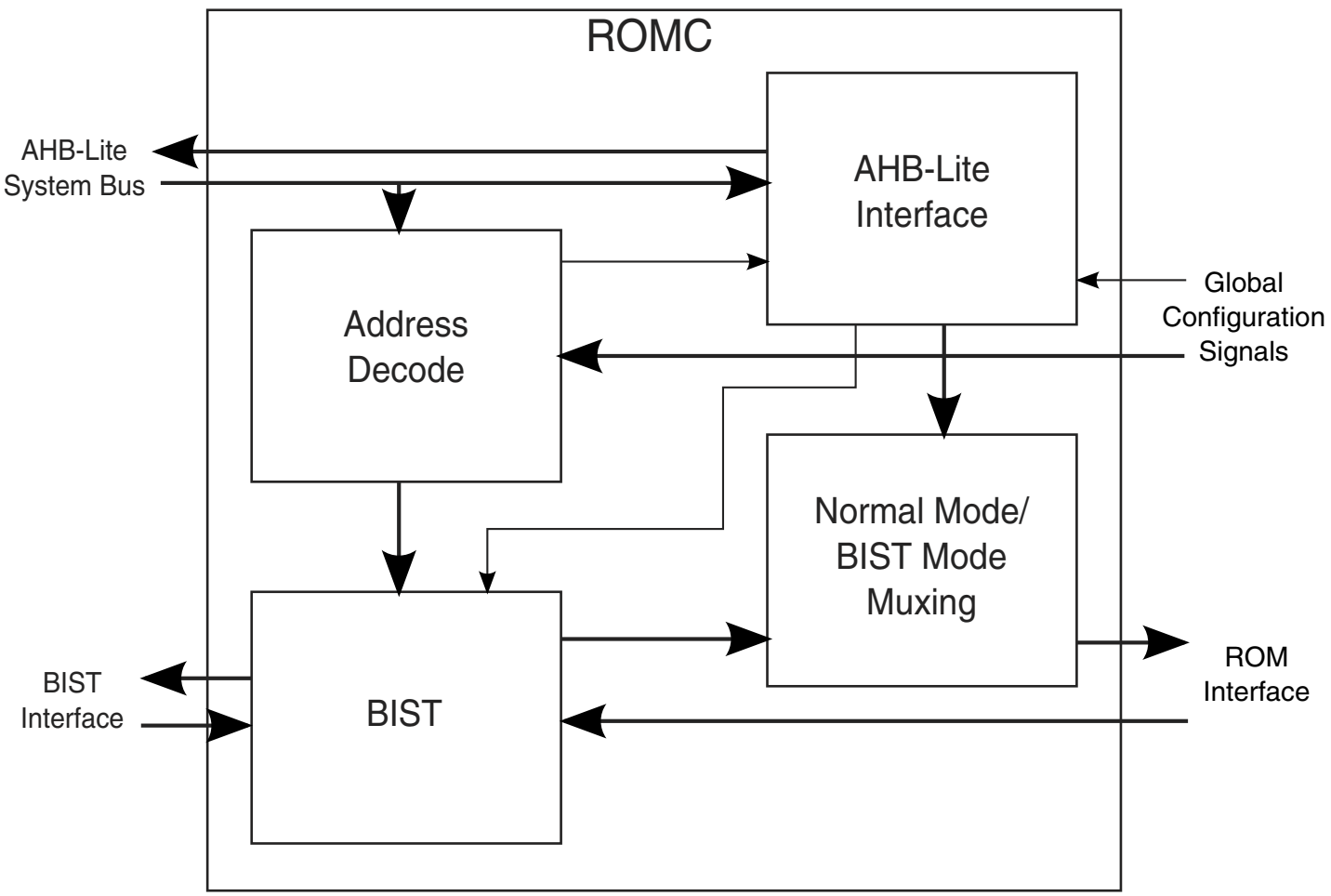


Figure 45-2. ROMC Internal Architecture

45.2.1.3 AHB-Lite Interface

The ROMC output signals on the AHB-Lite Interface indicate the timing of the response to any access (wait-stated or not) and any error conditions (writing to ROM or access to a valid ROM hole).

45.2.2 ROMPATCH Functional Description

45.2.2.1 ROMPATCH Disabling

All the bits in the ROMPATCHENL register are cleared on Reset, disabling all the address comparators. Once the comparators have been enabled, the ROMPATCH functions of data fixing and opcode patching can be quickly disabled by setting the DIS bit in the ROMPATCHCNTL register. This bit is used to enable secure operations in which patching functions need to be disabled. This bit is cleared on Reset.

45.2.2.2 ROMPATCH Event Priority

The ROMPATCH has a total of 16 address comparators. The first 8 (0 through 7) comparators can be programmed for the data fixing function (via the 8 data fix enable bits in the ROMPATCHCNTL register) while the rest are for opcode patching by default. This allows for potential multiple matching events involving both data fixing and opcode patch types. In these cases the ROMPATCH module assigns the highest priority to a data fixing event. For example, if the ROMPATCH is set up to data fix a certain address with comparator 4 and also opcode patch the same address with comparator 7, it will let comparator 4 have higher priority in indicating a match, and data from ROMPATCHD4 will be returned as the orverride value.

If multiple address matches of the same type level occur concurrently, then the ROMPATCH will choose the source with the highest source number. For example, the ROMPATCH is setup to data fix the same location with address comparators 4 and 7, then address comparator 7 will have higher priority in indicating a match, and the value from ROMPATCHD7 will be returned as the orverride value. The same priority applies for an opcode patch event, except the override data is in the form of an SWI instruction with the comment field set to the source number with the highest priority.

45.2.2.3 Data Fixing

The data fixing feature allows ROM data to be updated by direct replacement when it's being read. This data usually is from data tables but can include ARM instructions. To enable data fixing on a certain address, this address value is written into one of the first eight (0 through 7) of ROMPATCHAxx registers and the same numbered bit set in the ROMPATCHENL and ROMPATCHCNTL registers. The data to be used for replacement is placed in the corresponding ROMPATCHDxx. The ROMPATCH module looks for a read access to ROM (either code fetch or data load) by snooping the AHB interface for read transactions. The address is compared with the values stored in the ROMPATCHAxx[22:2] registers. If a match occurs from one of the comparators, the ROMPATCH places the value in the corresponding ROMPATCHDxx register on the read data bus by overriding the read data coming from the actual ROM (see the mux in Figure 45-1). In data fixing, the entire word is replaced so if a byte or half-word access occurs on a "data fix" location, the entire data word is replaced. The word being replaced is word aligned (The two LSBs of the matching ROMPATCHAxx are ignored in the data fix operation).

45.2.2.4 Opcode Patching

The opcode patch feature provides a mechanism to fetch updated versions of code routines that were originally programmed in ROM. This patching mechanism makes use of the SWI (software interrupt instruction) and a table of function pointers residing in writable memory. The opcode being patched is replaced with a SWI instruction by the ROMPATCH. Subsequent processing of the SWI reads from a function pointers table to obtain the address of the replacement code. Execution resumes with this code patch.

To enable opcode patching of a certain address, this address value is written into one of the ROMPATCHAxx registers and the corresponding bit set in the ROMPATCHENL to enable the associated comparator. The register's LSB (ROMPATCHxx[0]) should be set if THUMB mode patching is in effect for this address. The ROMPATCH module identifies a ROM read access by snooping the AHB interface. The address is compared with the values stored in the ROMPATCHAxx[22:2] registers. If a match occurs from one of the comparators, the ROMPATCH generates the opcode of a software interrupt (SWI) instruction with the comment field containing the number of the matching address comparator. This opcode and comment are placed on the read data bus

The type of SWI generated, either ARM or THUMB, is determined by the LSB of the ROMPATCHAxx register associated with the opcode patch. This bit is cleared for ARM mode (32 bits). The ROMPATCH generates a 32-bit SWI (opcode field is 0xEF, occupying bits [31:24] of the word), with the least significant 5 bits of the 24-bit comment field (bits [23:0]) containing the number of the matching address comparator.

The rest of the comment field is filled with zeros. This means that the ROMPATCH will use 16 of the 16777216 possible software interrupts. The ROMPATCH overrides the read data from the ROM.

If the LSB of the matching ROMPATCHAxx register is set, the opcode patch is in THUMB mode (16 bits or half word). The ROMPATCH generates a 16-bit SWI instruction (opcode field is 0xDF, occupying bits [15:8] of the half word) with the least significant 5 bits of the 8-bit comment field containing the source number of the address comparator. The rest of the comments field is filled with zeros. This means that the ROMPATCH will use 16 of the 256 possible software interrupts. The ROMPATCH puts this 16 bit SWI instruction value on the proper half of the rompatch_romc_hrdata bus. The other half is zeroed out. Which half of the bus contains the SWI opcode and comment depends on the mode (big endian or little endian) and the bit 1 of the matching ROMPATCHAxx register. In little endian mode, the lower half is bits {15:0] and the upper half is bits {31:16]. The order is reversed in Big Endian mode.

In little endian mode , if bit 1 of the matching ROMPATCHAxx is cleared (lower half word selected) then the SWI instruction is put on the lower 16 bits of the read data bus and the upper 16 bits are zeroed out. Only the lower 16 bits of the read data bus is overwritten by the rompatch data. If ROMPATCHAxx[1] is set (upper half word selected), the SWI instruction is put on the upper 16 bits of the read data bus and the lower 16 bits are zeroed out. Only the upper 16 bits of the read data bus is overwritten.

In big endian mode , if bit 1 of the matching ROMPATCHAxx is cleared (lower half word selected) then the SWI instruction is put on the upper 16 bits of the read data bus while the lower 16 bits are zeroed out. Only the upper 16 bits of the read data bus is overwritten. If ROMPATCHAxx[1] is set (upper word selected), the SWI instruction is put on the lower 16 bits and the upper 16 bits are zeroed out. Only the lower 16 bits of the read data bus is overwritten.

The eventual execution of the SWI causes the ARM to save the CPSR in SPSR_SVC, the address of the next instruction after the SWI in R14_SVC, enter Supervisor mode, and fetch the SWI vector at 0x8, which then takes it to a handler for further processing as described in the next section.

45.2.2.4.1 Typical Software Response to Opcode Patch

When the SWI handler executes it needs to determine whether the SWI was generated by the ROMPATCH. This is done by loading the SWI instruction and extracting its comment field. The state of the ARM Cortex-A8(ARM or THUMB) when the SWI was executed dictates whether to load the instruction word (ARM) or half word (THUMB). This state information can be determined by testing the T bit (bit 5) of the SPSR. If it's set, the execution was in THUMB mode.

By convention, if the comment field of the SWI is greater than 16, the software interrupt was initiated by software (i.e. an operating system call), and a branch is taken to the appropriate handler routine for further processing. If the comment field is less than 16, the SWI was generated by the ROMPATCH module performing a code patch operation. In this case, the software then reads from a table of function pointers, using the value in the SWI comment field as the index into the table. The value that is read is the address of the code patch. This value is loaded into the PC to begin the execution of the code patch. The following code segment illustrates a typical handling of the SWI.

```
stmfd        sp!, {r0-r1,lr}       @ push register onto SWI stack
mrs         r0, spsr        @ get saved status register
tst          r0, #0x20     @ check if call was in THUMB mode
ldrneh      r0, [lr,#-2]       @ yes: load opcode half-word and
bicne       r0, r0, #0xff00   @ yes: extract THUMB comment
ldreq       r0, [lr,#-4]       @ no: load opcode word and
biceq       r0, r0, #0xff000000 @ no: extract ARM comment
                    @ now r0 has comment field
cmp         r0, #16      @ compare to 16 (maximum for ROMPATCH)
ldrlt          lr, =rompatch_tbl_ptr @ < 16: get top of current ROMPATCH
                    @ table; global variable which is
                    @ changeable per context
ldrlt         r1, [lr, r0, lsl #2]   @ < 16: read function pointer from
                    @ table assumed an array of pointers
                    @ patch functions
strlt          r1, [sp, #8]     @ < 16: store function pointer onto
                    @ stack in position of link register
ldmltfd       sp!, {r0-r1,pc}^      @ < 16: "fake" return from SWI, will
                    @ vector core to appropriate patch
                    @ function and set core back to previous
                    @ mode of operating
ldr          r1, =swi_hdler   @ >= 16: pointer to standard SWI
                    @ handler
mov         lr, pc         @ >= 16: set link register
bx          r1          @ >= 16: jump to standard SWI
                    @ handler
ldmfd        sp!, {r0-r1,pc}^      @ >= 16: pop registers from stack
```

45.2.2.5 Alternate Masters and ROMPATCH

The ROMPATCH module sits on the AHB bus of the internal ROM (ROMC). This means that the ROMPATCH can modify values on the read data bus going to the master. Therefore, any master which reads an opcode patched or data patched location will read patched data.

45.3 Programmable Registers

The following figure shows a sample memory map.

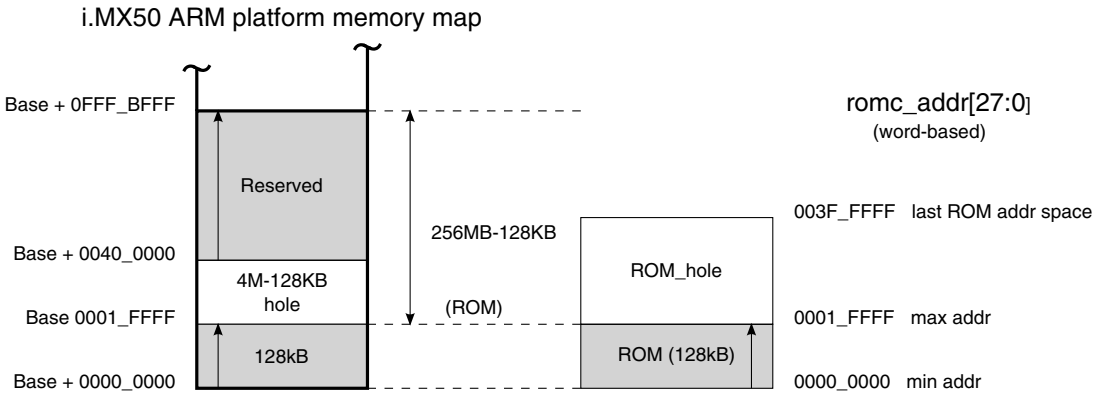


Figure 45-3. Memory Map-Example 128 K ROM Connected

The ROMCP supports ROM sizes with a range of 16 Kbyte to 4 Mbyte with an increment of 1 Kbyte. The 16 Kbyte lower limit was chosen because the minimum size of security code on an ARM platform is approximately 16 Kbyte of code, which is only accessible in supervisor mode. Note that it is the MMU that controls whether any region of memory is secure. For the i.MX50, ROMCP is designed to support 96 Kbyte ROM, and address decoding from continuous 0x0000_0000~0x0001_7FFF address space. Other addressess in 4 Mbyte range are reserved.

The exception vectors must be secured as well, and must be put in the same area as the security code. Since they must reside at address 0x0000_0000, the entire 16 Kbyte of ROM which can only be accessible in supervisor mode is located at the very beginning of the platform memory map.

If the user chooses not to use the security code, a memory size smaller than 16 Kbyte can be connected to the platform (minimum of 1 Kbyte). The MMU can be programmed to allow any kind of access into this memory.

The reason for the big memory hole is because the ROMCP is designed to be as flexible as possible to comply with demands of different operating systems. For example, the requirement of Windows CE OS is to be able to manage memory regions through the MMU page tables (to control accesses to privileged, cacheable, or bufferable areas) which requires regions to be separated into 1 Mbyte page tables. Since it is preferrable to offer the flexibility of having the first 16 Kbyte as a secure region with vectors and security code, but allowing the rest of the ROM to be user-mode accessible, two separate 1MB page tables are used.

To start a potentially unsecure region directly on a 1 MB boundary (starting on address 0010_0000) requires additional address translation within the ROMCP. That causes an extra delay in the address path through the complicated shifting logic in an already critical design timing path. To prevent the extra delay, the first bit above the maximum address of the maximum ROM size that can be connected to the platform is used to select

the region. After entering the ROMCP this bit is only used in the ROM hole decode, and is not passed to the ROM on the address bus. Therefore, no translation is necessary and the address timing path is unaffected by the mapping scheme.

The maximum size of ROM connectable to the ROMCP is 4 Mbyte. To address it the address bits haddr[21:2] are needed. The address range from 128KB to 4 Mbyte is the ROM hole in the i.MX50, and upper addresses above 4 Mbyte in the ROM region are reserved. It also implies that if an address outside the given regions is accessed, the ROMCP will respond with an error response.

ROMCP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_80D4	ROMPATCH Data Registers (ROMCP_ROMPATCHD7)	32	R/W	0000_0000h	45.3.1/ 2517
63FB_80D8	ROMPATCH Data Registers (ROMCP_ROMPATCHD6)	32	R/W	0000_0000h	45.3.1/ 2517
63FB_80DC	ROMPATCH Data Registers (ROMCP_ROMPATCHD5)	32	R/W	0000_0000h	45.3.1/ 2517
63FB_80E0	ROMPATCH Data Registers (ROMCP_ROMPATCHD4)	32	R/W	0000_0000h	45.3.1/ 2517
63FB_80E4	ROMPATCH Data Registers (ROMCP_ROMPATCHD3)	32	R/W	0000_0000h	45.3.1/ 2517
63FB_80E8	ROMPATCH Data Registers (ROMCP_ROMPATCHD2)	32	R/W	0000_0000h	45.3.1/ 2517
63FB_80EC	ROMPATCH Data Registers (ROMCP_ROMPATCHD1)	32	R/W	0000_0000h	45.3.1/ 2517
63FB_80F0	ROMPATCH Data Registers (ROMCP_ROMPATCHD0)	32	R/W	0000_0000h	45.3.1/ 2517
63FB_80F4	ROMPATCH Control Register (ROMCP_ROMPATCHCNTL)	32	R/W	0840_0000h	45.3.2/ 2518
63FB_80FC	ROMPATCH Enable Registers Low (ROMCP_ROMPATCHENL)	32	R/W	0000_0000h	45.3.3/ 2519
63FB_8100	ROMPATCH Address Registers (ROMCP_ROMPATCHA0)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8104	ROMPATCH Address Registers (ROMCP_ROMPATCHA1)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8108	ROMPATCH Address Registers (ROMCP_ROMPATCHA2)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_810C	ROMPATCH Address Registers (ROMCP_ROMPATCHA3)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8110	ROMPATCH Address Registers (ROMCP_ROMPATCHA4)	32	R/W	0000_0000h	45.3.4/ 2520

Table continues on the next page...

ROMCP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_8114	ROMPATCH Address Registers (ROMCP_ROMPATCHA5)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8118	ROMPATCH Address Registers (ROMCP_ROMPATCHA6)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_811C	ROMPATCH Address Registers (ROMCP_ROMPATCHA7)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8120	ROMPATCH Address Registers (ROMCP_ROMPATCHA8)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8124	ROMPATCH Address Registers (ROMCP_ROMPATCHA9)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8128	ROMPATCH Address Registers (ROMCP_ROMPATCHA10)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_812C	ROMPATCH Address Registers (ROMCP_ROMPATCHA11)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8130	ROMPATCH Address Registers (ROMCP_ROMPATCHA12)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8134	ROMPATCH Address Registers (ROMCP_ROMPATCHA13)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8138	ROMPATCH Address Registers (ROMCP_ROMPATCHA14)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_813C	ROMPATCH Address Registers (ROMCP_ROMPATCHA15)	32	R/W	0000_0000h	45.3.4/ 2520
63FB_8208	ROMPATCH Status Register (ROMCP_ROMPATCHSR)	32	R/W	0000_0000h	45.3.5/ 2521

45.3.1 ROMPATCH Data Registers (ROMCP_ROMPATCHD*n*)

The ROMPATCH data registers (ROMPATCHD7 through ROMPATCHD0) store the data to use for the 8 1-word data fix events. Each register is associated with an address comparator (7 through 0). When a data fixing event occurs, the value in the data register corresponding to the comparator that has the address match is returned as read data to the ARM core.

If more than one address comparators match, the highest-numbered one takes precedence, and the value in corresponding data register is used for the patching event.

Addresses: ROMCP_ROMPATCHD7 is 63FB_8000h base + D4h offset = 63FB_80D4h
ROMCP_ROMPATCHD6 is 63FB_8000h base + D8h offset = 63FB_80D8h
ROMCP_ROMPATCHD5 is 63FB_8000h base + DCh offset = 63FB_80DCh
ROMCP_ROMPATCHD4 is 63FB_8000h base + E0h offset = 63FB_80E0h
ROMCP_ROMPATCHD3 is 63FB_8000h base + E4h offset = 63FB_80E4h
ROMCP_ROMPATCHD2 is 63FB_8000h base + E8h offset = 63FB_80E8h
ROMCP_ROMPATCHD1 is 63FB_8000h base + ECh offset = 63FB_80ECh
ROMCP_ROMPATCHD0 is 63FB_8000h base + F0h offset = 63FB_80F0h

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

ROMCP_ROMPATCHD*n* field descriptions

Field	Description
31–0 DATAX	Data Fix Registers. Stores the data used for 1-word data fix operations. The values stored within these registers do not affect the writes to the memory system. They are selected over the read data from ROM when a data fix event occurs. If any part of the 1-word data fix is read, then the entire word is replaced. Therefore, a byte or half-word read will cause the ROMPATCH to replace the entire word. The word is word address aligned.

45.3.2 ROMPATCH Control Register (ROMCP_ROMPATCHCNTL)

The ROMPATCH control register (ROMPATCHCNTL) contains the module disable bit and the data fix enable bits. The module disable bit provides a means to disable the ROMPATCH data fix and opcode patching functions, even when the address comparators are enabled. The External Boot feature is not affected by this bit. The eight data fix enable bits (0 through 7), when set, assign the associated address comparators to data fix operations

NOTE

Bits 27 and 22 always read as 1s.

Address: ROMCP_ROMPATCHCNTL is 63FB_8000h base + F4h offset = 63FB_80F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-		DIS	-												
Reset	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								DATAFIX							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMCP_ROMPATCHCNTL field descriptions

Field	Description
31–30 -	Reserved
29 DIS	ROMPATCH Disable. This bit, when set, disables all ROMPATCH operations. This bit is used to enable secure operations. 0 Does not affect any ROMPATCH functions (default) 1 Disable all ROMPATCH functions: data fixing, and opcode patching
28–8 -	Reserved N/A - Bits 27 and 22 always read as 1's
7–0 DATAFIX	Data Fix Enable. Controls the use of the first 8 address comparators for 1-word data fix or for code patch routine. 0 Address comparator triggers a opcode patch 1 Address comparator triggers a data fix

45.3.3 ROMPATCH Enable Registers Low (ROMCP_ROMPATCHENL)

The ROMPATCH enable register low (ROMPATCHENL) controls whether or not the associated address comparator can trigger an opcode patch or data fix event. This implementation of the ROMPATCH only has 16 comparators. Therefore, the upper half of ROMPATCHENL is reserved, and ROMPATCHENL[15:0] are associated with comparators 15 through 0.

Address: ROMCP_ROMPATCHENL is 63FB_8000h base + FCh offset = 63FB_80FCh

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

ROMCP_ROMPATCHENL field descriptions

Field	Description
31–16 -	Reserved
15–0 ENABLE	Enable Address Comparator. This bit enables the corresponding address comparator to trigger an event. 0 Address comparator disabled 1 Address comparator enabled, ROMPATCH will trigger a opcode patch or data fix event upon matching of the associated address

45.3.4 ROMPATCH Address Registers (ROMCP_ROMPATCHA*n*)

The ROMPATCH address registers (ROMPATCHA0 through ROMPATCHA15) store the memory addresses where opcode patching begins and data fixing occurs. ROMPATCHA0 through ROMPATCHA15 are associated with address comparators 0 through 15. Each of the ROMPATCH address registers is 21 bits wide and dedicated to one 4 Mbyte memory space. Bits 21 through 2 are address bits, to be compared for a match; bit 1 is also an address bit used for half word selection. Bit 0 is the mode bit (set to 1 for THUMB mode). All 16 registers can be used for code patch address comparison. Only the first 8 registers can be used for a 1-word data fix address comparison.

Addresses: 63FB_8000h base + 100h offset + (4d × *n*), where *n* = 0d to 15d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-										ADDRX[21:1]					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDRX[21:1]															THUMBX
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROMCP_ROMPATCHA*n* field descriptions

Field	Description
31–22 -	Reserved
21–1 ADDRX[21:1]	Address Comparator Registers. Indicates the memory address to be watched. Bit 1 is ignored if data fix. Only used in code patch

Table continues on the next page...

ROMCP_ROMPATCHA*n* field descriptions (continued)

Field	Description
0 THUMBX	THUMB Comparator Select. Indicates that this address will trigger a THUMB opcode patch or an ARM opcode patch. If this watchpoint is selected to be a data fix, then this bit is ignored as all data fixes are 1-word data fixes. 0 ARM patch 1 THUMB patch (ignore if data fix)

45.3.5 ROMPATCH Status Register (ROMCP_ROMPATCHSR)

The ROMPATCH status register (ROMPATCHSR) indicates the current state of the ROMPATCH and the source number of the most recent address comparator event.

Address: ROMCP_ROMPATCHSR is 63FB_8000h base + 208h offset = 63FB_8208h

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

ROMCP_ROMPATCHSR field descriptions

Field	Description
31–18 -	Reserved
17 SW	ROMC AHB Multiple Address Comparator matches Indicator. Indicates that multiple address comparator matches occurred. Writing a 1 to this bit will clear this it. 0 no event or comparator collisions 1 a collision has occurred
16–6 -	Reserved
5–0 SOURCE	ROMPATCH Source Number. Binary encoding of the number of the address comparator which has an address match in the most recent patch event on ROMC AHB. If multiple matches occurred, the highest priority source number is used. 0 Address Comparator 0 matched 1 Address Comparator 1 matched 15 Address Comparator 15 matched

Chapter 46
Smart Direct Memory Access Controller (SDMA)

46.1 Introduction

The Smart Direct Memory Access (SDMA) controller offers highly-competitive DMA features combined with software-based virtual-DMA flexibility. It enables data transfers between peripheral I/O devices and internal/external memories.

The SDMA controller helps maximize system performance by off-loading the ARM core in dynamic data routing.

46.1.1 Overview

The figure below shows a block diagram of the SDMA controller. It includes the custom RISC core along with its RAM, ROM, DMA units, the CRC unit, and the scheduler.

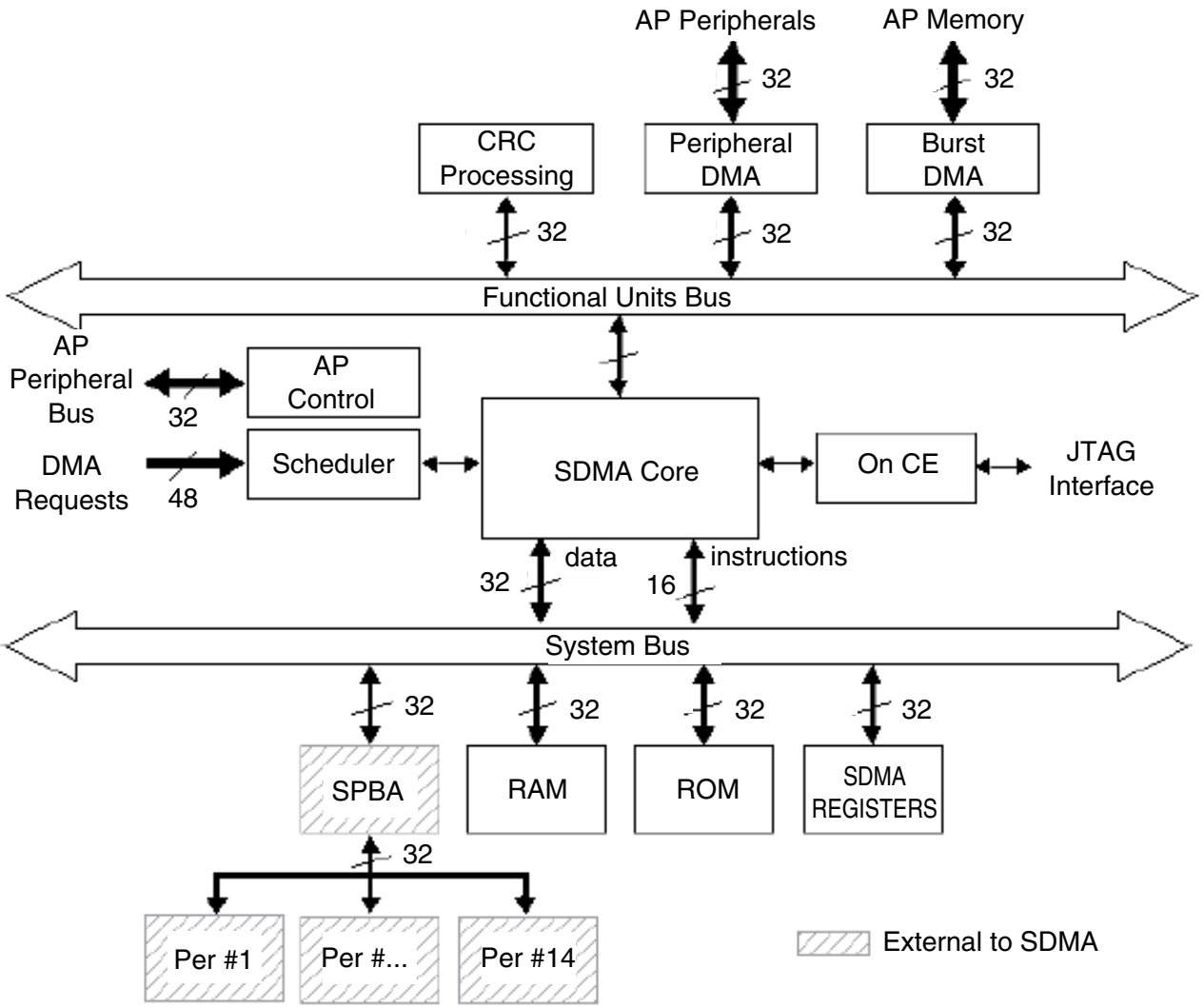


Figure 46-1. SDMA Block Diagram

The SDMA core executes short routines that perform DMA transfers; these routines are called *scripts*. The SDMA core interfaces to its own memory via the SDMA system bus. The SDMA system bus supports a 32-bit data path and a 16-bit address bus. The system bus datapath is used for both 16-bit instruction (program) memory access and 32-bit data access. DMA units interface to the core via the Functional Unit Bus and use dedicated registers to perform DMA transfers.

The SDMA memory contains a ROM and a RAM. The ROM contains startup scripts (for example, boot code) and other common utilities, which are referenced by the scripts that reside in the RAM. The internal RAM is divided into a context area and a script area (more details about this mapping are available in Instruction Memory Map and Data Memory Map).

Every transfer channel requires one context area to keep the contents of all the core and unit registers while inactive. Channel scripts are downloaded into the internal RAM by the SDMA using a dedicated channel that is started during the boot sequence. Downloads are invoked using commands and pointers provided by the ARM platform. Every channel contains a corresponding channel script located in RAM and/or ROM that can be reconfigured independently as-needed. Channel scripts can be stored in an external memory and downloaded when needed. The SDMA can be configured with any mixture of scripts to enable an endless combination of supported services.

The scheduler monitors and detects DMA requests, mapping them to channels, and mapping individual channels to a pre-configured priority. At any given point, the scheduler presents the highest priority channel that requires service to the SDMA core. A special SDMA core instruction is used to "conditionally yield" the current channel being executed to an eligible channel that requires service. If (and only if) there is an eligible channel pending, will the current channel execution be preempted.

There are two yield instructions that differently determine the eligible channels: In the first version, eligible channels are pending channels with a strictly higher priority than the current channel priority. In the second version (yieldge), eligible channels are pending channels with a priority that is greater or equal to the current channel priority. The scheduler detects devices that need service through its 48 DMA request inputs. After a request is detected, the scheduler determines the channel(s) that is (are) triggered by this request and marks it (them) as pending in the "Channel Pending (EP)" register. The priorities of all the pending channels are continuously evaluated in order to update the highest pending priority. The channel pending flag is cleared by the channel script when the transfer has completed.

The ARM platform control block contains the control registers used to configure the 32 individual channels. There are 48 Channel Enable registers, and every register maps one DMA request to any desired combination of channels. The 32 Priority registers are used to assign a programmable 1-of-7 level priority to every possible channel. This block also contains all other control registers that the ARM platform can access.

The 48 DMA requests that are connected to the scheduler come from a variety of sources. The "receive register full" and "transmit register empty" signals found in the UART and USB ports are typical examples of DMA requests that can be connected to the SDMA. These requests can be used to trigger a specific SDMA channel, or several channels.

There is an OnCE compatible debug port for product development. The OnCE includes support for setting breakpoints, single-step and trace, and register dump capability. In addition, all memory locations are accessible from the debug port.

46.1.2 Features

The following are the SDMA features:

- Multi-channel DMA supporting up to 32 time-division multiplexed DMA channels
- Hardware or software driven triggers for each channel
- 48 hardware driven triggers that can be mapped to any channel.
- Memory accesses including linear addressing, FIFO addressing and 2D addressing
- Fast context-switching with two-level, priority-based preemptive multi-tasking
- 16-bit instruction-set micro-RISC engine (the SDMA core)
- Two DMA units with some or all the following features:
  - Auto-flush and prefetch capability
  - Flexible address management (increment, decrement, and no address changes on source and destination address)
  - Misaligned data-transfer support
  - Uni-directional and bi-directional flows (copy mode)
  - Up to eight-word buffers for configurable burst transfers
- Support of byte-swapping and CRC calculations
- An available API and library of scripts
- Little-Endian and Big-Endian modes
- Hardware handshakes for low-power entry sequence
- Security support to lock contents of the SDMA script RAM.
- 4-Kbyte ROM containing startup scripts (for example, boot code) and other common utilities that can be referenced by RAM-located scripts
- 8-Kbyte RAM area is divided into a processor context area and a code space area used to store channel scripts that are downloaded from the system memory
- Debug support, including a OnCE port, real-time monitors, and embedded cross-trigger events
- Supported clock frequencies in process:
  - Configurable clock options for the SDMA core and the ARM platform DMA units
    - 1:2 ratio with maximum of SDMA core running at ARM platform Peripheral Bus speed and DMA running at max DMA frequency.
    - 1:1 ratio when both SDMA core and ARM platform DMA clocks are set to the ARM platform Peripheral Bus speed.
- Peripheral bus interface for configuration register programming by the ARM platform
- The SDMA RISC engine (arithmetic and logic operations, plus CRC acceleration), which is referred to as the "SDMA core."
- An internal peripheral bus connected to the Shared Peripherals Bus Interface (SPBA) that enables access to up to 14 shared peripherals. SDMA supports 32-bit accesses to word peripherals and 16-bit accesses to half-word peripherals.

- The peripheral DMA unit that is hooked-up to the ARM platform Crossbar Switch to service ARM peripherals
- The burst DMA unit is able to perform burst accesses to the external memory
- All the DMA units are 32-bit AHB masters. They are connected to different buses, thus allowing concurrent accesses.

46.2 Functional Description

Figure 46-2 shows the SDMA topology, and is composed of the following components:

- SDMA Core (SDMA Core)
- SDMA Scheduler (Scheduler)
- Functional Units:
  - CRC (CRC Calculation Unit)
  - Burst DMA (Burst DMA Unit)
  - Peripheral DMA (Peripheral DMA Unit)
- ARM platform Control for ARM control register access.
- Internal RAM and ROM Memory (SDMA Programming Model)
- OnCE debug Port (The OnCE Controller)

The functional unit bus provides access by the SDMA core to the CRC hardware accelerator and the DMA units. The system bus provides access to SDMA internal memory and also supports up to 14 peripherals.

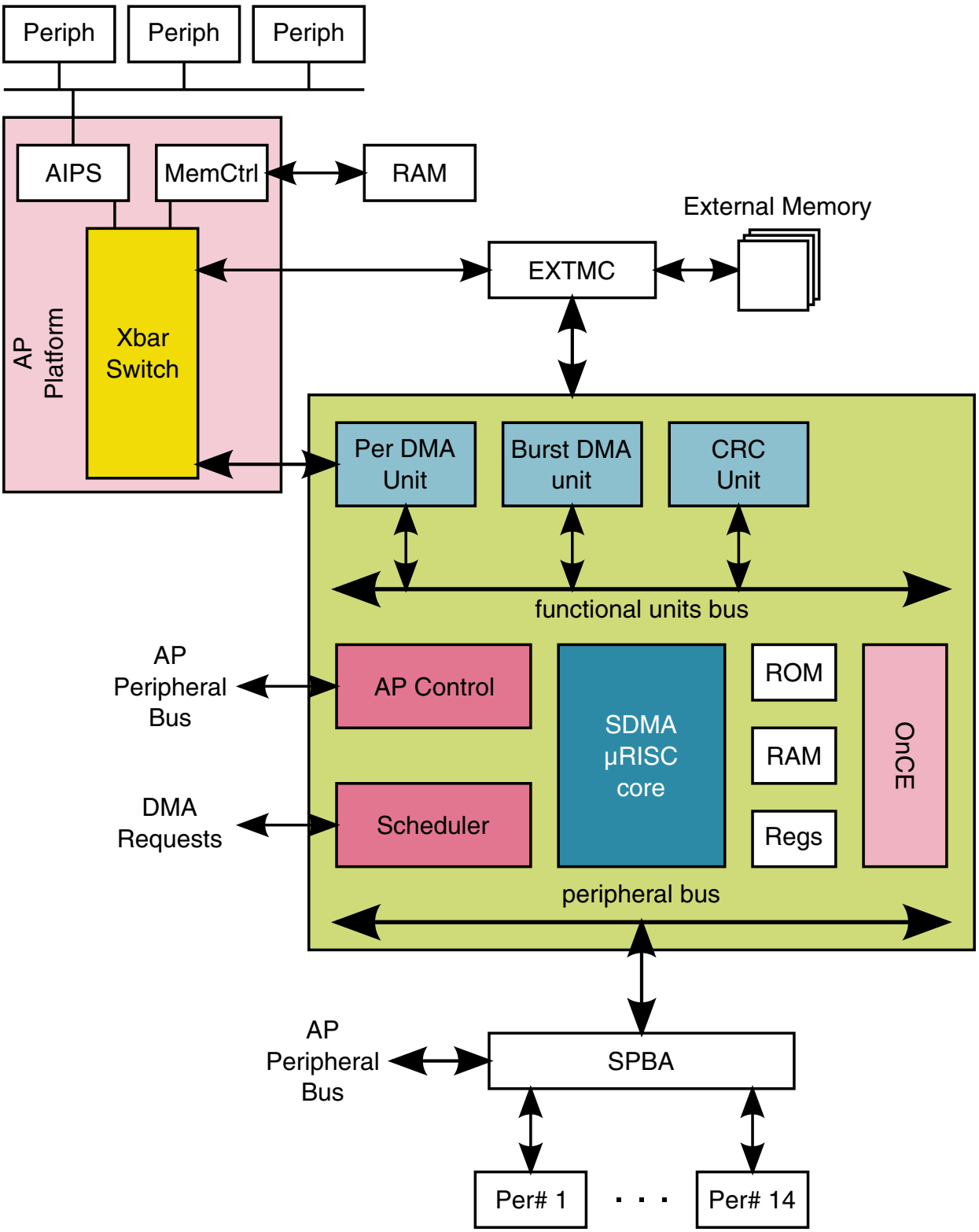


Figure 46-2. SDMA Connections

NOTE

EDITOR NOTE: This copy of the figure is for the case where the BP interfaces are NOT connected. Unconnected interfaces are removed from this diagram since not visible at all to the user.

46.3 SDMA Core

The SDMA core is a customized RISC-like processor that is specifically developed to control DMA units and perform L1 tasks like byte-stuffing or framing.

The SDMA core incorporates on-chip debug capability using the OnCE.

The SDMA core is based on a 32-bit register architecture with 16-bit instructions. There are eight general purpose 32-bit registers, four flags (T, LM, SF, and DF), and four PCU registers (PC, RPC, SPC, and EPC) that can address 16,384 16-bit instructions.

46.3.1 SDMA Core Structure

The following figure shows the structure of the SDMA core. It also shows the different registers, calculation resources, and possible data movements.

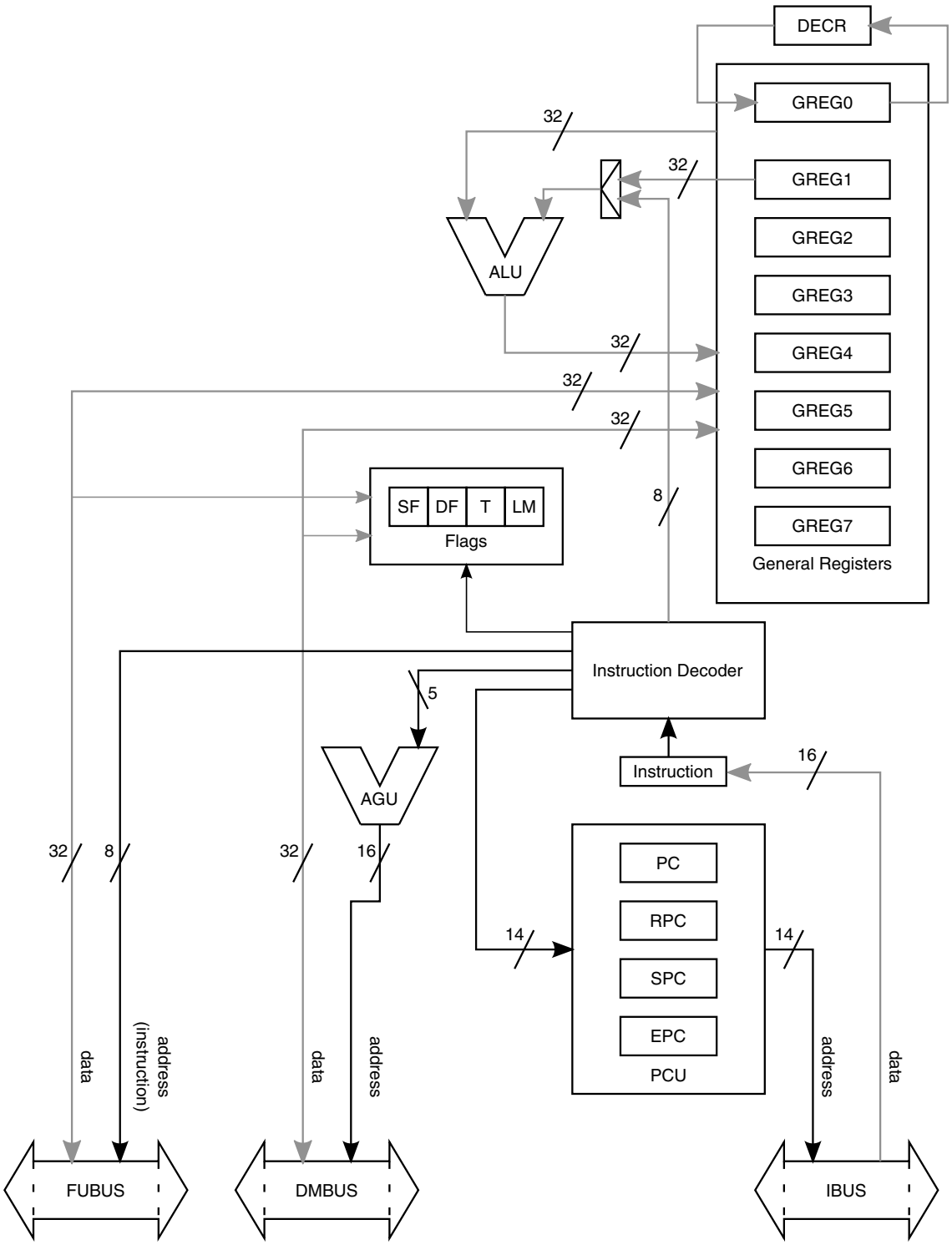


Figure 46-3. SDMA Core

- The Program Control Unit (PCU) is described in Program Control Unit (PCU). It handles the state of the core and generates the instruction fetch addresses. Instructions are retrieved from the Instruction Bus (IBUS) and stored in the SDMA

core instruction register prior to their decoding. The PCU contains the following registers:
    - The Program Counter (PC) contains the address of the current instruction.
    - The Return Program Counter (RPC) contains the address of the instruction that follows a jump to the subroutine.
    - The Start Program Counter (SPC) contains the address of the first instruction of the current hardware loop.
    - End Program Counter (EPC) contains the address of the last instruction of the current hardware loop.
- The other core registers are the general purpose registers (GREGn) and the flags.
    - The general purpose registers can be used to hold data and addresses. They can be loaded with immediate values (for example, 8-bit data that are encoded in the instruction), results of calculations that were performed with the ALU, 32-bit data that comes from the memory or peripherals via the Data Memory Bus (DMBUS), 32-bit data that comes from the DMAs or CRC via the Functional Units Bus (FUBUS) or another general purpose register. Their content can be the operands of the ALU, the data to send on either bus (DMBUS or FUBUS), or a pointer to memory (DMBUS address).
    - The general register 0 (GREG0) is also the hardware loop counter. In hardware loops, it cannot be used for any other purpose. This register uses a dedicated decrement unit (DECR) shown in Figure 46-3.
    - The flags reflect the status of operations:
        - SF and DF are set when the last load or store on either bus (FUBUS or DMBUS) received an error response.
        - LM is set when the core is executing instructions inside a hardware loop.
        - T is set when the ALU operation result was 0 or the loop counter reaches 0 (the latter is preponderant when an ALU operation is the last instruction of a hardware loop).
- The ALU has two operands: any general register and either a second general register or an immediate value. The result is always stored into the first general register. A NOP function can be utilized by moving a register's contents into itself (For example, the instruction: mov R0,R0).
- The 16-bit instructions are fetched via the instruction bus (IBUS) whose address is driven by the PC. The SDMA RAM and ROM are visible to the core as 16-bit devices through this interface.
- The memory (RAM and ROM), memory mapped registers, and external peripherals are accessed via the DMBUS. The address is always taken from a general register whose content is added to a 5-bit immediate value. This is the only available addressing mode. The DMBUS is a 32-bit data bus. Except for the peripherals that

are external to the SDMA, the address accuracy is the 32-bit word (for example, adding 1 to an address points to the next word, not the next byte).
- The functional units are accessed via the FUBUS connection. The data is exchanged with any general register, but the address (which in fact is the instruction and the selector of the functional unit) comes from an 8-bit field of the corresponding load or store.

46.3.2 Program Control Unit (PCU)

This part of the SDMA core is dedicated to the control of the RISC engine, as implied by the instructions that are executed. Its behavior is determined by the instruction type and the inputs of the SDMA.

It contains the PC, RPC, SPC, and EPC registers that are described in SDMA Core Structure.

46.3.2.1 Instruction Types

The state sequence and the delay of execution vary according to the type of the instruction. There are six possible categories of instructions, as follows:

1. Standard: Most of the instructions belong to this category, and always last 1 cycle.
2. ldf/stf: These are respectively the load and store instructions that access the functional units. They last 1+n cycles where n is the number of wait-states of the targeted functional unit.
3. ld/st: These are the load and store instructions that access the memory and peripherals. They last 1+n cycles where n is the number of wait-states of the targeted device (1 for the ROM, RAM, and memory mapped registers, 1 + the external peripheral wait-states). These instructions always last at least two cycles, but the core is able to handle them in one cycle. The first wait-state is inserted outside the core.
4. Branch: These are all the instructions that cause the Program Counter to point to another instruction other than the following one (for example, one that breaks the sequential flow). There are the absolute jumps, the conditional branches, the jump to the sub-routines, and the return from the sub-routine.
5. Loop,Modified Load or Store: The hardware loop instruction modifies the potential behavior of any load or store inside the loop (for example, when the LM flag is set). A jump may be implied after any such load or store if it received an error. The error causes an early exit of the loop, which means a jump to the instruction that follows the one that is pointed to by EPC. An additional cycle is required by the PCU to perform the jump (+1 to the ld/st/ldf/stf original execution delay). Although there is

usually an implicit jump after the last instruction of the loop when the PC goes back to SPC, this is performed at no cycle cost.

6. Done: The done, yield, or yieldge instructions are used to control channel switching. When no channel switching is performed, these instructions last a single cycle. When there is a change of channel or context switch, the delay is variable and depends on many factors (as detailed in Context Switching).

46.3.2.2 PCU States

The PCU state is visible through outputs of the SDMA (see Real-Time Debug Outputs) or the OnCE status register(see OnCE Status Register (OSTAT)).

The PCU state is a four-bit field that can take the values shown in the following table. Figure 46-4 shows the possible state transitions and the corresponding conditions.

Table 46-1. PCU States

Value	State	Description
0	Program	The is the usual instruction cycle.
1	Data	This state is inserted when there are wait-states during a load or a store on the data bus (ld/st type).
2	Change of Flow	This is the second cycle of any instruction that breaks the sequence of instructions (branch and done types). This state lasts only a single cycle; it is always followed by the Program state.
3	Error in Loop	This state is used when an error causes a hardware loop exit (loop-modified load or store type). This state only lasts a single cycle; it is always followed by the Program state.
4	Debug	he SDMA is stopped in debug mode.
5	Functional Unit	This state is inserted when there are wait-states during a load or a store on the functional units bus (ldf/stf type).
6	Sleep	No script is running: The core is idle after saving the last channel context.
7	Save	The context switch FSM is saving the current channel.
8	Program in Sleep	Same as Program except there is no associated channel, this state is used when instructions are executed after entering debug mode, whereas the core was in either Sleep mode.
9	Data in Sleep	This is the same as Data except there is no associated channel.
10	Change of Flow in Sleep	This is the same as Change of Flow except there is no associated channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
11	Error in Loop in Sleep	This is the same as Error in Loop except there is no associated. channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
12	Debug in Sleep	This is the same as Debug except the core was put in debug mode when no channel was active.

Table continues on the next page...

Table 46-1. PCU States (continued)

Value	State	Description
13	Functional Unit in Sleep	This is the same as Functional Unit except there is no associated channel.
14	Sleep after Reset	This shows that no script is running, and the core is idle after a reset. When a channel becomes active, no context is restored but the core starts its boot program located at address 0 (or the address available in register in Channel 0 Boot Address (SDMAARM_CHN0ADDR)).
15	Restore	The context switch FSM is restoring the next channel context.

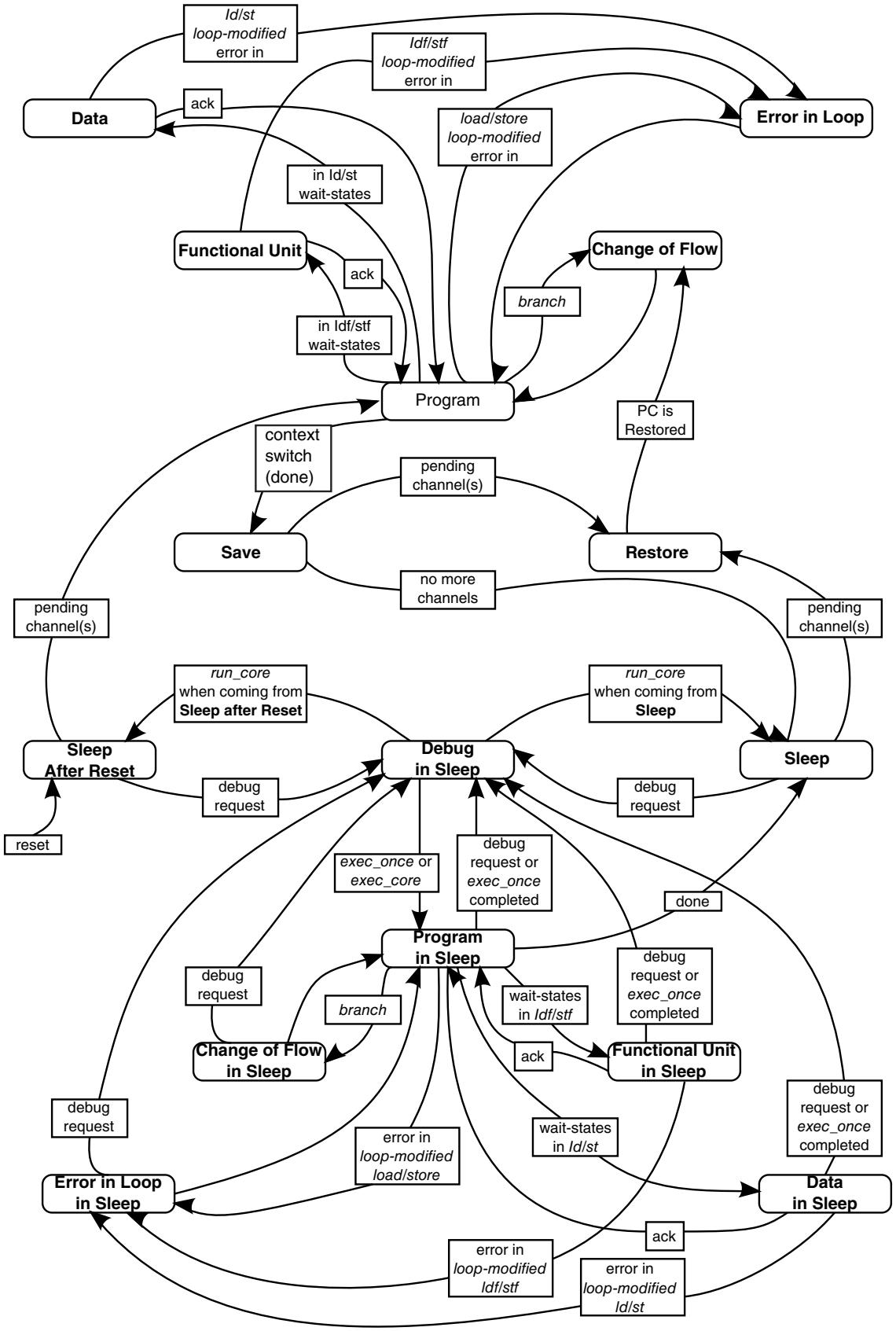


Figure 46-4. PCU State Diagram

46.3.3 SDMA Core Memory

The SDMA has two memory spaces: one for the instructions and one for the data. As both spaces share the same resources (ROM and RAM devices), the system bus manages possible conflicts when the core accesses the same resource for both an instruction read and a data read or write.

Program and data memory is further described in Address Space.

Instructions of 16-bit width are stored in 32-bit wide devices and can be accessed as data. The mapping is Big Endian: an even instruction address (terminated by 0) accesses the most significant part of the 32-bit data (bits [31:16]), and an odd instruction address (terminated by 1) accesses the least significant part of the 32-bit data (bits [15:0]). Instructions can be fetched out of internal ROM or RAM.

Data can be read from ROM, RAM, memory mapped registers, and external peripherals, and written to the same devices (except the ROM).

The ROM contains bootload scripts, channel scripts, and common subroutines which may be referenced by channel scripts elsewhere in the ROM or RAM.

The RAM is divided into a context area and a code space area which may be used to store channel scripts. The RAM contains undefined values after a hardware reset. Channel scripts and initial context values are downloaded into RAM using channel 0 which is reserved for bootload functions.

46.4 Scheduler

All channel scheduling hardware is included in the Scheduler.

46.4.1 Primary Functions

The scheduler is a hardware-based design used to coordinate the timely execution of 32 virtual DMA channels by the SDMA core on the basis of channel status and priority.

The scheduler performs the following functions:

- Monitors, detects, and registers the occurrence of any one of the 48 DMA requests
- Links a specific request to a channel or group of channels (channel mapping)
- Ignores requests that are not mapped to a previously configured channel
- Maintains a list of all the channels that are requesting service

- Assigns a pre-programmed priority level (1 of 7) to every channel requesting service
- Detects and flags overrun/underrun conditions

46.4.2 Channels and DMA Requests

46.4.2.1 Channels

A Virtual Channel (hereafter simply called a channel) manages a flow of data through the SDMA. Flows are typically unidirectional.

The SDMA can have up to 32 simultaneously operating channels, numbered from 0 to 31. Channel 0 is usually dedicated to control the SDMA script downloading. All the channels can be assigned by the ARM platform software.

46.4.2.2 DMA Requests

A DMA request is caused by externally (for example, external to the SDMA) controlled conditions (for example, UART receive FIFO reaches a threshold). The SDMA currently supports up to 48 DMA requests.

46.4.2.3 Mapping from DMA Requests to Channels and Priorities

A channel can stall waiting on a single DMA request. A single DMA request can awake more than one channel (in fact, any request can awake any combination of channels).

The mapping between DMA requests and channels is program-controlled. There is a storage element assigned for each of the 48 requests that contains a bitmap table of the channels that are awakened by the event.

Every channel also has a three-bit register that indicates its priority.

46.4.3 Scheduler Functional Description

Scheduler Overview describes the behavior of the SDMA scheduler-from the channel enabling conditions to the highest priority pending channel selection.

46.4.3.1 Scheduler Overview

The scheduler algorithm is built in hardware. It is provided with possibilities for the ARM platform to control its behavior.

The scheduler processes incoming DMA requests, maps detected requests to 0, one, or several channels, maintains a list of channels that are requesting service (pending channels), identifies the top priority and its associated channel, and selects the next active channel when the current channel yields.

The following figure shows a functional overview.

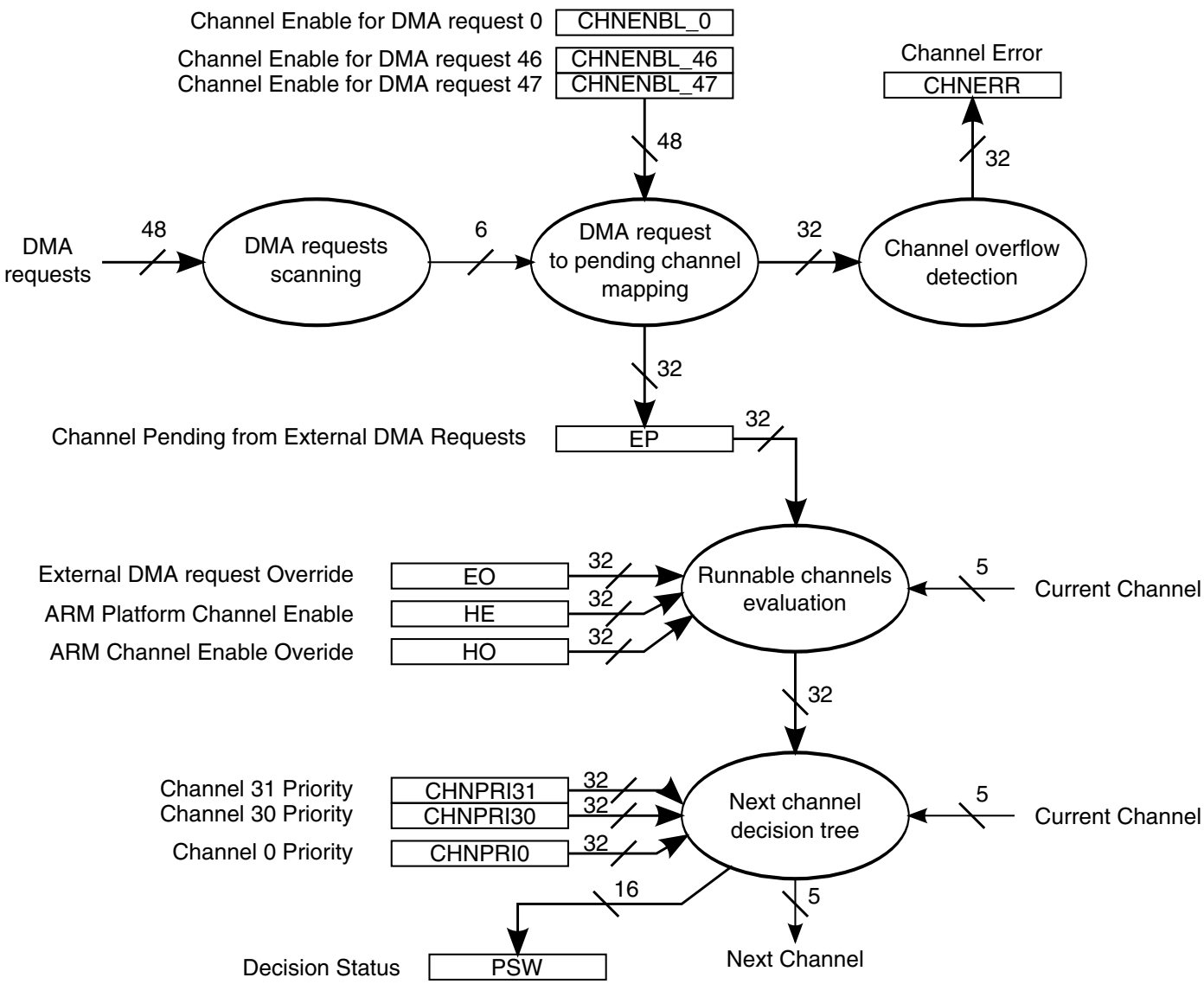


Figure 46-5. SDMA Hardware Scheduler

46.4.3.2 DMA Requests Scanning

The scheduler contains a 48-bit edge detection device that detects the rising edge of every DMA request and transmits the request number to the next stage.

The DMA requests are assumed to be generated on the same reference clock as the SDMA core clock; they are detected as soon as the signal goes from a 1-to-n-cycles low state to a 1-to-m-cycles high state.

This system is able to detect single-cycle pulses as well as level-based DMA requests such as a FIFO threshold crossing. In this case, the SDMA provides a memory mapped register that can be used by the channel script to monitor the DMA requests lines, and thus determines whether the data transfer is done or not done, and then continues with the transfer or closes the channel.

When several DMA requests are detected at the same time, they are forwarded to the next scheduler stage at the rate of one request per cycle. No request is lost.

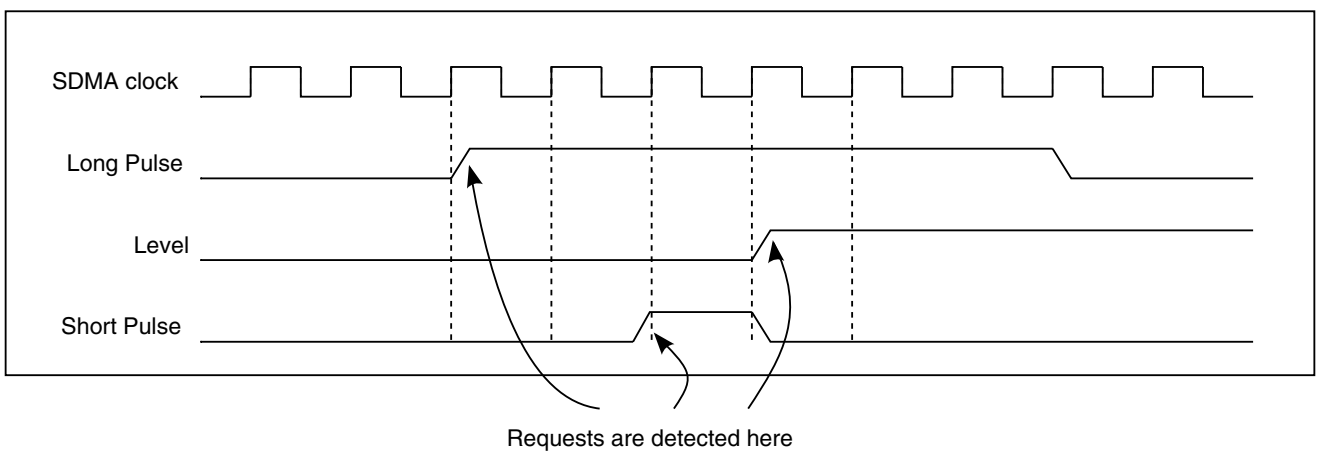


Figure 46-6. Examples of Valid DMA Requests

The DMA request inputs are connected to various sources that depend on the SoC. The exact list of DMA request inputs and their associated number is available in each respective project-specific chapter.

46.4.3.3 Mapping DMA Requests to Pending Channels

Whenever a DMA request is detected by the first stage, its number is used in the second stage to determine the channels that have to be activated.

This is performed with an array of 48 registers that are 32 bits wide: There are 48 Channel Enable Registers (CHNENBLn), one register per DMA request. The DMA request number selects the Channel Enable Registers, and every bit of this 32-bit register indicates that the corresponding channel must be activated when it is a 1.

This information is passed on the EP register. For every bit of the Channel Enable Register that is set, the corresponding bit of the EP register is also set, and the remaining bits of EP are left unchanged. The transformation of EP is summarized by the following equation:

EP = EP or CHNENBLn

The EP register is used to know which channels require service because they received a DMA request.

Typical contents of the CHNENBLn registers are all 0s, except for a single bit set. For example, a DMA request triggers one channel, but all 0s or several 1s are possible. One DMA request could activate several channels, and the channel execution sequence can be controlled by the channel priorities and numbers, as explained in the next sections. The following table illustrates an example configuration.

NOTE

From the table, the DMA request 0 is programmed to simultaneously trigger channels 0, 1, and 31. Also, DMA requests 30-47 are not used in this example. The remaining channels 2 to 30, are configured to be triggered by DMA requests 29 to 1, respectively.

Table 46-2. Channel Enable RAM Programming Example

DMA Request Number	Channel																															
	31																															0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 46-2. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																															
	31																															0
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table continues on the next page...

Table 46-2. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel 31																														0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 46-2. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																															
	31																															0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.4.3.4 Channel Overflow

A channel overflow occurs when a DMA request requires service from channel *n* by setting bit *n* of the register EP, but this bit is already set, meaning channel *n* is already pending. This can come from an overrun/underrun condition.

This detection is possible only when the DMA requests are pulses, because a level-based DMA request stays high until it is serviced, even though an underrun or overrun condition occurs, thus preventing another edge detection of the DMA request.

The channel overflow information is saved in the 32-bit CHNERR register (1 bit per channel). You can configure the SDMA to trigger an interrupt to the ARM platform when there are 1s in CHNERR. Every bit of CHNERR is masked with the corresponding bit of INTRMASK and if it gives a 1, the corresponding bit of INTR is set, triggering the interrupt.

46.4.3.5 Runnable Channels Evaluation

The EP register is used in conjunction with several other 32-bit registers to determine the channels that are runnable.

Registers EO, DO, HO and HE, are controlled by the ARM platform. EP is controlled by the DMA requests and their mapping to channels.

Several channels may be runnable at any given time. The i^{th} channel is runnable if (and only if) the condition below is true:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

After reset, the HE[i], HO[i], EP[i], and EO[i] bits are all cleared whereas the DO[i] bits are all set. The functions associated with DO are not available for this device. When DO[i] is set, the scheduler condition becomes:

(HE[i] or HO[i]) and (EP[i] or EO[i])

The registers in these equations are controlled as follows:

- ARM platform (host) channel enable flag HE[i] may be set or cleared by the ARM platform with the HSTART and STOP_STAT registers. It can also be cleared by the i^{th} channel script.

  Typical usage is for the ARM platform to set this flag to activate the channel. The flag is cleared by the SDMA core when the transfer is done.

- Externally triggered channel pending flag EP[i] is set by the scheduler when the channel was activated by a DMA request. It can be cleared by the i^{th} channel script.
- The ARM platform channel override flag HO[i] may be set or cleared by the ARM platform. When set, it enables the i^{th} channel to run without the involvement of the ARM platform.

  Typical usage is for the ARM platform to set this flag for channels that do not need ARM platform supervision such as channels that are controlled by DMA request events (EP).

- DO should always be set to 1 so that the runnable channel evaluation considers only HO, HE, EP, and EO.
- Externally triggered channel override flag EO[i] may be set or cleared by the ARM platform. When set, it prevents the i^{th} channel from stopping and stalling on incoming peripheral DMA requests. This is the case when the channel is not handling data transfers with peripherals (for example, a memory to memory transfer).

The SDMA can clear the HE[i], and EP[i] bits by means of adone or notify instruction. The done instruction causes a reschedule; thus, enabling another channel to preempt the current one, while the notify instruction does not. The done and notify instructions can clear either HE[i] or EP[i] (never more than one at a time).

Table 46-3. Runnable Channel Selection Control

Register	Set by	Cleared By
HO	Write to HOSTOVR register	Write to HOSTOVR register
HE	Write to HSTART register	Write to STOP_STAT register or by the channel script with the done or notify instructions.
DO	Write to DSPOVR register	Write to DSPOVR register
EO	Write to EVTOVER register	Write to EVTOVER register
EP	Set by external DMA request event input.	By the channel script with the done or notify instructions

46.4.3.6 Next Channel Decision Tree

The next channel number is computed from the runnable channels list, the current channel number, and their respective priorities.

It is re-evaluated every cycle, but is only used when the current channel yields or terminates by executing a yield, yieldge, or done instruction.

The decision tree is based on the selection of the runnable channel that has the highest priority.

The highest priority channel is selected according to the following rules:

- Runnable channels are sorted by priority.
- If one of the channels with the highest priority had been preempted by a channel with a higher priority, but did not want to yield to a channel of the same priority (for example, it executed a yield, not a yieldge), it is elected as the next channel.
- The channels that belong to the highest priority group are sorted by their number and the channel that has the highest number in this group becomes the next channel. For example, if priorities are the same, channel 31 will be selected before channel 30.

When the current channel requires a reschedule with a yield(ge) or a done instruction, the context switch decision is based on the instruction parameter, the current channel number and priority, and the next channel number and priority. The possible cases are all listed in the following table. The grayed cells correspond to unusual cases that should not occur with a typical usage of the SDMA.

Table 46-4. Channel Switching Decision with a yield, yield(ge), or done

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
yield (done 0)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Current
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the ARM platform)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the ARM platform)
yieldge (done 1)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Next[1]
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the ARM platform)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the ARM platform)
done (done>1)	Not runnable	Not runnable	none	none[2]
	Runnable	Not runnable	none	Current[3] (occurs when the done instruction does not disable the channel runnable condition)
	Not runnable	Runnable	none	Next[1]
	Runnable	Runnable	none	Current[3] (occurs when the done instruction does not disable the channel runnable condition)

1. Current channel script execution is stopped, its context is saved; the next channel context is restored and its script execution resumes
2. Current channel context is saved and SDMA enters IDLE mode
3. Current channel context is saved, then restored, and the current channel script resumes execution

Finally, when the SDMA is in IDLE mode and a runnable channel is elected as the next channel, its context is immediately restored and the script execution resumes.

The *combinatorial-decision* tree supports dynamic modifications of the EP, EO, HE, HO, and DO flags as well as dynamic modifications of the channel priorities. The propagation times are detailed in Scheduler Pipeline Timing Diagram.

The decision tree status is available in the PSW register, which is continuously updated. It contains the next channel priority, the next channel number, the current channel priority, and the current channel number. When a priority is read as 0, it means the channel is not runnable.

A few examples of decisions are presented below:

- Channel 31 is running with priority 5, channels 13 and 24 are pending with the same priority 5; channel 24 is eligible as the next channel since 24 > 13.
- Channel 31 is running with priority 7, channels 13 and 24 are pending with priority 5; channel 31 is the next channel because its priority is greater than the other pending channels.
- Channels 7, 23, and 29 are pending with the same priority. Channel 7 is active and runs a yieldge; it is preempted by channel 29. After a period of time, channel 29 runs a yieldge, it is then preempted by channel 23 that is the selected channel since channel 29 is the current channel. Later, channel 23 runs a yieldge and is preempted by channel 29. Channels 23 and 29 will go on switching after every yieldge until one of them terminates. It is only at that point that channel 7 becomes eligible again.
- Channel 11 is running with priority 3, and channel 15 is pending with priority 4. When the channel 31 script executes a yield instruction, it gets preempted by channel 15; then channels 6 and 18 with priority 3 become pending. Because channel 11 was preempted after executing a yield and there is no pending channel with a strictly greater priority, it is eligible as the next channel (although its number 11 < 18).

46.4.3.7 Scheduler State Diagram

The Figure 46-7 summarizes the behavior of the SDMA scheduler with details about the exact mechanism of the priority decision tree. It is important to understand the scheduler is a hardwired pipeline, which means all the stages are performed simultaneously every cycle, but a change on any given stage is reflected on the next stage after the delays presented in Scheduler Pipeline Timing Diagram.

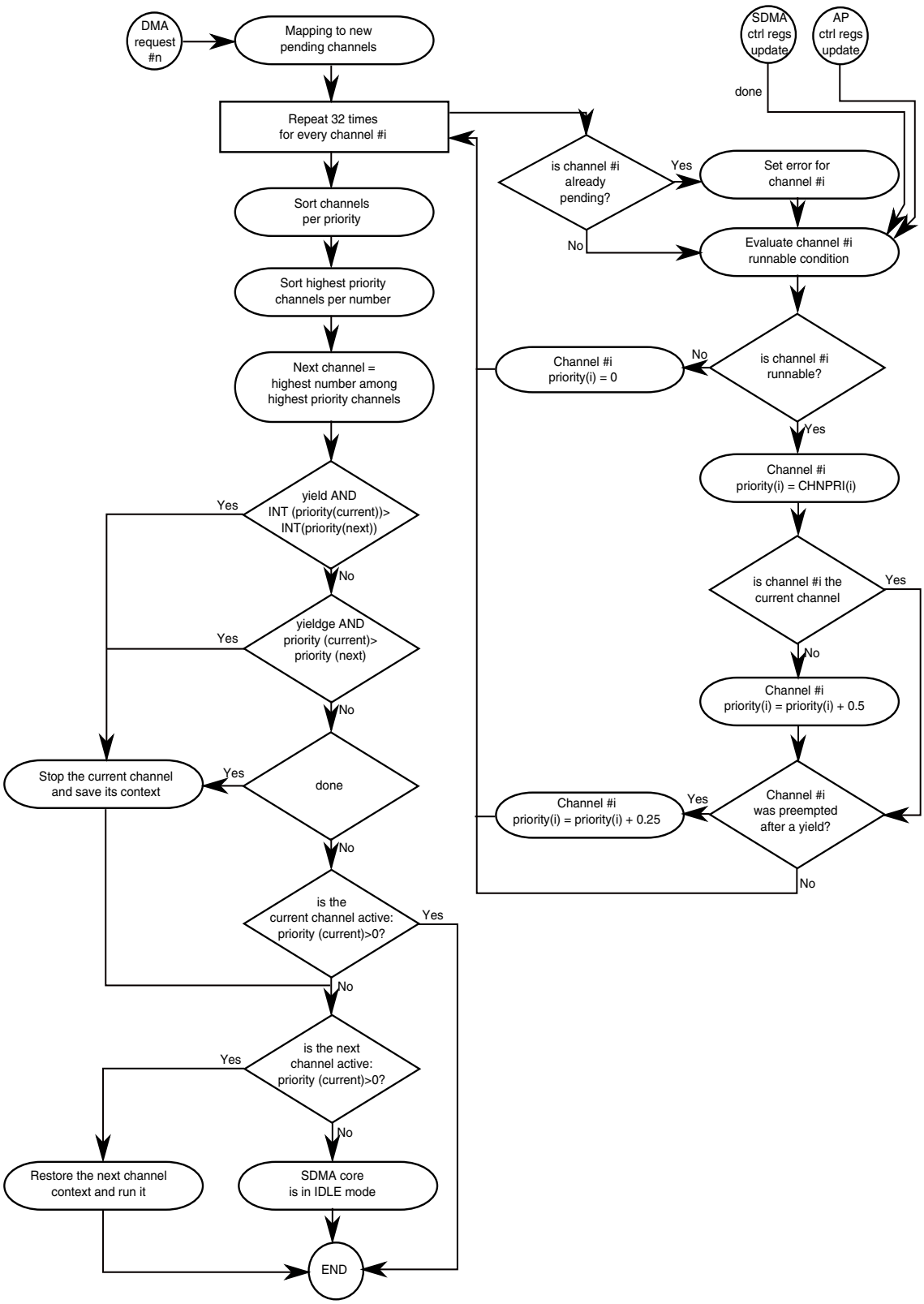


Figure 46-7. Scheduler State Diagram

46.4.3.8 Scheduler Pipeline Timing Diagram

The SDMA scheduler process of DMA-request and control-register modifications is not immediate.

The figure below shows the exact delays of all the tasks. The reference clock is the SDMA core clock.

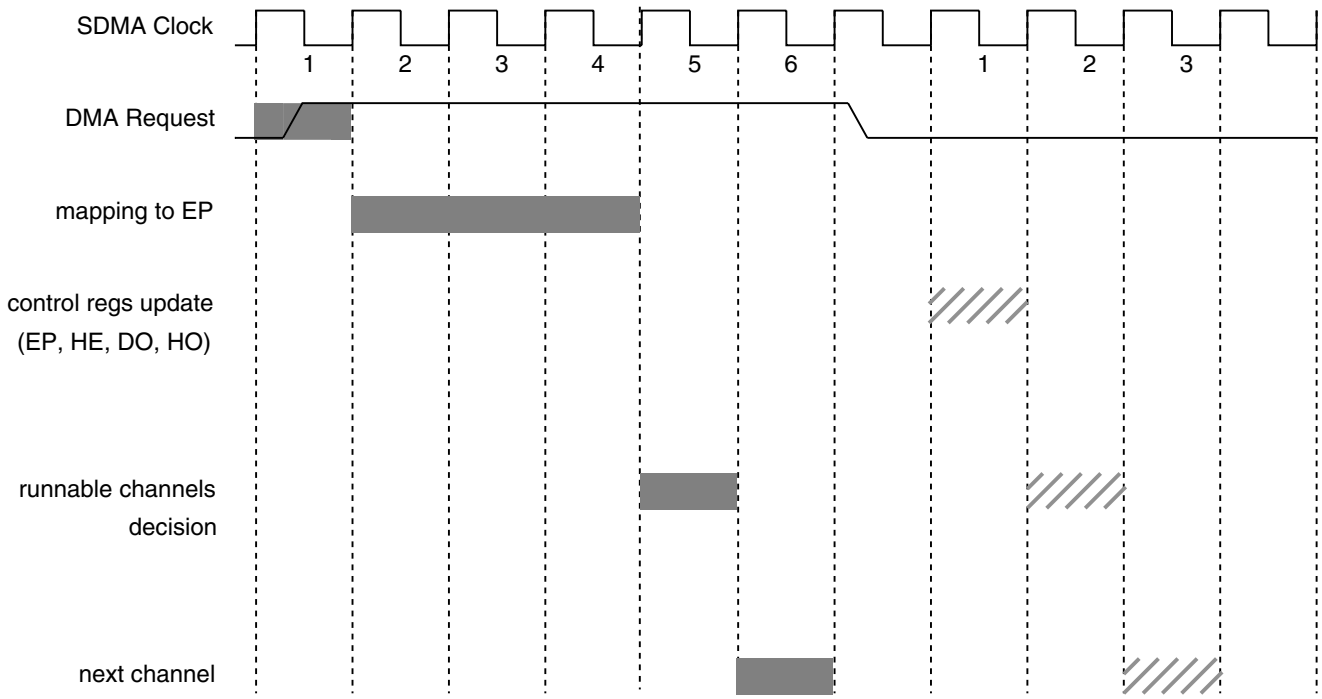


Figure 46-8. Scheduler Timing Diagram

Two numbers can be inferred from this timing diagram. First, it takes six SDMA core clock cycles to update the next channel from a DMA request. Second, it takes three SDMA core clock cycles to update the next channel from a direct modification of the condition registers (EP, DO, HE, or HO) by any processor. The processors that can modify these bits include SDMA with a done instruction or the ARM platform with a write access through the corresponding control port on their respective peripheral bus).

46.4.3.9 Channel-DMA Request Mapping

The 48 DMA request inputs to the SDMA scheduler are listed in project-specific chapters. Refer to the respective chapters for this information.

46.4.3.10 Examples: How to Start a Channel

A channel can be started when the following equation is true for channel *i*:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

Once this equation is true, the scheduler can start this channel according to the priority of all pending channels. Several examples of configuration are listed below:

1. To start a channel triggered by ARM platform software:
   - Initially, configure HO[i]=0, DO[i]=1, and EO[i]=1 using registers indicated in Table 46-3.
   - ARM platform software triggers the channel by writing to the HSTART register to set HE[i]=1, thereby setting the above equation true.
2. To start a channel triggered by DMA request event.
   - Initially, configure HO[i]=1, DO[i]=1, and EO[i]=0 using registers indicated in Table 46-3.
   - The DMA request is asserted to trigger the channel by setting EP[i]=1, which makes the above equation true.

46.4.4 Context Switching

On execution of a done or yield(ge) instruction, the current channel may be changed either because it has finished (which necessarily happens when the done instruction is executed), or it was preempted by a higher priority channel (which is possible but not systematic when the yield(ge) is executed).

Upon a channel change the SDMA goes through a context switch procedure. When the current channel yields or ends, the context for that channel is saved into the context RAM locations for that channel. When the next channel starts running, its context is first restored from RAM.

Since context RAM is not yet initialized by reset, there will be no context restore at the beginning of the first channel (bootload channel) run after reset. It is expected that the bootload channel will be used to initialize the context for all other channels. When the bootload channel finishes running or yields, SDMA will enter its SAVE state and save that channel's context into RAM. Then, if the bootload channel is called again later, the context will be restored from RAM when the channel starts again.

The context structure for each channel is defined in Context Switching-Programming andTable 46-10. There will be one context area reserved for each channel. When a channel ends or yields, the SDMA core registers are automatically saved into the context RAM and later restored from the context RAM when the channel is next run. The total

RAM space reserved for 32-channel contexts is either 3K or 4K depending on whether the SMSZ bit is set in the CHN0ADDR register, which enables an additional 8 words of scratch RAM for each context.

46.4.4.1 Context Switch Modes

The exact procedure to save the context of the old channel, and to restore the context of the new channel depends on the context switch mode selected by the ARM platform in the CONFIG control register.

The following are the context switch modes:

- By default, the "dynamic" context switch is set. This mode provides the most efficient context switch for an average of eight cycles to stop the current channel, save its context, restore the next channel context, and resume its execution. It consists of saving modified registers of the current channel in the background (for example, during the channel execution)-which leaves very few registers to save when the switch is decided-resuming execution of the next channel as soon as possible (for example, when the minimal set of registers is restored), and continuing the restore phase during this execution.
- In "dynamic with no loop" mode, the same principle is followed except the modified registers are only saved in the background when the loop flag is not set. This mode offers almost the same effectiveness as the previous one, but it prevents the system from accessing the RAM during loops to save power. This is the recommended mode for an efficient context-switch when the loop bodies are short.
- In "dynamic power" mode, no background saving is performed, which reduces power consumption to the minimum. The modified registers are only saved when the context switch starts. The restore phase is the same as before. This is the mode that achieves the optimal power consumption at the cost of a slower context-switch.
- In a "static" context switch, all the registers are saved when a context switch is decided, and all the registers are restored before starting the execution of the new channel. This mode enables a predictable behavior of the context switch since all the registers are restored prior to the channel start and all registers are saved after the channel termination.

NOTE

Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized during the context SAVE phase when the channel is done or yields. Subsequent calls to the same channel or different channels may use any of the dynamic context modes. This will ensure that all context locations for the bootload

channel are initialized, and prevent undefined values in context RAM from being loaded during the context restore if the channel is re-started later.

46.4.4.2 Context Switch Procedure

The Program Control Unit goes into the *save* state, the current context is spilled into memory, and the next channel context is restored according to the context-switch mode that was selected by the ARM platform.

The context switch procedure is as follows:

1. Load the current context's spill base address.
2. Spill the modified registers of the current channel to memory according to the selected context switch mode while the channel is running.

   On a done or yield(ge) that causes the channel preemption, the PCU goes into the *save* state. In *static* mode, all the registers are saved; whereas, in either *dynamic* mode, the registers that were modified but not yet saved are then saved, and the PCU registers and flags are finally saved.

3. Put the SDMA core into *sleep* and wait for new channels to be serviced. This step is skipped if there are pending channels when the current channel is saved.

   As soon as there is at least one pending channel, the PCU goes into its *restore* state to restore the context of the channel that was elected by the scheduler.

   Once a channel is elected, it remains the current channel until its script requests a rescheduling operation with a done or yield(ge) instruction. That means the current channel cannot be modified by the ARM platform, even if it is no more runnable or if its priority is modified.

   The ARM platform can however force a reschedule by writing the corresponding bit in the CONFIG register, which has the same effect as if the script had executed a done instruction. That feature should only be used to stop the SDMA in emergency cases.

4. Load the context base-address of the new channel.

   In "static" mode, all the registers are restored. In either "dynamic" modes, only the PCU registers are restored.

   The new channel is running. In "static" mode, no more activity regarding context restoring or saving is performed. In either "dynamic" modes, the registers are restored in the background every time an access to the context RAM is possible, and

priority is given to restoring the registers that are required by the next instruction to be executed. When a register has not been restored and the next instruction needs it, this instruction gets stalled until the register was restored.

In "dynamic" and "dynamic with no loop" modes, background saving of dirty registers is performed every time an access to the context RAM is possible and allowed by the context switch mode.

NOTE

The contents of a channel context space in the context RAM depends on the selected context switch mode. In "dynamic" and "dynamic with no loop" modes, the contents of the context RAM tend to match the contents of the SDMA registers (except for the PCU registers and flags that are never saved in the background). In "dynamic power" and "static" modes, the contents of the context RAM remain unchanged until the channel terminates with a done or gets preempted.

46.4.4.3 Context Map in Memory

Refer to Context Switching-Programming.

46.5 Functional Units

The functional units are small systems that are used by the SDMA core to perform complex calculations like the CRC unit, or to handle data transfers between the core and a bus domain external to the SDMA.

The SDMA core is able to control and exchange data with these systems by sending instructions and reading or writing data from/to the functional units' registers via the FUBUS. This is done with the ldf and stf instructions.

The following sections provide introductions to the available functional units. Functional Units Programming Model provides descriptions the functional units' behaviors.

46.5.1 CRC Calculation Unit

The Cyclic Redundancy Check (CRC) unit can perform CRC calculation. A single byte of data can be processed every cycle, but up to four bytes can be simultaneously loaded.

The CRC unit supports the following set of polynomials:

CRC32: $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$

CRC16: $X^{16}+X^{15}+X^2+1$

CCITT16: $X^{16}+X^{12}+X^5+1$

IS136: $X^{12}+X^{10}+X^8+X^5+X^4+X^3+1$

CRC10: $X^{10}+X^9+X^5+X^4+X+1$

CRC8: X^8+X^2+X+1

parity: X^8+1

46.5.1.1 CRC Structure

The following figure describes the overall structure of the CRC unit and introduces its registers that are accessible by the SDMA core via the FUBUS.

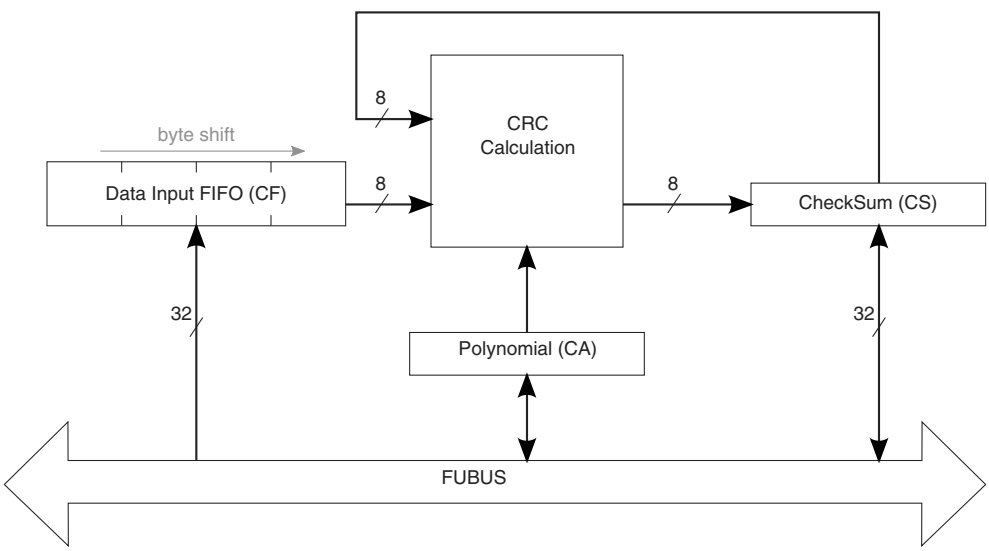


Figure 46-9. CRC Structure

46.5.1.2 CRC Data Processing

CRC processing requires the following three stages:

1. The preliminary initialization stage consists of selecting the desired polynomial, and storing the initialization pattern into the checksum register.
2. Data processing itself, which comes to feeding incoming bytes to the CRC input FIFO - bytes can be written one at a time (8-bit write access from the SDMA core), by pairs (16-bit write), or groups of four (32-bit write). One byte is processed every

cycle: Any subsequent access may stall the SDMA core until completion of the previous calculation.

3. The CRC result (or checksum) can be retrieved at any time, but all data must be processed before it is made available.

46.5.1.3 CRC Registers

The CRC enables reading and writing to three register addresses, which trigger the operations described in CRC Data Processing. The following are the three registers:

- *CA (CRC algorithm)*-The CA selects the desired polynomial. Reading and writing this register correspond to retrieving and changing the polynomial.
- *CS (CRC checksum)*-Writing to this register initializes the checksum accumulator, and reading this register yields the result from the CRC calculation (the result width depends on the polynomial size).
- *CF (CRC FIFO)*-Writing to this register loads the data into the input FIFO and triggers the CRC calculation process. It is not possible to read this register.

46.5.1.4 CRC Summary

Every operation mentioned in CRC Registers takes time to be executed by the CRC unit.

When the CRC receives a command, it immediately acknowledges the SDMA core provided it is not busy completing a previous operation. Therefore, wait-states are inserted by the CRC when a command succeeds another command that is not yet competed.

The following table lists the number of cycles that the CRC takes to execute every possible command. When an operation lasts one cycle, it means the CRC is able to process another command in the next clock cycle (for example, no wait-state is inserted by the CRC because of the former operation that is processing).

Table 46-5. CRC Processing Summary

Operation	Command	Delay	Comments
Set the polynomial	Write CA	1	
Retrieve the polynomial	Read CA	1	
Initialize the checksum	Write CS	1	
Retrieve the checksum	Read CS	1	
Store 1 byte to process	Write CF	1	
Store 2 bytes to process	Write CF	2	CRC is busy for 1 cycle

Table continues on the next page...

Table 46-5. CRC Processing Summary (continued)

Operation	Command	Delay	Comments
Store 4 bytes to process	Write CF	4	CRC is busy for 3 cycles

46.5.2 Burst DMA Unit

The burst DMA unit enables the SDMA core to perform data transfers to and from the ARM platform memory.

It is optimized for accessing SDRAM-like devices. It does not provide control to assign a privilege level to the DMA access. The burst DMA unit provides the SDMA with means to do the following:

- Perform up to 8-beat read and write bursts to the ARM platform memory, which optimizes throughput when accessing SDRAM-type devices because of an internal, 36-byte FIFO
- Access the ARM platform memory at once or twice the SDMA core frequency
- Copy data from one ARM platform memory location to another ARM platform memory location at the ARM platform bus speed, which provides a very high throughput
- Control the method for addressing the ARM platform memory (automatic increment of addresses or frozen addresses-the former aimed at accessing RAM-like memory and the latter aimed at accessing single-address FIFOs)
- Enable or disable automatic prefetch when reading data from the ARM platform memory. When the prefetch mode is selected, the burst DMA automatically triggers external bursts to fill its FIFO without waiting for the SDMA core to request the corresponding data, greatly improving throughput.
- Rely on the DMA to automatically flush its FIFO content when there is enough data to generate an 8-beat burst to the ARM platform memory. Or, it forces a flush when a data transfer must terminate.
- In the former case, the SDMA core may only be stalled when it tries writing data and there is not enough room left in the FIFO. In the latter case, the core is stalled until the data is effectively written to the ARM platform memory.

In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the ARM platform memory. This error status is retrieved by a later access to the burst DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the ARM platform memory is caught.

- Handle address alignment issues between the ARM platform memory map and the SDMA core data. This enables the core to read or write 32-bit data from the burst DMA, whereas the corresponding ARM platform address is not 32-bit aligned. This drastically improves the SDMA scripts' efficiency since the same loop that transfers 32 bits at a time can be used regardless of the start and end addresses in the ARM platform memory space.

This unit structure and registers are described in Burst DMA Structure and Burst DMA Registers.

46.5.2.1 Burst DMA Structure

The burst DMA is essentially made up of a 36-byte FIFO, address registers, and a controlling state-machine. The 36-byte FIFO enables eight-word buffering with address alignment, and the state-machine manages clock adaptation when required.

The burst DMA is depicted in the figure below.

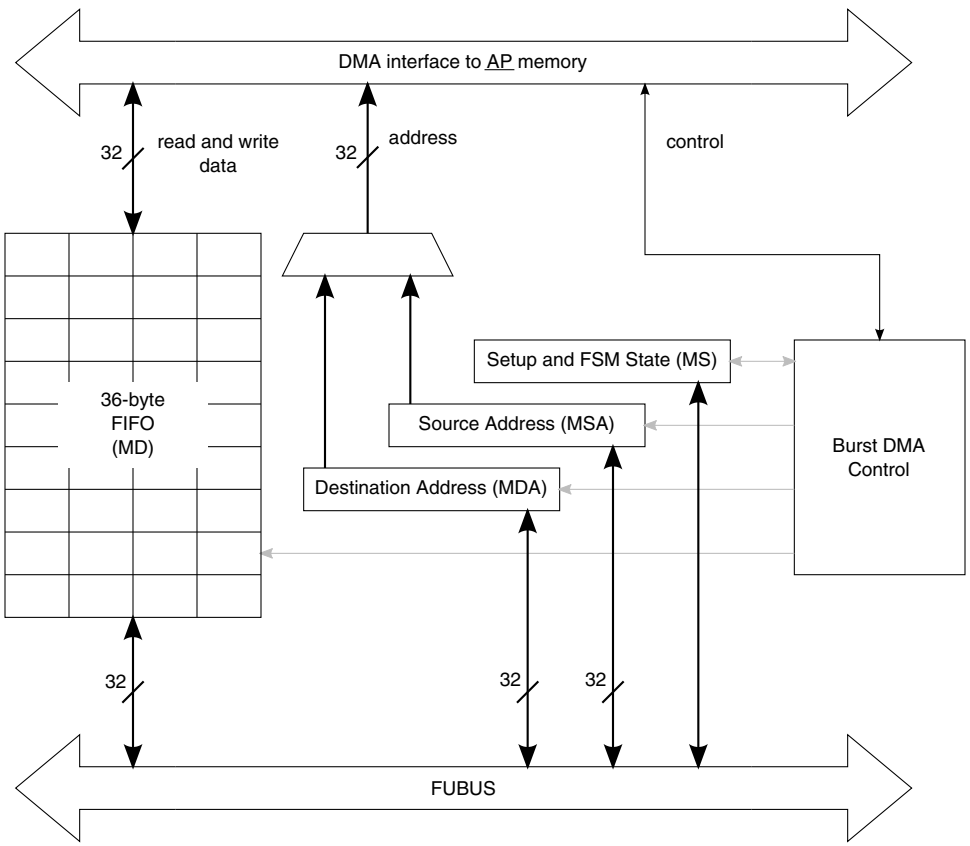


Figure 46-10. Burst DMA Structure

46.5.2.2 Burst DMA Registers

There are four registers, as follows, that may be accessed from the SDMA core:

- MSA (Memory Source Address) - Holds the source byte address in the ARM platform memory map for reading data from this location. This register is automatically modified every time the core reads new data from the FIFO.
- MDA (Memory Destination Address) - Holds the destination byte address in the ARM platform memory map for writing data to this location. This register is automatically modified every time the core writes new data into the FIFO.

- MD (Memory Data) - Labels the 36-byte FIFO access point: Reading a byte, halfword, or word from MD respectively retrieves the first 1, 2, or 4 bytes of the FIFO (for example, the bytes that were stored first by the DMA state-machine when transferring data from the ARM platform memory).
- When the FIFO does not hold as many bytes as required by the SDMA core, the core is stalled until the missing bytes are read from the ARM platform memory. In the case of prefetch mode, the DMA controller decides when it should start a burst to ARM platform memory in order to reduce the risk to not have the required data for the future accesses of the core. When there is no prefetching, a burst is triggered when the required data is not available in the FIFO.

  Writing a byte, halfword, or word to MD stores 1, 2, or 4 bytes, respectively, at the end of the FIFO (for example, these bytes are transmitted to the ARM platform memory after all the other bytes that were previously stored in the FIFO). When the FIFO does not have enough room left to hold the written data, the SDMA core is stalled until a sufficient amount of FIFO contents are flushed out to the ARM platform memory. Flushing is decided by the DMA controller when there are enough bytes in the FIFO to perform the largest allowed burst to ARM platform memory (the exact size depends on the burst start address and the AHB 1 Kbyte boundary rule). However, the SDMA core has the ability to force the flushing operation at any time, for example, when at the end of the data transfer, prior to channel closure.

- MS (Memory Setup) - Contains the state of the burst DMA control, the two flags that define whether each address register is incremented after every access to the external memory, and another flag that is set when a bus error occurred.

46.5.2.3 Burst DMA Data Transfers

Three typical usages have been identified that involve the burst DMA: the data transfer startpoint, the endpoint, or both.

Every case requires a different procedure, as listed in the following sections:

46.5.2.3.1 Data Retrieval from the ARM platform Memory

The following steps retrieve data from ARM platform memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the source address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the source address register itself (MSA).
- Read data from the FIFO using the *ldf MD* instruction as many times as needed. If an error occurred during the fetch from ARM platform memory, the DMA control tags

the error status on the data and the SDMA core SF flag is set when reading this data from the FIFO.

46.5.2.3.2 Storing Data Into the ARM platform Memory

The following steps store data from ARM platform memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the destination address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the destination address register itself (MDA).
- Store data into the FIFO using the *stf MD* instruction as many times as needed.
- When the transfer is finished and if the DMA worked in automatic flush mode, force the flush of the FIFO. This instruction is stalled until all the FIFO data is effectively sent to the ARM platform memory and the error status of the transfer is available in the DF flag.

46.5.2.3.3 Transferring Data Between Two ARM platform Memory Locations-Burst DMA Unit

The following steps copy data between two ARM platform memory locations using the burst DMA unit:

- Set up the MS flags to reflect the modes for the source and destination addresses (all the combinations are possible), then initialize the source address register (MSA) and the destination address register (MDA). Both addresses must be word-aligned.
- Use as many *stf MD* instructions with the *COPY* flag as needed. Every instruction triggers a burst read of a given number of words from the source address (this number is provided to the burst DMA via the SDMA core general purpose register, which is referenced in the *stf* instruction). Once all the data is loaded into the FIFO, the DMA empties it with a write burst of the same count to the destination address. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the burst DMA to check the error status.

46.5.3 Peripheral DMA Unit

The peripheral DMA unit is the second functional unit that connects the SDMA to the ARM platform memory.

Unlike the burst DMA, it does not support burst transfers and is optimized for accessing peripherals. It does not provide control to assign a privilege level to the DMA access. Its feature list comprises the following:

- Access to the ARM platform peripherals or memory at once or twice the SDMA core frequency
- Data copy from one ARM platform memory location to another ARM platform memory location at memory bus speed, improving throughput
- Control of the method for addressing the ARM platform memory (automatic increment or decrement of addresses or frozen addresses, the first ones aimed at accessing RAM-like memory and the last one aimed at accessing single-address FIFOs)
- Selectable automatic prefetch when reading data from the ARM platform memory. In prefetch mode, the peripheral DMA automatically fetches another data-without waiting for the SDMA core to request it-when its data register is empty, which improves the throughput
- Selectable automatic flush. In this mode, the SDMA core may only be stalled when it tries writing data and the previous write operation is not finished yet; whereas, in forced flush mode, the core is stalled until the data is effectively written to the ARM platform memory.
- In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the ARM platform memory or the peripheral. This error status is retrieved by a later access to the peripheral DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the ARM platform memory has been caught.

  This unit structure and registers are described in Peripheral DMA Structure and Peripheral DMA Registers.

46.5.3.1 Peripheral DMA Structure

The peripheral DMA is made up of a 32-bit data register, two address registers, and a controlling state-machine. The state-machine manages clock adaptation, when required.

It is shown in the following figure.

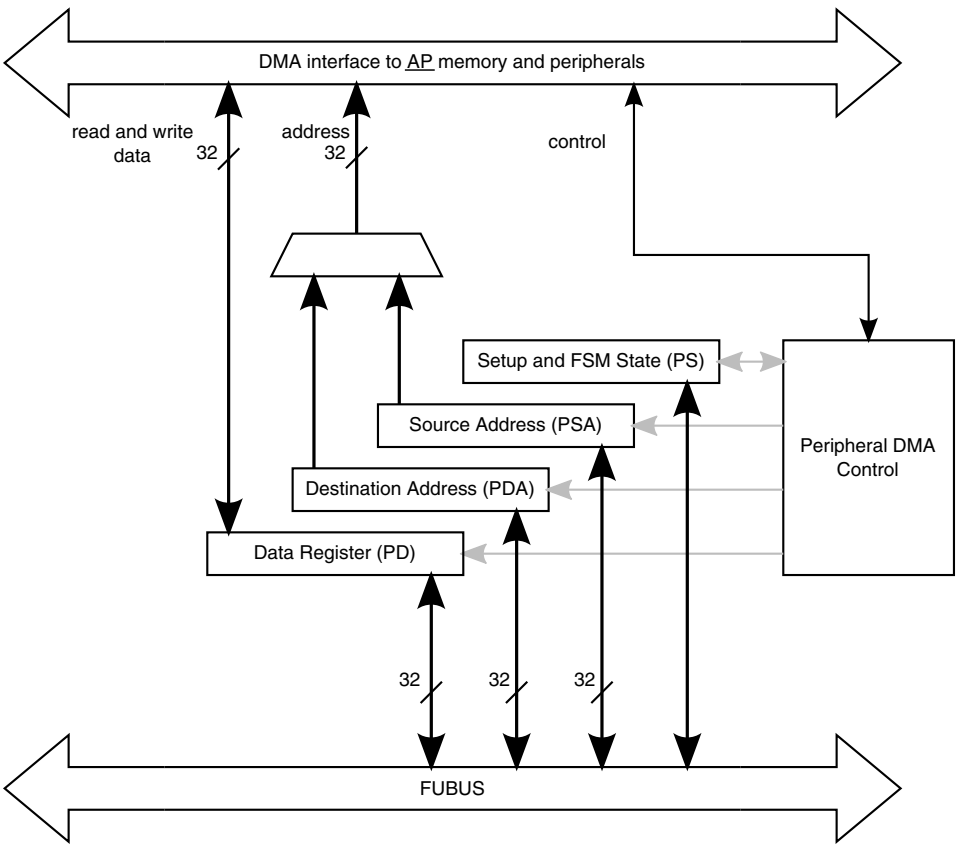


Figure 46-11. Peripheral DMA structure

46.5.3.2 Peripheral DMA Registers

According to Figure 46-11, the peripheral DMA has four registers that may be read or written by the SDMA core:

- *PD (Peripheral Data)* is the DMA 32-bit data register.
- *PSA (Peripheral Source Address)* holds the source byte address in the ARM platform memory map for reading data from this location. This register is automatically modified every time the core reads a new data from PD.

- *PDA (Peripheral Destination Address)* holds the destination byte address in the ARM platform memory map for writing data to this location. This register is automatically modified every time the core writes a new data into PD.
- *PS (Peripheral Setup)* contains the state of the peripheral DMA control, two configuration fields that define the way address registers are modified after every data access, two additional configuration fields that define the data size to access the source and destination devices, and another field that contains the latest transfer error status.

46.5.3.3 Peripheral DMA Data Transfers

There are three typical usages that involve the peripheral DMA, whether it is the data transfer start-point, endpoint, or both.

Every case requires a different procedure, as described in Data Retrieval from the ARM platform Memory or Peripheral, Storing Data into the ARM platform Memory or Peripheral, and Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit.

46.5.3.3.1 Data Retrieval from the ARM platform Memory or Peripheral

The following steps retrieve data from ARM platform memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the source (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the source address register itself (PSA) with an address that is aligned to the programmed data size.
- Read data from PD using the ldf PD instruction as many times as needed. If an error occurs during the fetch from the ARM platform memory or peripheral, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from PD.

46.5.3.3.2 Storing Data into the ARM platform Memory or Peripheral

The following steps store data to ARM platform memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the destination (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the destination address register itself (PDA) with an address that is aligned to the programmed data size.

- Store data into PD using the *stf PD* instruction as many times as needed.
- When the transfer is finished and if the peripheral DMA worked in automatic flush mode, force the flush of PD. This instruction is stalled until PD contents are effectively sent to the ARM platform memory or peripheral, and the error status of the transfer is available in the DF flag.

46.5.3.3.3 Transferring Data Between Two ARM platform Memory Locations-Peripheral DMA Unit

The following steps copy data between two ARM platform memory locations using the peripheral DMA unit:

- Set up the PS fields to reflect the modes and data size for the source and destination addresses (all the combinations of addressing modes are possible, but both data sizes must be identical), then initialize the source address register (PSA) and the destination address register (PDA). Both addresses must be aligned with the programmed data size.
- Use as many *stf PD* instructions with the *COPY* flag as needed. Every instruction triggers a single read from the source address; a single write of the received data immediately follows. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the peripheral DMA to check the error status.

46.6 SDMA Security Support

The SDMA provides support to SDMA software to block unauthorized updates to the scripts in RAM.

SDMA supports the following Security modes:

- Open Mode: has full control to load scripts and context into SDMA RAM. This is the default mode.
- Locked Mode: The ARM platform loads scripts and channel contexts at startup when it is still executing known safe software. When finished, it locks the SDMA to prevent further updates to RAM and selected registers. More details described in Locked Mode.

46.6.1 Locked Mode

The LOCK bit in the SDMA_LOCK register provides support for SDMA scripts to freeze RAM contents after the initial bootload routine to prevent future unauthorized updates to SDMA RAM.

After initial RAM contents are uploaded, ARM platform software can set the LOCK bit to secure the RAM contents to prevent future updates by an unauthorized. After the LOCK bit is written with a '1', the SDMA is "locked" until reset.

The LOCK bit can be read in the SDMA's internal memory map in the LOCK register (see Section SDMA LOCK (SDMAARM_SDMA_LOCK)). SDMA scripts which load information into RAM can check the value of the LOCK bit to determine if an upload to RAM is allowed. If not allowed, the script can refuse to allow the request to copy data into the RAM to continue. The exact use of the LOCK bit in SDMA scripts for security control will be described in SDMA software documentation (see SDMA Scripts).

While SDMA is locked, attempts to write to the SDMA_LOCK, CHN0ADR, ILLINSTADDR, and ONCE_ENB registers will be ignored. All registers remain readable. Writes to other registers are still allowed.

Once the SDMA is locked, the LOCK bit can only be cleared by a reset. A hardware reset will always clear the LOCK bit. A software reset initiated by writing to the RESET register will only clear the LOCK bit if the SRESET_LOCK_CLR bit in the SDMA_LOCK register is set. Since SDMA_LOCK register cannot be updated if SDMA is locked, the SRESET_LOCK_CLR bit must be configured before setting the LOCK bit. The SREST_LOCK_CLR bit will also be cleared by resets that clear the LOCK bit.

The SDMA RISC core uses the ILLINST and CHN0ADDR registers as pointers to determine where to jump to after an illegal instruction or upon boot after a reset. The LOCK bit prevents updates to these registers to protect against unauthorized changes to these pointers.

While SDMA is locked, the ONCE_ENB register cannot be written to prevent the OnCE under ARM platform control from being used to gain access to SDMA internal memory. If ARM platform control of the OnCE is enabled before setting the LOCK bit, the ARM platform can use the ONCE for debug purpose after LOCK is set.

46.7 OnCE and PCU Debug States

The SDMA has two different debug modes in which the OnCE performs debug instructions.

Refer to Figure 46-4 for an example of the PCU states in debug. The following are the two debug states:

- When a channel is running (that is, when CCR and CCPRI are different from 0, which can be read in the PSW register), SDMA can execute a SoftBkpt instruction from the channel script or receive a debug request. When either happens, the SDMA enters its "Classical" *Debug* state, which is described in OnCE and Real-Time Debug.
- When a channel is not running, the SDMA can be in *Sleep* state or in *Sleep after Reset* state. If a debug request is sent to the core, it enters its *Debug in Sleep* state. This debug mode works similarly to the "Classical" *Debug* state, except it returns to the original state (*Sleep* or *Sleep after Reset*) when the debug mode is left via the exec_core instruction of the OnCE. From this *Debug in Sleep* state, the SDMA can execute a program whereas no channel is running. If a new debug request is sent to the core or if a SoftBkpt is executed, it comes back to this *Debug in Sleep* state.

The OnCE is provided with several instructions that can be executed when the core is in either debug state. The following table summarizes the behavior of these OnCE debug instructions. There exists other secondary OnCE instructions that are described in OnCE and Real-Time Debug.

Table 46-6. SDMA in Debug Mode

Instruction	Debug	Debug in Sleep
exec_once	exec_once <instruction> SDMA executes the <instruction> and returns to the *Debug* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.	exec_once <instruction> SDMA executes the <instruction> and returns to the *Debug in Sleep* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.
run_core	run_core <instruction> SDMA executes the <instruction>, leaves the *Debug* state and continues executing the channel script from the position where it stopped. This command must not be used with an instruction that modifies the PC value.	run_core <instruction> SDMA executes the <instruction> and returns to its *Sleep* or *Sleep after Reset* initial state. This command must not be used with an instruction that modifies the PC value.
exec_core	exec_core <instruction> It is similar to run_core except it requires an instruction that changes the PC value (jump, branch...): the SDMA jumps to the new PC value, leaves the *Debug* state and starts executing instructions from this new PC value.	exec_core <instruction> If the previous state was *Sleep after Reset*, the SDMA returns to this state, and Chn0Addr value overrides the PC value. Otherwise, the SDMA jumps to the new PC value and starts executing instructions from this new PC.

NOTE

The feature exec_core in *Debug in Sleep* after *Sleep after Reset* was added for the Channel boot (channel 0) to allow the debugger to return to *Sleep after Reset* state with a new PC

value. The SDMA will be ready to boot at the Chn0Addr address.

46.8 SDMA Clocks and Low Power Modes

The SDMA receives several root clocks from the SoC clock controller block and performs adaptive clock gating to optimize its power consumption. From a user standpoint, clock gating and power mode selection are fully automatized inside the SDMA.

Root clock control is available from the SoC clock controller block.

There are numerous clock sources that are used in the SDMA. They belong to one of two possible clock domains listed in the following table, and have frequency constraints within each domain. Clocks are considered asynchronous between domains.

Within the ARM platform/SDMA clock domain, all clocks must come from the same DPLL. The ARM platform DMA interfaces (peripheral DMA and burst DMA) receive their clock from the ARM platform DMA clock source whose frequency can be once or twice the frequency of the SDMA core clock. The DMA interfaces are designed to work at the ARM platform DMA frequency, but the SDMA core is physically limited to a maximum 104 MHz frequency. Since this is lower than the maximum ARM platform DMA frequency, the SDMA core clock is tied to the ARM platform peripheral clock frequency.

The ARM platform Peripheral Bus Clock source must be an exact sub-frequency of the SDMA Core clock source (any integer value greater or equal to 1).

Table 46-7. Clocking Scheme

Clock Domain	Source Clock	Comments
ARM platform	SDMA core (SDMA main core)	Source clock for the core and all its operations; this clock is thus used by most of the SDMA sub-blocks.
	ARM platform DMA	DMA interface for the peripheral DMA and the burst DMA. It is balanced with the main clock source, and its frequency is either once or twice the main clock frequency.
	ARM platform peripheral	Connection to the ARM platform peripheral bus. It is a sub-frequency of the main clock frequency.
JTAG	TCK	Clock for JTAG access, limited to maximum of 1/8 of the SDMA core clock frequency.

The JTAG clock is sampled by the SDMA main clock to determine its rising edge. This simplifies design and clock management, but it also adds a ratio constraint between those two clocks. It is guaranteed the JTAG interface works properly when the frequency of TCK is lower than $1/8^{th}$ of the frequency of the SDMA main clock (which is about 8 MHz when the SDMA core clock frequency is 66 MHz).

46.8.1 Clock Gating and Low Power Modes

The SDMA automatically performs power saving without requiring user involvement. It implements two levels of automatic clock gating.

46.8.1.1 Coarse Clock Gating

Every sub-block clock comes from one of the five available sources, and is gated with the sub-block specific enabling condition.

The following table displays the sub-block clocks and their source. It also indicates the relationships that may exist between different sub-blocks clock enables.

Table 46-8. Sub-blocks Clocks

Sub-block	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
Core	SDMA Main Core	The core sub-block clock is running when the core is not in one of its sleep states (Sleep or Sleep after Reset) or there is a pending channel. Typically, the core sub-block clock is stopped once all the channels are processed and the core enters its sleep state. A new pending channel awakes the core sub-block clock.	None
Memories	SDMA Main Core	The clock activation only occurs during a core access.	Disabled when Core sub-block clock is disabled or no memory access in progress
Scheduler	SDMA Main Core	Its clock only runs when scheduling is needed: for example, when there are pending channels, upon reception of a DMA request, and anytime the ARM platform modifies the channel running conditions.	None
ARM platform Control	SDMA Main Core & ARM platform peripheral	The ARM platform peripheral clock is solely used to determine the frequency ratio with the SDMA main clock. The control registers' clock is based on *SDMA main clock*; it is active when the ARM platform or the SDMA modifies the contents of one of these registers.	None
CRC	SDMA Main Core	The CRC clock is based on *SDMA main clock* and is only active during data processing.	Disabled when Core sub-block clock is disabled

Table continues on the next page...

Table 46-8. Sub-blocks Clocks (continued)

Sub-block	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
Burst DMA	SDMA Main Core & ARM platform DMA	The burst DMA has two clocks: The first clock is derived from the SDMA main core clock and drives registers that are connected to the FUBUS. The second clock is derived from the ARM platform DMA clock and drives registers that are connected to the ARM platform DMA bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the burst DMA is not used by the running channel script).	Disabled when Core sub-block clock is disabled
Peripheral DMA	SDMA Main Core & ARM platform DMA	The peripheral DMA has two clocks: The first clock is derived from SDMA main clock and drives registers that are connected to the FUBUS. The second clock is derived from the ARM platform DMA clock and drives registers that are connected to the ARM platform DMA bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the peripheral DMA is not used by the running channel script).	Disabled when Core sub-block clock is disabled
OnCE	SDMA Main Core	The OnCE clock is derived from main source clock. It is disabled by default. In order to use the OnCE, its clock must be explicitly turned on, either by enabling the OnCE access from the ARM platform peripheral bus (register ONCE_ENB), or by driving the clk_gating_off input pin high. This is a SDMA input whose driver depends on the SoC implementation (typically a JTAG controller). The OnCE also receives the TCK input, which is the JTAG clock. It does not use it as a functional clock; the TCK input is sampled instead. Refer to Synchronization Implementation.	When enabled, all other clocks are systematically on (clock gating is off)

46.8.1.2 Refined Clock Gating

The SDMA implements a second level of clock gating on a register-per-register basis.

Unlike the first level that covers all the SDMA flip-flops, except the synchronizers (only five flip-flops are always running), the second level is only available for eligible registers, which amounts to about 90% of the SDMA flip-flops.

These gated registers are only clocked when the hardware logic detects a new data loading. This additional gating further reduces dynamic power consumption.

46.8.1.3 Low Power Modes and User Control

Power savings are automatically managed by the SDMA hardware without any user involvement; however, one can distinguish three different power modes: SLEEP, RUN, and DEBUG.

The following table describes these modes, and shows how to switch from one mode to another.

Table 46-9. Power Modes

Power Mode	Sub-blocks								Comments
	Core	Memories	Scheduler	ARM platform Control	CRC	Burst DMA	Peripheral DMA	OnCE	
SLEEP	off[1]	off	wait[2]	wait	off	off	off	off	Set when the PCU state is either *Sleep* or *Sleep after Reset* and the SDMA is not in DEBUG mode. This is the default mode after reset.
RUN	on[3]	wait	wait	wait	wait	wait	wait	off	Set for the other PCU states that are reachable out of debug: *Program*, *Data*, *Change of Flow*, *Error in Loop*, *Debug*, *Functional Unit*, *Save,* or *Restore*.
DEBUG	on	on	on	on	on	on	on	on	Set regardless of the PCU state when clock gating is turned off to use the OnCE features (either *clk_gating_off* pin high or ONCE_ENB[0] set).

1. *off*: no clock
2. *wait*: only clocked when accessed or stimulated
3. *on*: clock is always running

It is possible to control the SDMA power mode. The procedures to force the SDMA into either mode are described in SLEEP Mode.

46.8.1.3.1 SLEEP Mode

This is the default mode after reset; therefore, resetting the SDMA forces this mode.

However, the common procedure is as follows:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Disable all channels (via the STOP_STAT control register, and the HO, DO, EO if necessary).
- Wait for the active channels to complete or force a reschedule via the reschedule bit in the RESET register.
- The SDMA is in SLEEP mode making it possible to completely shut off its clock from the chip level clock controller using the procedure described in Stop Mode Response.

46.8.1.3.2 RUN Mode

This is the default mode when a channel is running:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Activate at least one channel (via the HSTART control registers, a DMA request, and/or the HO, DO, EO register bits).

46.8.1.3.3 DEBUG Mode

The DEBUG mode must be set when one needs to use the debugging facilities of the SDMA.

- Ensure the SDMA clocks are running from the CCM.
- Set the *clk_gating_off* pin high or use the SDMA to set ONCE_ENB[0].

46.8.1.4 Stop Mode Response

The SDMA receives a stop request from the chip level clock controller. This request may be asserted when the chip enters the stop low power mode.

If the SDMA is running when the request is received, then the SDMA will complete all pending channels before returning to the SLEEP state. The SDMA sends an acknowledgement to the clock controller when the SLEEP state is entered indicating that the SDMAs clocks can be turned off.

46.8.2 Reset

After reset (either received from the reset block or a software reset required by the ARM platform), the SDMA is in IDLE mode. It will start its boot code located at address 0 once a channel is activated.

Activating a channel can be done by the ARM platform after programming a positive priority and setting the channel bit in the EVTPEND register.

There will not be a context RESTORE for the first channel (bootload channel) called after a reset because the context data in RAM has not been initialized. Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized. Subsequent calls to the same channel or different channels may use any of the dynamic context modes

46.9 Software Interface

Appendix A fully describes the SDMA Application Programming Interface (API).

46.10 Initialization Information

This section discusses the following:

- Hardware Reset
- Channel Script Execution
- Initialization and Script Execution Setup Sequence

46.10.1 Hardware Reset

After reset, the program RAM, context RAM, data RAM, and RAM containing the channel enable registers (CHNENBLn) have unpredictable contents.

The active register set is assigned to channel 0 and the PC is initialized to all zeros. However, since the channel enable register is all zeros, there are no active channels and the SDMA is halted waiting for the boot channel to start.

The ARM platform will have to setup the SDMA in order to boot it. The CONFIG register must be initialized to determine the DMA/core clock ratio (1 or 2). Channel Enable Registers must also be initialized.

To start up the SDMA, the ARM platform first creates some channel control blocks (CCB) and buffer descriptors (BD) in ARM platform memory for the boot channel (channel 0) and then initializes the channel 0 pointer register (SDMA_MC0PTR) to the address of the first control block. Data Structures for Boot Code and Channel Scripts provides an overview of the data structure for the CCB and BD's. The SDMA_HSTART, SDMA_HOSTOVR and SDMA_EVTOVR registers are then configured according to Runnable Channels Evaluation to allow channel 0 to run.

Upon being enabled, the SDMA begins executing the script located at the address indicated by the Channel 0 Boot Address register (SDMA_CHN0ADDR) in the program memory. The reset value of SDMA_CHN0ADDR points to the default bootload script in ROM. This ROM script will read the channel 0 pointer register (SDMA_MC0PTR) to determine the location of the Channel Control Block (SDMA_CCB) in ARM platform memory. The script will then begin fetching by DMA the first channel control block which contains a pointer to the location channel 0 Buffer Descriptor chain which is also fetched via DMA. If the buffer descriptor contains a valid command, the script interprets the command in each buffer descriptor and proceeds to implement the command and move on to the next buffer descriptor control block. The buffer descriptor commands for

channel zero are typically set up to load SDMA's program RAM, Data RAM, and initial values for the channel contexts. Some channel scripts expect particular parameters to be passed

There are two ways to make the SDMA boot on a user-defined script. The OnCE (either via its JTAG interface or its ARM platform Control interface) can be used to download any code in the SDMA RAM and force the SDMA to boot on that code. Also, the SDMA_CHN0ADDR register in the ARM platform programming model can be modified to point to user code in RAM which would need to either have been loaded via the ONCE or default bootload routine (ex before a S/W reset).

46.10.2 Channel Script Execution

The execution of an SDMA script depends on both the instructions that make up the script, the data context upon which it operates, and commands or parameters passed in the buffer descriptor or. All these items must be initialized before the script is allowed to execute.

Each of the 32 channels has a separate context, but may share scripts and locations in data RAM.

Channels are initialized by the ARM platform by using channel 0 to download any required scripts and data values and the channels initial context. The context contains all the initial values of the SDMA core registers. This includes the Program Counter (PC) which is set to the start of the desired script in SDMA program memory.

The ARM platform selects which trigger conditions that must occur for the channel to start by configuring the SDMA_CHNENBL, SDMA_HOSTOVR and SDMA_EVTOVR registers. The trigger events include ARM platform setting HE (SDMA_HSTART) or a hardware DMA request asserts an event input to SDMA. The channel can become active according to its priority compared with other runnable channels when the selected trigger(s) cause the condition described in Runnable Channels Evaluation to evaluate as true.

The specific parameters to be passed to each script in the buffer descriptor or context are documented in the software documentation for each script. Please refer to SDMA Scripts for complete script documentation. Buffer Descriptor Format provides an overview of the buffer descriptor format.

46.10.3 Initialization and Script Execution Setup Sequence

To summarize, the following steps are minimally required to setup SDMA and run channel scripts.

- Perform Hardware Reset. The program RAM, context RAM, data RAM and SDMA_CHNENBLn registers have unpredictable contents after this reset.
- Initialize SDMA_CHNENBLn registers to map DMA request events to desired channels.
- Configure SDMA_CHNPRIn registers to select priority for runnable channels, A non-zero priority is required for the channel to run.
- Configure the SDMA_CONFIG register to select DMA to SDMA core clock ratio .
- Set up channel control blocks and buffer descriptors in ARM platform to specify the loading of SDMA program RAM and channel contexts for each SDMA channel to be used. Reference Data Structures for Boot Code and Channel Scripts.
- Configure SDMA_MC0PTR register with base address of ARM platform Channel Control Block base address.
- Initialize SDMA_CHNENBLn registers to map DMA request events to associated channel. Reference Mapping DMA Requests to Pending Channels.
- Configure SDMA_CHNPRIn registers to set priority for each channel to be run.
- For each channel to be run, configure SDMA_HOSTOVR (HO) and SDMA_EVTOVR (EO) registers to select which events (hardware and/or software trigger events) must occur for the channel to be runnable. Reference Runnable Channels Evaluation.
- Set bit 0 of the SDMA_HSTART register to set HE[0] and allow Channel 0 to run (assumes EO[0] andDO[0] were both set in previous step). This will cause SDMA to load the program RAM and channel contexts configured previously.
- Wait for Channel 0 to finish running. This is indicated by HI[0]=1 in the SDMA_SDMA_INTR register, or by optional interrupt to the ARM platform.
- Set the LOCK bit in the SDMA_SDMA_LOCK register to prevent un-authorized uploads of data to SDMA RAM.
- Additional channel scripts can now be run by enabling the selected software or hardware trigger event according to Runnable Channels Evaluation.

46.11 SDMA Programming Model

The following section describes the programming model for the SDMA RISC engine, including its processor, memory, and internal control registers.

All addresses are related to the internal SDMA memory map, which is completely different from the ARM platform memory maps. The ARM platform processor has no access to any hardware resource described, except when those resources are described in ARM Platform Memory Map and Control Register Summary. .

46.11.1 State and Registers Per Channel

The SDMA can be seen as a set of 32 identical devices that are able to perform one data transfer channel each. Only one channel can work at a time, but every channel state is available at any time.

This chapter lists the components of every channel state.

46.11.2 General Purpose Registers

Each channel has eight general purpose registers of 32 bits for use by scripts. General register 0 has a dedicated function for the loop instruction, but otherwise can be used for any purpose.

46.11.3 Functional Unit State

Each channel context has some state that is part of the functional units.

The specific allocation of this state is part of the functional unit definition that is described in Burst DMA Unit Programming, Peripheral DMA Unit Programming .

This state must be saved/restored on context switches.

46.11.3.1 Program Counter Register (PC)

The PC is 14 bits. Since instructions are 16 bits in width and all memory in the SDMA is 32 bits in width, the low order bit of the PC selects which half of the 32-bit word contains the current instruction.

A low order bit of zero selects the most significant half of the word.[1]

1. For example, big-Endian.

46.11.3.2 Flags

Each channel has the following four flags:

- The T bit reflects the status of some arithmetic and test instructions. It is set when the result of an addition or a subtraction is zero and cleared otherwise. It is also the copy of the tested bits. Finally, it can also be set when the loop counter (GReg0) reaches zero. When the last instruction of the hardware loop is an operation that can modify the T flag, its effect on T is discarded and replaced by the GReg0 status.
- Two additional bits, SF and DF, are used to indicate error conditions resulting from loading data sources and storing to destinations, respectively. Access errors set these bits, and successful transactions clear them. They can also be cleared by specific instructions (CLRF and loop). The source fault (SF) is updated by the loads LD and LDF; the destination fault (DF) is updated by the stores ST and STF.
- Access errors are caused by several conditions including writing to the ROM, writing to a read-only memory mapped register, accessing an unmapped address, or any transfer error received by a peripheral when it is accessed.

  The SF and DF flags have a major impact on the behavior of the hardware loop: If SF or DF is set when starting a hardware loop and it is not masked by the loop instruction, the loop body will not be executed. Inside the loop body, if a load or store sets the corresponding SF or DF flag, the loop exits immediately. Testing the status of the T flag at the end of the loop (as well as testing both SF and DF) tells if the loop exited abnormally as any anticipated exit prevents GReg0 from reaching the zero value and thus setting the T flag. This is also valid if the fault occurs at the last instruction of the last loop.

- The last flag is the loop mode flag, LM, which is composed of two bits. The most significant bit indicates when the processor is currently operating in loop mode. It is set by the loop instruction and is cleared after execution of the last instruction of the last loop. The least significant bit is set when the program counter points to the last instruction of a loop on the last path. It is used for a channel that is restored with this configuration to know that the next program counter is EPC. As with the dynamic context switch Greg0, which indicates when the program must get out of the loop, it can be restored only on the last instruction of the loop. This, however, is too late to fetch the next instruction after the loop.

46.11.3.3 Return Program Counter (RPC)

The RPC is 14 bits. It is set by the jump to the subroutine instructions and used by the return from the subroutine instructions.

Instructions are available to transfer its contents to and from a general register.

46.11.3.4 Loop Mode Start Program Counter (SPC)

The SPC is 14 bits. It is set by the loop instruction to the location immediately following it.

46.11.3.5 Loop Mode End Program Counter (EPC)

The EPC is 14 bits. It is set by the loop instruction to the location of the next instruction after the loop.

46.11.4 Context Switching-Programming

Each channel has a separate context consisting of the eight general purpose registers and additional registers representing the state of the functional units.

The active registers and functional units contain the context of the active channel. The context of inactive channels is stored in SDMA RAM, which is part of the SDMA address space.

In a function of the selected context switching mode (Context Switching), modified registers by the program can be saved in the channel RAM space while the program is going on. In every cycle, a write access to the RAM is possible.

On a done or yield(ge) instruction, SDMA goes into "real" context switching. In one of the dynamic modes, modified registers not previously saved, as well as the PC-Loop registers, are stored into the context area of the channel that will be closed. The new PC-Loop registers are loaded from the context area of the new channel. All other registers are restored while the program is executed, giving priority to registers used by the decoded instruction. Therefore, in the best case, only the PC and Loop registers should be saved and restored during this context-switching phase, which only requires five SDMA cycles.

In static mode, the context switch stores all registers in the old channel RAM space, and restores all registers from the new channel RAM space. It requires 26 SDMA cycles.

The address of the context memory for channel *i* is *CONTEXT_BASE* + 24**i* or *CONTEXT_BASE* + *32*i w*here CONTEXT_BASE equals 0x0800. The table below presents the layout of a channel context in memory:

Table 46-10. Layout of a Channel Context in Memory for SDMA

OFFSET	31	30	29-16	15	14	13-0

Table continues on the next page...

Table 46-10. Layout of a Channel Context in Memory for SDMA (continued)

0	SF	-	RPC	T	-	PC
1	LM		EPC	DF	-	SPC
2	GR0					
3	GR1					
4	GR2					
5	GR3					
6	GR4					
7	GR5					
8	GR6					
9	GR7					
10	MDA (burst DMA)					
11	MSA (burst DMA)					
12	MS (burst DMA)					
13	MD (burst DMA)					
14	PDA (peripheral DMA)					
15	PSA (peripheral DMA)					
16	PS (peripheral DMA)					
17	PD (peripheral DMA)					
18	CA (CRC)					
19	CS (CRC)					
20	Reserved[1]					
21	Reserved[1]					
22	Reserved[1]					
23	Reserved[1]					
24	Scratch RAM (optional)					
25	Scratch RAM (optional)					
26	Scratch RAM (optional)					
27	Scratch RAM (optional)					
28	Scratch RAM (optional)					
29	Scratch RAM (optional)					
30	Scratch RAM (optional)					
31	Scratch RAM (optional)					

46.11.5 Address Space

The SDMA has four internal buses, as follows:

- The Instruction bus reads instructions from the memory. Its address map is described in Instruction Memory Map.
- The Data bus (DMBUS) accesses the same memories as those visible on the Instruction bus, some memory-mapped registers (scheduler status and OnCE registers), and up to 14 peripherals. Its address map is described in Data Memory Map.
- The Functional Units bus (FUBUS) accesses the burst DMA, peripheral DMA, and CRC internal registers. The addressing mechanism is further detailed in Functional Units Programming Model.
- The Context Switch bus reads/writes registers into context-switch RAM space. It is a 64-bit bus dedicated for accessing this RAM space for updating the context of the running channel. While the program is going on, this bus has the lowest priority compared to the Instruction and Data buses, except for restoring a register needed for the decoded instruction to be executed. On the save part of a context switch (when the PCU is in its slave state), this is the only one used. On the restore part, the Instruction bus has the priority to read the next instruction at the restored PC and otherwise the Context Switch bus is used. It is not possible to control the actual data transfers that occur on this bus.

46.11.5.1 Instruction Memory Map

The instruction memory map is based on a 14-bit address bus and a 16-bit data (instruction) bus. Each address corresponds to a 16-bit data location.

Instructions are fetched from either program ROM or program RAM. An SDMA script is able to change the contents of the program RAM, which is also visible from the data bus.

The first two instruction locations (at 0 and 1) are special. Location 0 is where the PC is set on reset. Location 1 is where the PC is set upon the execution of an illegal instruction. It is expected that both of these locations will contain a jmp to handle routines.

Table 46-11. SDMA Instruction Memory Space

Device	SDMA Address (Hex)	Base Address Label	Block Name	WS	Description
ROM	0x0000 ↓ 0x07FF	SDMA_IBUS_ROM_ADDR	-	0	4 Kbyte internal ROM with boot code and standard routines.
RAM	0x1000 ↓ 0x1FFF	SDMA_IBUS_RAM_ADDR	-	0	8 Kbyte internal RAM with channels context and user data/routines.

46.11.5.2 Data Memory Map

All of the data accessible to SDMA scripts make up the data memory space of the SDMA.

This address space has several components:

- ROM (also visible on the Instruction bus)
- RAM (also visible on the Instruction bus)
- Shared Peripherals Registers
- SDMA Internal Registers (scheduler, OnCE, and registers that are also accessible by the ARM platform)

SDMA scripts can read and write to the context RAM, data RAM, shared peripheral registers, and internal registers.

The address range is 16 bits and the data width is 32 bits. Each address corresponds to a 32-bit data word. When accessing peripheral registers (USB and so on), the data width may be different. The exact address map for the peripherals depends on the project (as presented in each respective chapter).

The SDMA can perform only 32-bit access to shared peripheral registers. For this device, shared peripherals registers are aliased as if byte addressed. This is a consequence of connections through the SPBA shared peripheral bus outside of the SDMA. The result is that although address space 4 Kwords (16Kbyte) is allocated for each peripheral, only the first 4 Kbyte of the peripheral's register space can accessed. For example, the shared peripheral register at address 0x3000 is mapped also to addresses 0x3001, 0x3002, 0x3003. A read or write access to any of any of these 4 addresses will respond as if the access was to address 0x3000.

Data access is performed with *ld* and *st* instructions that take the address from a general purpose register in the core (GReg*n*). The mapping between the general purpose register contents and the address bus is given in the following table:

Table 46-12. GReg*n* to DMBUS Address Mapping

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

Grayed bits are simply discarded but they must be cleared to ensure forward-script compatibility.

- sz (bit 31) indicates the peripheral data width: 0 is used for a 32-bit peripheral and 1 is used for a 16-bit peripheral.
- address (bits 15 down to 0) is the address of the accessed resource (internal memory, internal register, or shared peripheral).

Table 46-13. SDMA Data Memory Space

Device	SDMA Address (Hex)	Size	Description
ROM	0x0000 → 0x03FF	4 Kbyte	4 Kbyte internal ROM with boot code and standard routines
Reserved	0x0400 → 0x07FF	4 Kbyte	4 Kbyte Reserved
RAM	0x0800 → 0x0FFF	8 Kbyte	8 Kbyte internal RAM with channels contexts and user data/routines
per1	0x1000 → 0x1FFF	16 Kbyte	*peripheral 1* memory space (4 Kbyte peripheral's address space)
per2	0x2000 → 0x2FFF	16 Kbyte	*peripheral 2* memory space (4 Kbyte peripheral's address space)
per3	0x3000 → 0x3FFF	16 Kbyte	*peripheral 3* memory space (4 Kbyte peripheral's address space)
per4	0x4000 → 0x4FFF	16 Kbyte	*peripheral 4* memory space (4 Kbyte peripheral's address space)
per5	0x5000 → 0x5FFF	16 Kbyte	*peripheral 5* memory space (4 Kbyte peripheral's address space)
per6	0x6000 → 0x6FFF	16 Kbyte	*peripheral 6* memory space (4 Kbyte peripheral's address space)
Registers	0x7000 → 0x7FFF	16 Kbyte	Memory mapped registers
per7	0x8000 → 0x8FFF	16 Kbyte	*peripheral 7* memory space (4 Kbyte peripheral's address space)
per8	0x9000 → 0x9FFF	16 Kbyte	*peripheral 8* memory space (4 Kbyte peripheral's address space)
per9	0xA000 → 0xAFFF	16 Kbyte	*peripheral 9* memory space (4 Kbyte peripheral's address space)
per10	0xB000 → 0xBFFF	16 Kbyte	*peripheral 10* memory space (4 Kbyte peripheral's address space)
per11	0xC000 → 0xCFFF	16 Kbyte	*peripheral 11* memory space (4 Kbyte peripheral's address space)
per12	0xD000 → 0xDFFF	16 Kbyte	*peripheral 12* memory space (4 Kbyte peripheral's address space)
per13	0xE000 → 0xEFFF	16 Kbyte	*peripheral 13* memory space (4 Kbyte peripheral's address space)
per14	0xF000 → 0xFFFF	16 Kbyte	*peripheral 14* memory space (4 Kbyte peripheral's address space)

46.12 SDMA Initialization

Appendix A describes the setup of the SDMA . This section provides a quick description of several initialization procedures.

NOTE

There may be differences with the actual implementation in the API.

46.12.1 Hardware Reset-SDMA

After reset, the RAM that holds contexts, data, scripts, and the DMA request-channels matrix has unpredictable content.

The core registers are all reset to 0, including the PC; the PCU state is *Sleep after Reset*. No channel can be activated because all of the priorities are also reset to 0.

46.12.2 Standard Boot Sequence

The following is the standard boot sequence:

1. Initialize the CONFIG register-detailed in Configuration Register (SDMAARM_CONFIG)-to determine the ARM platform DMA/core clock ratio (1 or 2)
2. Initialize the DMA request-channels matrix (seeChannel Enable RAM (SDMAARM_SDMA.CHNENBL*n*)).
3. Program the channel control registers-Channel Event Override (SDMAARM_EVTOVR), Channel BP Override (SDMAARM_DSPOVR), Channel BP Override (SDMA_HOSTOVR), and Channel Event Pending (SDMAARM_EVTPEND)-according to the channel allocation.
4. Perform any necessary setup as required by the standard boot script in ROM (this is described in Appendix A).
5. Trigger channel 0 with the Channel Start (SDMAARM_HSTART) register, which starts the execution of the ROM script starting at address 0. This boot downloads channel scripts and contexts in RAM.

46.12.3 User-Defined Boot Sequence

The following is a user-defined boot sequence:

1. Initialize the Configuration Register (SDMAARM_CONFIG)Channel Enable RAM (SDMAARM_SDMA.CHNENBL*n*), Channel Event Override (SDMAARM_EVTOVR), Channel BP Override (SDMAARM_DSPOVR), Channel ARM platform Override (SDMAARM_HOSTOVR), and Channel Event Pending (SDMAARM_EVTPEND).
2. Use the OnCE (either via its JTAG interface or its ARM platform control registers) to download any code in the SDMA RAM. Accessing the Memory describes how to write data to the RAM via the OnCE.
3. Use the OnCE instructions to make the PC default value point to the new boot script start address, or rely on the ROM startup script, which first jumps to the address in

Channel 0 Boot Address (SDMAARM_CHN0ADDR). (This register default address points to the standard boot script.)

46.12.4 Script Loading and Context Initialization

The execution of an SDMA script depends on both the instructions that make up the script and the data context upon which it operates. Both must be initialized before the script is allowed to execute.

Each of the 32 channels has a separate data context, but may share scripts and locations in the data RAM.

The ARM platform manages the space in program RAM and data RAM. It also manages the assignment of SDMA channels to the device drivers that need them. Channels are initialized by the ARM platform via the channel 0 boot script. The boot channel downloads any required scripts with their data and the channels' initial contexts. Every context contains all the initial values of the registers, including the PC. Then the ARM platform can enable any channel that becomes active and begins fetching and executing instructions from its script.

46.13 Instruction Description

The following sections introduce the instruction of the SDMA. Instruction set details are available in Instruction Set.

46.13.1 Scheduling Instructions

The following are scheduling instructions:

- done-The instruction causes certain scheduling or interrupt bits to be set or cleared, which may cause a change in the schedule-ability of the running channel. Then the instruction causes the SDMA to evaluate the current scheduling priorities and to choose the highest priority ready channel. If this channel is not the current channel, a context switch will take place. If there are no runnable channels, the SDMA will enter the stopped mode. The done 5 has a special usage reserved for debug, as explained in Debug Instructions.
- yield-These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater than the current channel priority.

- yieldge-These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater or equal to the current channel priority.
- notify-The notify instruction affects the scheduling bits, but does not cause rescheduling.

46.13.2 Conditional Branch Instructions

The conditional branch instructions of an 8-bit displacement, which is sign-extended and added to the current PC (which points to the next instruction) if the condition is satisfied.

Otherwise, control passes to the next sequential instruction.

- BF-Branch if False. The branch is taken if the T bit in the processor status is zero (false).
- BT-Branch if True. The branch is taken if the T bit in the processor status is one (true).
- BSF-Branch if Source Fault. The branch is taken if the SF bit in the processor status is one.
- BDF-Branch if Destination Fault. The branch is taken if the DF bit in the processor status is one.

46.13.3 Unconditional Jump Instructions

There are two varieties of unconditional control transfers: an absolute transfer and a through-register transfer.

Absolute transfers have a 14-bit address field that replaces the current PC.

- JMP-Jump. Causes the processor to jump to an absolute address encoded in the instruction itself.
- JSR-Jump to Subroutine. Causes the processor to jump to a subroutine, the address of which is encoded in the instruction itself.
- JMPR-Jump through Register. Causes the processor to jump to an absolute address contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.
- JSRR-Jump to Subroutine through Register. Causes the processor to jump to a subroutine, the address of which is contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.

46.13.4 Subroutine Return Instructions

The following are subroutine return instructions:

- RET-Return from Subroutine. The RET restores the contents of RPC to PC.
- LDRPC-Load from RPC to Register. THe LDRPC instruction is meant to be used when more than one level of subroutines are required. It stores the contents of RPC in any General register.

46.13.5 Loop Instruction

The following is a loop instruction:

LOOP-Enters Loop Mode. Before entering loop mode, the loop instruction can optionally clear the fault flags (SF and/or DF) based on a 2-bit field in the instruction. This feature is linked to the fact that setting SF or DF in loop mode will cause an immediate exit of the loop.

46.13.6 Miscellaneous Instructions

The following are miscellaneous instructions:

- CLRF-Clear Fault Flags. This instruction clears any combination of SF and DF.
- MOV r,s-This moves data from GReg[s] to GReg[r].
- LDI r,immediate-This loads GReg[r] with a zero-extended immediate value.

46.13.7 Logic Instructions

The following are logic instructions:

- XORr,s-This performs an exclusive or between GReg[r] and GReg[s], and stores the result in GReg[r].
- XORIr,immediate-This performs an exclusive or between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].
- ORr,s-This performs an or between GReg[r] and GReg[s], and stores the result in GReg[r].
- ORIr,immediate-This performs an or between GReg[r] and a zero-extended immediate value and, stores the result in GReg[r].

- ANDNr,s-This performs an and between GReg[r] and the negated GReg[s], and stores the result in GReg[r].
- ANDNIr,immediate-This performs an and between GReg[r] and the negated zero-extended immediate value, and stores the result in GReg[r].
- ANDr,s-This performs an and between GReg[r] and GReg[s], and stores the result in GReg[r].
- ANDIr,immediate-This performs an and between GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].

46.13.8 Arithmetic Instructions

Arithmetic instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the result is zero, otherwise it is cleared.

- ADD r,s-This performs the addition of GReg[r] and GReg[s], and stores the result in GReg[r].
- ADDI r,immediate-This performs the addition of GReg[r] and a zero-extended immediate value, and stores the result in GReg[r].
- SUB r,s-This performs the subtraction of GReg[s] from GReg[r], and stores the result in GReg[r].
- SUBIr,immediate-This performs the subtraction of a zero-extended immediate value from GReg[r], and stores the result in GReg[r].

46.13.9 Compare Instructions

Compare instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the comparison is true, otherwise it is cleared.

NOTE

Only one version of the immediate form is implemented. Non-equality comparisons to immediate values will require two instructions.

- CMPEQ r,s-This sets T when registers GReg[r] and GReg[s] are equal.
- CMPEQIr,immediate-This sets T when register GReg[r] and the zero-extended immediate value are equal.
- CMPLTr,s-This sets T when register GReg[r] is less than and not equal to GReg[s]. The comparison is signed.
- CMPHS r,s-This sets T when register GReg[r] is greater than or equal to GReg[s]. The comparison is signed.

46.13.10 Test Instructions

Test instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if any bit in the result is one, otherwise it is cleared.

- TSTr,s-This performs an and between GReg[r] and GReg[s], and sets T if the result is not zero.
- TSTIr,immediate-This performs an and between GReg[r] and a zero-extended immediate value, and sets T if the result is not zero.

46.13.11 Byte Permutation Instructions

These instructions shuffle the bytes in a register. For the purpose of describing these instructions, have the bytes in a register be numbered from the most significant as b_3, b_2, b_1, b_0.

- RORBr-The rotate right byte. The result is b_0, b_3, b_2, b_1.
- REVBr-The reverse bytes in word. The result is b_0, b_1, b_2, b_3.
- REVBLOr-The reverse, two low-order bytes. The result is b_3, b_2, b_0, b_1.

46.13.12 Bit Shift Instructions

The following are bit shift instructions:

- ROR1r-The rotate right 1 bit. This instruction does a circular right shift of 1 bit.
- LSR1r-The logical shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by a 0.
- ASR1r-The arithmetic shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by itself.
- LSL1r-The logical shift left 1 bit. This instruction shifts all bits to the left by 1. The low order bit is replaced by zero.

46.13.13 Bit Manipulation Instructions

- BCLRIr,n-The bit clear is immediate; clears bit number *i* in register *r*.
- BSETIr,n-The bit set is immediate; sets bit number *i* in register *r*.
- BTSTIr,n-The bit test is immediate; tests bit number *i* in register *r* (T becomes equal to the selected register bit).

46.13.14 SDMA Memory Access Instructions

All memory accesses are 32 bits.

Any memory location that is implemented with less than 32 bits (for example, peripheral registers) causes unimplemented bits to be read as 0s.

All memory accesses will cause either the SF or DF flags in the processor status to be set if they cause a fault.

What constitutes a fault, especially when accessing peripheral registers, is a property of the memory location.

- LDr,(b,d)-The load instruction creates an address by adding the displacement field (d) to the contents of the base register (b). The SDMA location at the resulting address is read and placed in the destination register (r).
- STr,(b,d)-The store instruction creates an address in the same manner as the load instruction. The register (r) is stored in the SDMA location at the resulting address.

46.13.15 Functional Unit Instructions

The functional unit instructions have an 8-bit field that is placed on the functional unit bus.

Some of these bits are used to select which functional unit should be involved in the transfer. The remaining bits are decoded by the selected functional unit so their specific use depends on the functional unit. See Functional Units Programming Model.

There are two functional unit instructions, as follows:

- LDFr,fub-The 8-bit field is placed on the functional unit bus and a read is issued to the selected functional unit. As a result of this instruction, the SF may be set in the processor status.
- STFr,fub-The 8-bit field is placed on the functional unit bus and a write is issued to the selected functional unit. As a result of this instruction, the DF may be set in the processor status.

46.13.16 Illegal Instructions

All instruction encodings that are illegal cause the following actions:

- The current PC (which points to one beyond the offending instruction) is put in the EPC register.
- The loop mode bit is cleared.
- The PC is set to the value stored in the Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR) register (the default value is 0x0001).

ILLEGAL-Although any instruction other than those indicated in the SDMA specification will trigger the illegal instruction mechanism, the ILLEGAL instruction code is preferred as it will always be kept as *illegal* in the possible future versions of the SDMA core.

46.13.17 Debug Instructions

The following are debug instructions:

- SOFTBKPT-The software breakpoint instruction causes the core to stop and enter debug mode. The core can then be accessed and started by the OnCE debug block only.
- done 5-This instruction is used for debugging, as it copies the contents of the PCU registers and flags to the context memory. Information on this instruction is described in Saving the Context.
- CpShReg-This instruction copies the context memory into the PCU registers and flags. Modifying the corresponding memory location before executing this instruction enables you to have the channel continue from a new instruction address. This instruction is described in Restoring the Context.

46.14 Functional Units Programming Model

The functional unit instructions cause an 8-bit code, found in the low eight bits of the instruction, to be asserted on the functional unit control bus.

Some of these bits are used to select one of several functional units. Functional units which can be selected include SDMA registers such as MSA and MSD which are not mapped in the SDMA memory map, and are accessible only through the functional unit bus. These Functional Unit Registers are listed in the following table. In order to establish

a programming convention, assume the selection bits are some number of the most significant bits of the 8-bit code. Furthermore, some number of the least significant bits is decoded by a given functional unit to establish the type of operation to perform.

Table 46-14. Functional Unit Registers

Functional Unit	Register	Register Name	Section/Page
Burst DMA Unit Programming	SDMSA	Memory Source Address Register	Memory Source Address Register (MSA)
	MDA	Memory Source Address Register	Memory Destination Address Register (MDA)
	MD	Memory Data Buffer Register	Memory Data Buffer Register (MD) (Write) Burst DMA Write (stf) (Read) Burst DMA Read (ldf)
	MS	Memory State Register	State Register (MS)
Peripheral DMA Unit Programming	PSA	Peripheral Source Address Register	Peripheral Source Address Register (PSA)
	PDA	Peripheral Source Address Register	Peripheral Destination Address Register (PDA)
	PD	Peripheral Data Buffer Register	Peripheral Data Register (PD) (Write) Peripheral DMA Write (stf)-Write Mode (Read) Peripheral DMA Read (ldf)-Read Mode
	PS	Peripheral State Register	Peripheral State Register (PS)
CRC Unit	CA	Polynomial Register	Polynomial Register (CA)
	CS	Accumulator Register	Accumulator Register (CS)

More information regarding the functional units can be found in CRC Calculation Unit,Peripheral DMA Unit, and Burst DMA Unit.

46.14.1 Burst DMA Unit Programming

The DMA instructions control the DMA state machine and may cause a DMA cycle on the associated memory bus.

There are four registers associated with the burst DMA unit: a Memory Source Address register (MSA), a Memory Destination Address register (MDA), a Memory Data buffer (MD), and a state register (MS). The burst DMA has two different uses:

- A data transfer between External Memory Interface and SDMA general register
- A data transfer in copy mode where blocks of data are transferred from the source address to the destination address

46.14.1.1 Memory Source Address Register (MSA)

The source address register contains the pointer into EXTMC memory associated with the next read data transfer. It has byte granularity.

Reading the register with the ldf instruction has no side effects, and gives the address value in the EXTMC memory of the next data that is read by the SDMA during an ldf MD instruction.

Writing the source address register has two side effects: If the prefetch bit is set, a DMA read cycle (8-word read access) is issued with the new address. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MSA to guarantee all the data is effectively written to memory.

The MSA register has two modes of programming:

- Frozen-In frozen mode, the MSA register is not modified after DMA accesses.
- Incremented (default mode)-In incremental mode, MSA is incremented by the number of bytes transferred during read cycles.

46.14.1.2 Memory Destination Address Register (MDA)

The destination address register contains the pointer into EXTMC memory associated with the next write data transfer. It has byte granularity.

Reading the MDA register with the ldf instruction has no side effects. It gives the address value in the EXTMC memory where the next SDMA data (stf r,MD instruction) is stored when MD FIFO is flushed.

Writing the destination address register has one side effect. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MDA to guarantee all the data is effectively written to memory.

The MDA register has two modes of programming:

- Frozen-In frozen mode, the MDA register is not modified after DMA accesses.
- Incremented (default mode)-The MDA register is incremented by the number of bytes transferred during write cycles.

46.14.1.3 Memory Data Buffer Register (MD)

The data buffer register consists of a bank of 36 bytes that behave like FIFO.

This FIFO stores the eight words received when a read burst is triggered by the DMA (DMA is in read mode).

The MD register is in write mode after a writing in MDA or after an stf MD instruction.

In that case, a burst write access is automatically triggered when there are more than eight words in MD. For bandwidth optimization, any transfers between DMA and the EXTMC controller are based on burst accesses.

An ldf r,MD|SIZE instruction that reads the data buffer may cause a DMA cycle, as follows:

- If there are less bytes in the FIFO than the size parameter of the instruction. For instance, if only two bytes are available in MD and a 4-byte read is requested, a burst read access is executed to complete the two bytes.
- If the prefetch bit is set, and after reading there is enough space in the FIFO to store a full burst, a burst read access is triggered.

An stf r,MD|SIZE instruction that writes to the data buffer may cause a DMA cycle if the number of written bytes in MD is higher than 32 (eight words) or if the flush bit is set.

When DMA is used for data transfer between SDMA and EXTMC (reading or writing), no immediate error is possible because the block manages a data misalignment issue; therefore, it is allowed to read/write a word to/from a half-word address. However, the addresses (source or destination) must belong to the EXTMC memory mapping. The only potential error, in this mode, would be the error sent back by the EXTMC controller when an access to a super-user page is detected. The whole transfer on the DMA associated bus will be considered successful when there are no errors seen on the bus during the transfer. In copy mode, an immediate error could be returned to SDMA as described in Burst DMA Unit Error Management.

46.14.1.4 State Register (MS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer. MS is also accessed to set-up the conditional yielding feature.

The initialization value of this register is 0 and it consists of the following:

Table 46-15. SDMA_MS Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	spriv	stype	0	0	dpriv	dtype
W																
R	0	0	0	0	y	d	e		0	0	n					
W																

Table 46-16. SDMA_MS Field Descriptions

Field	Description
31-22	Reserved
21 spriv	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
20 stype	Source Mode. Indicates if MSA has to be incremented (or not) during accesses. 0 Frozen-MSA is not modified. 1 Incremented-MSA is incremented by the number of transferred bytes during read access.
19-18	Reserved
17 dpriv	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved
16 dtype	Destination Mode. Indicates if MDA has to be incremented (or not) during accesses. 0 Frozen-MDA is not modified. 1 Incremented-MDA is incremented by the number of transferred bytes during write access.
15-12	Reserved
11 y	Conditional Yielding selector. When selected, theyield/yieldge instructions will not switch channels if the Burst DMA is in Write Mode, and it has less than four bytes in its FIFO. This is aimed at reducing the number of inefficient FIFO flushes due to context switches. 0 Always yields 1 Yields conditionally (when there are less than four bytes in the FIFO in write mode)

Table continues on the next page...

Table 46-16. SDMA_MS Field Descriptions (continued)

Field	Description
10 d	Access Direction or DMA Mode. DMA is in write mode when data was written into MD by stf MD instructions, or if a previous DMA cycle on the external bus was a write access. Writing MDA or MSA changes the DMA mode to the respective value. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by SDMA with an ldf MD instruction. Reading MDA or MSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9-8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 error read burst
7-6	Reserved
5-0 n	Number of bytes in the MD FIFO.

46.14.1.5 Burst DMA Write (stf)

When received from a stf instruction, the function code bits are interpreted as follows, depending on the addressed register:

Table 46-17. STF Code Bits

Register	7	6	5	4	3	2	1	0
MSA	s		p	freeze	r			spriv
MDA								dpriv
MD			f	cpy			sz	
MS								

Table 46-18. STF Code Bit Field Descriptions

Field	Description
7-6 s	Functional Unit selector 00 for Burst DMA
5 p (MSA)	Prefetch Flag 0 No prefetch 1 Prefetch required from new MSA

Table continues on the next page...

Table 46-18. STF Code Bit Field Descriptions (continued)

Field	Description
5 f (MD)	Forced Flush Flag 0 Automatic flush 1 FIFO contents are flushed (including the new written data).
4 freeze (MSA/MDA)	Address Freeze Mode 0 Address is normally incremented. 1 Address is frozen.
4 cpy (MD)	Copy Mode selection 0 Write Mode 1 Copy Mode
3-2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1-0 sz (MD/MS)	Transfer Size 00 size 0 (no data stored in the FIFO) 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
0 spriv (MSA)	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
0 dpriv (MDA)	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved

The possible write instructions are listed in the table below (unused bits should always be cleared).

Table 46-19. Burst DMA STF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	stf r,MSA	Writes content of the SDMA general register (r) to the source address register. MSA is in incremented mode.
00_0_1_00_00	stf r,MSA|FR	Writes content of the SDMA general register (r) to the source address register. MSA is in frozen mode.
00_1_0_00_00	stf r,MSA|PF	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access. MSA is in incremented mode.

Table continues on the next page...

Table 46-19. Burst DMA STF Instruction List (continued)

Binary	Assembly	Comments
00_1_1_00_00	stf r,MSA|PF|FR	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access.
00_0_0_01_00	stf r,MDA	Writes content of the SDMA general register (r) to the destination address register. MDA is in incremented mode.
00_0_1_01_00	stf r,MDA|FR	Writes content of the SDMA general register (r) to the destination address register. MDA is in frozen mode.
00_1_0_10_00	stf r,MD|SZ0|FL	No data transfers between the SDMA and MD, but all valid written data of the MD is flushed to the memory. An acknowledge or error is sent back to the SDMA core on transfer completion.
00_0_0_10_01	stf r,MD|SZ8	8-bit (byte) transfer to write buffer MD
00_1_0_10_01	stf r,MD|SZ8|FL	8-bit (byte) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_10	stf r,MD|SZ16	16-bit (half-word) transfer to write buffer MD
00_1_0_10_10	stf r,MD|SZ16|FL	16-bit (half-word) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_11	stf r,MD|SZ32	32-bit (word) transfer to write buffer MD
00_1_0_10_11	stf r,MD|SZ32|FL	32-bit (word) transfer to write buffer MD and flush after transfer. All valid written data of MD is flushed to memory.
00_0_1_10_00	stf r,MD|CPY	No data transfer between SDMA and MD but starts a copy transfer whose length is given by the 4 LSB of *r* register. (Maximum burst length is eight words.)
00_0_0_11_11	stf r,MS	32-bit (word) transfer to status register MS
00_0_0_11_00	stf r,MS|SZ0	Clears the error flag (if set). Other MS bits are unchanged; this instruction is also known as clref MS.

NOTE

When a flush bit is set, the SDMA flushes the FIFO including the newly written data. An acknowledge is sent to the core before the flush completes (except if size 0 is used). The goal of this flush bit is to force a flush, but it is recommended to use it only when needed (for example, when finishing a row of pixels during 2D data transfers). Indeed, if this bit is omitted and if there are more than 32 bytes in the FIFO, a burst write access is automatically triggered.

Since all the stf r,MD instructions (including the copy mode) acknowledge the SDMA core before the store is effective (except if size 0 is used), it is recommended to perform an ldf from MS before terminating a channel in order to check the final error status. (The ldf from MS will stall the core until all the data was flushed out and the transfer status is known.)

After every stf MD instruction, the MDA is incremented by the number of bytes that are written in MD, except when it is programmed in frozen mode.

46.14.1.6 Burst DMA Read (ldf)

When received from an ldf instruction, the function code bits are interpreted as follows, depending on the addressed register:

Table 46-20. LDF Code Bits

Register	7	6	5	4	3	2	1	0
MSA	s				r			
MDA								
MD			p				sz	
MS								

Table 46-21. LDF Code Bit Field Descriptions

Field	Description
7-6 s	Functional Unit selector 00 for Burst DMA
5 p (MD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
3-2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1-0 sz (MD)	Transfer Size 00 reserved 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

The table below lists the possible write instructions (unused bits should always be cleared).

Table 46-22. Burst DMA LDF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	ldf r,MSA	Copies the source address register value into an SDMA general register. It gives the memory address of the next data that will be read with an ldf MD instruction.
00_0_0_01_00	ldf r,MDA	Copies the destination address register value into an SDMA general register. It gives the memory address where the next incoming data will be flushed.
00_0_0_10_01	ldf r,MD|SZ8	8-bit (byte) read
00_1_0_10_01	ldf r,MD|SZ8|PF	8-bit (byte) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_10	ldf r,MD|SZ16	16-bit (half-word) read
00_1_0_10_10	ldf r,MD|SZ16|PF	16-bit (half-word) read. If after this reading, and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_11	ldf r,MD|SZ32	32-bit (word) read
00_1_0_10_11	ldf r,MD|SZ32|PF	32-bit (word) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_11_00	ldf r,MS	Copy the status register value into an SDMA general register.

NOTE

Read data is 0-extended before writing in the SDMA general registers. When reading the MD register, the DMA takes data from the FIFO if it is available. If part or whole data is not in the FIFO, an external burst read access is performed to provide the missing data. The SDMA is stalled as long as the required read data is not complete.

After every reading, MSA is incremented by the number of read bytes from MD FIFO, except when MSA is programmed in frozen mode.

46.14.1.7 Prefetch/Flush and Auto-Flush Management-Burst DMA Unit

The prefetch and auto-flush management enables the SDMA RISC machine to go on while a DMA access is performed.

When the RISC core requires a prefetch (p = 1) to the Burst DMA, it will receive an immediate transfer acknowledge before the DMA has finished the external access. This enables the RISC core to do other things like accessing another DMA machine.

The basic principle in prefetch mode is for the DMA to anticipate data reads from the SDMA RISC engine by fetching external bursts of data as soon as there is enough space in the DMA FIFO to store it. If ever the RISC engine required data that is not available in the FIFO, the read acknowledge is delayed until the data is available, but it does not have to wait until the burst completes.

The auto-flush basic principle is similar: An automatic flush is triggered every time there are eight words to be written in the FIFO. If the FIFO is full and the RISC engine requires another write, it is stalled until the burst has started and enough space was freed in the FIFO to store that new data. This means the SDMA RISC engine does not have to wait for the completion of a burst to receive its acknowledge and continue its processing.

In particular, an auto-flush is executed when DMA is in write mode and if the following is true:

- If the FIFO is empty and the first write is to a word-aligned address of any size (ex: the 2 LSB of MDA[1:0]= 0x0), the auto-flush is triggered immediately after the write of the 32'nd byte.
- If the FIFO is empty, and if MDA is an odd byte address (1, 3, 5, 7,...) and an stf MD|SZ8 is executed, the byte is flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is a half-word address (2, 6, 0xA,...) and an stf MD|SZ16 is executed, the two bytes of the incoming data are flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is not a word-aligned address (ex 1, 2, 3, 5, 6, 7, 9,...), and an stf MD|SZ32 is executed, the first 1 to 3 bytes will be flushed up to the next word aligned address. Afterwards, an auto-flush will be triggered each time the FIFO receives 32-bytes.
- Therefore, if an stf MD|SZ32 is executed with MDA equal to 0x1 and with an empty MD FIFO, the bytes located at addresses 1, 2, and 3 are flushed, and the byte located at address 4 remains in MD FIFO. This solves the misalignment issue. Additionally, the next write instructions (stf) complete the FIFO until it contains eight words; then a burst write is executed by the DMA to empty the FIFO. Protocol on the external bus does not support bursts of different data types (byte, half-word, or word).

  For example, consider the case where data is written using a byte access, stf MD| SZ8. The value of MDA during the very first byte write determines when the auto-flush will occur as follows:

- If MDA=0x0, the flush occurs following the write of byte 32
- If MDA=0x1, the flush occurs following the write of byte 1, byte 3 and byte 35.
- If MDA=0x2, the flush occurs following the write of byte 2 and byte 34.

- If MDA=0x3, the flush occurs following the write of byte 1 and byte 33.
- If MDA=0x4, the flush occurs following the write of byte 32

The flush command forces the DMA to flush all MD valid bytes to the EXTMC controller. An acknowledge is sent immediately to the SDMA, and any potential error is reported on a future access. It is thus essential to conclude a transfer with a last read from MS, which will stall the core until all data was flushed out and returned to the transfer status (acknowledge or error).

NOTE

During this kind of auto-flush (which occurs only at the beginning of a misaligned write transfer) no acknowledge is sent back to the SDMA, which is stalled until a flush is completed.

46.14.1.8 Data Alignment and Endianness-Burst DMA Unit

46.14.1.8.1 Burst DMA in Read Mode

For every read access to MD, the data returned to the SDMA core and the new FIFO state depends on the MSA status and the access size.

The FIFO is considered as a stack of 36 bytes: Data is fetched externally on a 32-bit bus, but the valid bytes only are stored in the FIFO and left-aligned (for a transfer of consecutive words, it is only the first word that may be truncated). The following table shows the FIFO byte alignment strategy and the corresponding MSA, the returned data, and the new FIFO state for any access size of an internal read from MD.

Table 46-23. FIFO Read Configuration

Before read		Internal read access size	Read data	After read	
MSA[1:0]	FIFO state			MSA[1:0]	FIFO state
00	x0 x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 and so on...	sz8	00 00 00 x0	01	x1 x2 x3 y0 y1 y2 y3 z0
		sz16	00 00 x0 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz32	x0 x1 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3

Table continues on the next page...

Table 46-23. FIFO Read Configuration (continued)

Before read		Internal read access size	Read data	After read	
MSA[1:0]	FIFO state			MSA[1:0]	FIFO state
01	x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 and so on...	sz8	00 00 00 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz16	00 00 x1 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz32	x1 x2 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
10	x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 and so on...	sz8	00 00 00 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz16	00 00 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz32	x2 x3 y0 y1	10	y2 y3 z0 z1 z2 z3 t0 t1
11	x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 t2 and so on...	sz8	00 00 00 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz16	00 00 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
		sz32	x3 y0 y1 y2	11	y3 z0 z1 z2 z3 t0 t1 t2

46.14.1.8.2 Burst DMA in Write Mode

For every write access to the MD, the new FIFO state depends on the MDA status and the access size.

The FIFO is considered as a stack of 36 bytes: Data is stored in the FIFO according to the internal access size and the former MDA value. The following table shows the FIFO byte alignment strategy corresponding to MDA, as well as the new FIFO state for any access size of an internal write to MD.

Table 46-24. FIFO Write Configuration

Before write		Internal write access size	Written data	After write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
00	tt uu vv ww ?? ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	01	tt uu vv ww x0 ?? ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	10	tt uu vv ww x0 x1 ?? ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	00	tt uu vv ww x0 x1 x2 x3 ?? ?? ?? ??
01	tt uu vv ww xx ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	10	tt uu vv ww xx x0 ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	11	tt uu vv ww xx x0 x1 ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	01	tt uu vv ww xx x0 x1 x2 x3 ?? ?? ??
10	tt uu vv ww xx yy ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	11	tt uu vv ww xx yy x0 ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	00	tt uu vv ww xx yy x0 x1 ?? ?? ?? ??
		sz32	x0 x1 x2 x3	10	tt uu vv ww xx yy x0 x1 x2 x3 ?? ??

Table continues on the next page...

Table 46-24. FIFO Write Configuration (continued)

Before write		Internal write access size	Written data	After write	
MDA[1:0]	FIFO state			MDA[1:0]	FIFO state
11	tt uu vv ww xx yy zz ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	00	tt uu vv ww xx yy zz x0 ?? ?? ?? ??
		sz16	?? ?? x0 x1	01	tt uu vv ww xx yy zz x0 x1 ?? ?? ??
		sz32	x0 x1 x2 x3	11	tt uu vv ww xx yy zz x0 x1 x2 x3 ??

NOTE

If the FIFO mode changes from a write to a read mode, all remaining written bytes in MD are lost but no error is returned. Typically, this happens if an ldf MD is executed after stf MD instructions. Before a mode change, it is recommended to force the flush of a potential remaining byte by a stfMD|SZ0|FL instruction. In the same way, if a FIFO mode changes from a read to a write mode, all prefetched data present in the FIFO is lost and no error is returned.

46.14.1.8.3 Endianness-Burst DMA Unit

Big and Little Endian are supported by the Burst DMA, but data is always stored in MD in Big Endian.

Byte manipulation is performed when data is exchanged with an Burst controller (for example, during read or write burst accesses).

46.14.1.9 Burst DMA Unit Copy Mode

A mechanism is available to perform fast ARM-to-ARM transfers.

Data does not flow through the SDMA core: It is kept in the DMA FIFO. This mechanism is selected when writing MD with a special option in the instruction code (copy flag).

It is possible to transfer up to eight words in one SDMA instruction (this does not mean in one cycle). In this mode, every time an stf MD|CPY is executed, a read burst is executed and directly followed by a write burst transfer. Burst transfers are limited to eight words. The size of the transfer (in words)-given by the SDMA general register (4 LSB)-is also limited to eight. The following SDMA code shows how 100 bytes could be copied from the MSA address to the MDA address. This is sample code only.

Burst DMA copy mode example

```
          ldi r0,@src
          stf r0,MSA                          // Source address setup
          ldi r1,@dst
          stf r1,MSA                          // Destination address setup
          ldi r0,0x64                         // data transfer counter
          ldi r1,0x8
MAIN_XFER:
          cmphs r0,r1                         // Is r0 >= 0x8
          bf LAST_XFER                      // If not, jump to last transfer label
          stf r1,MD|CPY                        // Copy 8 words from MSA to MDA address.
          subi r0,0x8                         // Decrement counter
          jmp MAIN_XFER                     // return to main transfer loop
LAST_XFER:
          stf r0,MD|CPY
```

The main transfer loop is executed 12 times; then r0 equals 4 and the last transfer loop is run.

In this mode, an acknowledge is transmitted to the core as soon as the read burst can start; thus, a first copy instruction returns an immediate acknowledge and subsequent copy instructions will be acknowledged as soon as the previous copy has finished.

46.14.1.10 Burst DMA Unit Error Management

Another point to consider is the management of errors.

Because the DMA immediately sends an acknowledge to the RISC core (except for the stf MS|SZ0|FLS instruction), it assumes no error will occur. If an error occurs, it is flagged (transfer error acknowledge) for the following DMA access.

This should not be a problem if the DMA is used properly. The MD accesses are meant to stall the SDMA as little as possible to optimize throughput and hide calculation time. Therefore, final access to MS should be performed before closing a channel. This access waits until any pending operation is finished in the burst DMA and gather any remaining error.

In copy mode, an error could be immediately returned to the SDMA on execution of the ldf copy or stf copy instruction. It happens when MSA or MDA are not word addresses (for example, 0[4]). This is because copy mode must only be used for transferring a large packet of aligned data.

When an error is received during a *read* transfer to the external bus, which may occur during the burst accesses, the MD FIFO contains the valid beats of the burst, and the error flag of MS is set to 2'b11 (error read burst). It is possible to read MS ("n" field) to know how much valid data remains in MD and when MD is empty (after ldf instructions). The next read MD instruction sets the MS error flag to 2'b10 (error mode), and an error is sent back to the SDMA core. In error mode, it is possible to read MSA, which gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, gives rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

In "error read burst" mode, writing MDA, MSA, or MD, or starting a copy transfer by a stf MD|COPY instruction will cancel the error mode. The following table shows when an immediate error is sent back according to the executed instruction.

Table 46-25. Possibilities in ERROR READ BURST Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA (|U |PF) stf rn, MDA stf rn,MD|COPY	NO	Error mode is reset. MSA, MDA, or MD are updated and a DMA cycle may start. For the stf MD|COPY, a copy loop is executed.
stf rn, MS	NO	MS is updated.
ldf rn, MS ldf rn, MSA ldf rn, MDA	NO	MS, MSA, and MDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, MD	YES/NO	Immediate error if there is no more data available for read in the FIFO.

When an error is received during a *write* transfer, the error is reported to the next DMA access. In this case, an error is sent to the SDMA core and the DMA goes to its error mode. Reading MS gives the number of bytes that remain in MD; reading MDA gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, give rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

Table 46-26. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA stf rn, MDA	Yes	Any attempt to modify MD, MSA, MDA will raise an immediate error and burst DMA remains in error mode. When address registers are write-accessed, an error is returned.
stf rn, MS	No	This is the only way to exit error mode. MS[9:8] must be reset by an stf MS|SZ0 instruction.
ldf rn, MS ldf rn, MSA ldf rn, MDA	No	MS, MSA, and MDA could be read in error mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, MD	Yes	Whatever the DMA direction (read or write), an ldf rn triggers an immediate error.

46.14.1.11 Conditional Yielding-Burst DMA Unit

The standard SDMA transfer is based upon a hardware loop that has the following structure:

Hardware Loop

```
loop
load Rn,source            // can be ldf or ld
<computation>            // can be done through functional units
store Rn,dest              // can be st or stf
done 0               // yield
```

This structure needs to be kept independent of the functional units' particularities regarding the context switch. However, there can be variations in the context switch's efficiency, which can depend on the number of data received up to that point, and on the data itself.

The DMA, with its 8-word burst capability, has a preferable context switch period when its address register is 8-word aligned: It is the only moment that occurs once every eight loops when the succession of bursts is not broken by the context switch. When this is not the case, a context switch requires the storing (or loading) of less than eight words, which requires separate accesses and is far less efficient. The rest of the 8-word packet is stored (or loaded) after the context restore, and this is done as separate accesses.

The proposed solution is a conditional yielding, which occurs only when the DMA is in an optimum state. It does not require any modification to the scripts. The condition is decided at the DMA level.

The DMA can be programmed in two modes-conditional or always-true-for every channel, which provides complete flexibility. By default, the DMA is not in conditional mode.

The DMA condition is computed from the FIFO fill level and the various modes, as follows:

- When copy mode is selected, regardless of the transfer direction ('read' or 'write'), the condition is always true.
- In read mode, the condition is always true.
- In write mode, the condition is true when there are four bytes or less in the FIFO; it is false when there are more than four bytes. The 4-byte limit comes from the possibility of saving those bytes as MD with absolutely no impact on the bus accesses.

The aim at conditional yielding is to avoid splitting bus accesses (especially bursts).

46.14.2 Peripheral DMA Unit Programming

The peripheral DMA unit is connected to the Multi-Layer DMA Crossbar Switch of the ARM platform. Its goal is to perform data transfers between any blocks connected to the DMA bus of this platform.

These blocks are either peripherals or memories. The peripheral DMA could be seen as the ARM platform DMA controller.

The DMA performs data transfers in three modes:

- Read mode, where data is read from peripherals or from memory connected to the ARM platform and copied in a SDMA general register.
- Write mode, where data of a general register has to be written in a peripheral or a memory.
- Copy mode, where data is read from a peripheral (or memory) at a source address (PSA) and automatically written to a peripheral (or memory) at a destination address (PDA).

In copy mode, no SDMA general register is involved as transferred data only goes through the data register of the DMA.

The peripheral DMA has three addressing modes: frozen, incremented, and decremented, as follows:

- Frozen mode-When source or destination addresses are frozen, their value is not modified after a transfer. This mode is typically used for addressing peripheral FIFOs located at a fixed address.

- Incremented mode-When source or destination addresses are in incremented mode, after every transfer they are incremented by the number of bytes transferred.
- Decremented mode-In decremented mode, addresses are decremented by the number of bytes transferred.

The peripheral DMA registers are as follows:

- Two, 32-bit address registers (PSA and PDA) that respectively contain the source address for a read access and the destination address for a write access
- A 32-bit status register (PS) that contains information on the peripheral DMA configuration, such as the number of valid bytes in the data register, the error flag, the source and destination address mode, and so on.
- A 32-bit data register (PD) that stores data involved in a data transfer

46.14.2.1 Peripheral Source Address Register (PSA)

The source address register contains a pointer to a source peripheral or a memory associated with the next read data transfer. It has byte granularity.

It is based on the following:

- A 32-bit register (PSA) to store the address value
- A 2-bit register (stype) to store the source address mode (frozen, incremented, or decremented)
- A 2-bit register (ssize) to store the source target data path size (byte, half-word, or word)

Reading the register with the ldf instruction has no side effects and gives the address value of the next data that will be read by the SDMA during an ldf MD instruction. Writing the source address register may have side effects. If there is valid write data in the data register and the source address is changed, the write data is discarded. If the prefetch bit is set, a DMA read cycle is issued with the new address.

When PSA is to be written, you must specify the source target address mode, providing its size (byte, half-word, or word). This enables omission of the size field in all ldf MD instructions. When DMA performs a read cycle, its size is given by the value of the PSA source size register (ssize). If source is a memory in incremented mode, first programmed in word mode (stf PSA|SZ32|I), and if an SDMA script needs to read bytes from this memory, the size of the source target must be updated before executing new accesses. The source address mode and its size are given by labels added to the stf PSA instruction as described in the write section. The ssize and stype registers are part of the DMA status register (PS).

Writing to PSA may issue an immediate error if the source size is not compatible with the value to be written into the PSA register. For instance, writing a 2 in PSA and specifying that it is memory-accessed in word mode creates an immediate error.

46.14.2.2 Peripheral Destination Address Register (PDA)

The destination address register contains a pointer to a source peripheral or a memory associated with the next write data transfer. It has byte granularity.

It is based on the following:

- A 32-bit register (PDA) to store the address value
- A 2-bit register (dtype) to store the destination address mode (frozen, incremented, or decremented)
- A 2-bit register (dsize) to store the destination target data path size (byte, half-word, or word)

Reading the register with the ldf instruction has no side effects, and gives the address value of the next data that will be written by SDMA during an stfMD instruction. Writing the destination register has no side effect. Similar to the PSA register, the destination address mode and source are specified in the stf PDA instruction and may also generate an error in case of incorrect programming.

46.14.2.3 Peripheral Data Register (PD)

The data register of the peripheral DMA is a 32-bit register. When the destination address is correctly set up, any writing to PD will automatically flush the new input data.

The number of SDMA bytes that will be transferred is given by the PDA size register. Unlike other SDMA DMAs, PD is not a FIFO: It is not used to accumulate bytes that from the SDMA and must be packed before being sent to external memories. In read mode, and if the source address is correctly set up, an ldf instruction will empty PD. If a prefetch is required along with the instruction, the DMA will initiate a new read transfer.

Reading PD in prefetch mode only stalls the SDMA when the prefetched data is not yet available. Writing PD only stalls the SDMA if the previous write operation was not completed. As soon as the previous operation is over, the acknowledge is sent back to the SDMA RISC engine.

An error flag-part of PS-is set when an external access fails. The error is thus reported to the next SDMA instruction that involves the peripheral DMA.

46.14.2.4 Peripheral State Register (PS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer.

Although all PS fields can be written by an stf instruction, it is recommended to access only the error bit (to reset it). Modifying other PS fields will provide an un-guaranteed DMA behavior.

The initialization value of PS is 0, and it consists of the following structure:

Table 46-27. PS Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ssize		stype		dsize		dtype	
W																
R	0	0	0	0	0	d	e		0	0	0	0	0	n		
W																

Table 46-28. PS Field Descriptions

Field	Description
31-24	Reserved
23-22 ssize	Source Target Size. Determines the size of the read transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
21-20 stype	Source address Mode. Determines whether PSA is incremented, decremented, or kept unmodified after every read from the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
19-18 dsize	Destination Target Size. Determines the size of the write transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

Table continues on the next page...

Table 46-28. PS Field Descriptions (continued)

Field	Description
17-16 dtype	Destination address Mode. Determines whether PDA is incremented, decremented, or kept unmodified after every write on the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
15-11	Reserved
10 d	Direction Flag or DMA Mode. DMA is in write mode when data was written into PD by stf PD instructions, or if a previous DMA cycle on the external bus was a write access. Writing PDA or PSA does not change the DMA mode. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by the SDMA with an ldf PD instruction. Reading PDA or PSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9-8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 Error read
7-3	Reserved
2-0 n	number of bytes in PD

NOTE

dtype, dsize, stype, and ssize are updated when PSA and PDA are written.

46.14.2.5 Peripheral DMA Write (stf)-Write Mode

When written by an stf instruction, the function code bits are interpreted as follows:

Table 46-29. STF Code Bits

Register	7	6	5	4	3	2	1	0
PSA	s		p	ar	am		sz	
PDA								
PD			pdsel					
PS			pssel					

Table 46-30. STF Code Bits Field Descriptions

Field	Description
7-6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PSA)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
3-2 am (PSA/PDA)	Address Mode. Determines how PSA or PDA is modified after every read or write access to the PD. 00 Frozen-Address registers are not modified after the transfer. 01 Incremented-Address registers are incremented by the number of transferred bytes. 10 Decremented-Address registers are decremented by the number of transferred bytes. 11 Updated-PSA and PDA are not modified. Either address mode is not modified, but the width of the data path is updated by the sz field.
1-0 sz	Transfer Size 00 *reserved* 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
5-0 pdsel	PD access selector 001000 is the only valid option
5-0 pssel	PS access selector 111111 writes to PS 001100 only clears the error flag in PS

Due to the large number of possible stf instructions, the following table provides only a short list of all the possible write instructions:

Table 46-31. Peripheral DMA STF Instruction List

Binary	Assembly	Comments
11_00_00_01 11_00_00_10 11_00_00_11	stf Rn, PSA|SZ8 |F stf Rn, PSA|SZ16|F stf Rn, PSA|SZ32|F	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is frozen.
11_10_00_01 11_10_00_10 11_10_00_11	stf Rn,PSA|SZ8 |F|PF stf Rn,PSA |SZ16|F|PF stf Rn,PSA |SZ32|F|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • 1, 2, or 4 bytes are *fetched* from the peripheral source. • Source address is frozen.

Table continues on the next page...

Table 46-31. Peripheral DMA STF Instruction List (continued)

Binary	Assembly	Comments
11_00_01_01 11_00_01_10 11_00_01_11	stf Rn, PSA|SZ8 |I stf Rn, PSA|SZ16|I stf Rn, PSA|SZ32|I	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1,2 or 4 after read PD.
11_10_01_01 11_10_01_10 11_10_01_11	stf Rn, PSA|SZ8 |I|PF stf Rn, PSA|SZ16|I|PF stf Rn, PSA|SZ32|I|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1, 2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_10_01 11_00_10_10 11_00_10_11	stf Rn, PSA|SZ8 |D stf Rn, PSA|SZ16|D stf Rn, PSA|SZ32|D	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA-1,2, or 4 after read PD.
11_10_10_01 11_10_10_10 11_10_10_11	stf Rn, PSA|SZ8 |D|PF stf Rn, PSA|SZ16|D|PF stf Rn, PSA|SZ32|D|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA-1,2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_11_01 11_00_11_10 11_00_11_11	stf Rn, PSA|SZ8 |U stf Rn, PSA|SZ16 |U stf Rn, PSA|SZ32 |U	• *Update* source pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost.
11_10_11_01 11_10_11_10 11_10_11_11	stf Rn, PSA|SZ8 |PF|U stf Rn, PSA|SZ16 |PF|U stf Rn, PSA|SZ32 |PF|U	• *Update* source pointer, which becomes a pointer to a target accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost. • 1, 2, or 4 bytes are *fetched* from the memory source.
11_01_00_01 11_01_00_10 11_01_00_11	stf Rn, PDA|SZ8 |F stf Rn, PDA|SZ16|F stf Rn, PDA|SZ32|F	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is frozen.
11_01_01_01 11_01_01_10 11_01_01_11	stf Rn, PDA|SZ8 |I stf Rn, PDA|SZ16|I stf Rn, PDA|SZ32|I	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA + 1, 2, or 4 after write PD.
11_01_10_01 11_01_10_10 11_01_10_11	stf Rn, PDA|SZ8 |D stf Rn, PDA|SZ16|D stf Rn, PDA|SZ32|D	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA-1, 2, or 4 after write PD.
11_01_11_01 11_01_11_10 11_01_11_11	stf Rn, PDA|SZ8 |U stf Rn, PDA|SZ16 |U stf Rn, PDA|SZ32 |U	• *Update* destination pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PDA value is not modified by Rn • bytes present in PD are lost

Table continues on the next page...

Table 46-31. Peripheral DMA STF Instruction List (continued)

Binary	Assembly	Comments
11_00_10_00	stf Rn, PD	• Write "dsize" bytes of Rn in PD and automatically flush to destination target
11_11_11_11	stf Rn, PS	• Write status register
11_00_11_00	stf Rn,clrefPS	• Clear error flag if set

NOTE

When writing PD, size information is not important: It is embedded in the dsize field of PDA register. If dsize is 1, 2, or 4, then one, two, or four bytes from Rn is written to the PD register, and automatically flushed out to the destination target.

46.14.2.6 Peripheral DMA Read (ldf)-Read Mode

When received from an ldf instruction, the function code bits are interpreted as follows.

Table 46-32. LDF Code Bits

Register	7	6	5	4	3	2	1	0
PSA	s			ar	a			
PDA								
PD			p	cpy				
PS			pssel					

Table 46-33. LDF Code Bits Descriptions

Field	Description
7-6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
4 cpy (PD)	Copy Mode 0 standard access 1 copy mode access

Table continues on the next page...

Table 46-33. LDF Code Bits Descriptions (continued)

Field	Description
3 a	Register Set selection 0 PSA or PDA 1 PD or PS
5-0 pssel	PS access selector 111111 is the only valid option to read PS

Table 46-34. Peripheral DMA LDF Instruction List

Binary	Assembly	Comments
11_0_0_0_000	ldf Rn, PSA	Reads 32-bit of PSA value
11_0_1_0_000	ldf Rn, PDA	Reads 32-bit of PDA value
11_0_0_1_000	ldf Rn, PD	Reads programmed source size bytes of PD (0-extended)
11_1_0_1_000	ldf Rn, PD|PF	Reads programmed source size bytes of PD (0-extended), and starts a prefetch at PSA address.
11_0_1_1_000	ldf Rn, PD|COPY	Starts a copy transfer from the source target at the PSA address to the destination target at the PDA address. No data transmits through Rn, but Rn contents are lost (Rn is loaded with PD temporary contents that are *not* the copied data).
11_111111	ldf Rn, PS	Reads 32-bit of PS value

NOTE

When reading PD, size information is not important: It is embedded in the ssize field of the PSA register. If ssize is 1, 2, or 4, the one, two, or four bytes is transferred from PD to Rn. Read data is 0-extended.

46.14.2.7 Peripheral DMA Unit Copy Mode

Like burst DMA, the peripheral DMA unit has a copy mode that is used when data transfers do not involve SDMA general registers.

Data is read from the source target at a PSA address, stored in PD, and then automatically flushed to the destination target at the PDA address. Copy mode is only available for transfers that involve two targets of the same data path width.

Since copy mode is invoked with an ldf instruction, the *loaded* general purpose register loses its previous contents. (However, the new contents are unpredictable as they depend on temporary values that are seen on the external DMA bus.)

46.14.2.8 Error Management

Peripheral DMA generates two kinds of errors: the immediate error that sanctioned incorrect register programming; and the error triggered by the previous access and stored in the error flag of PS until a DMA instruction is executed.

46.14.2.8.1 Immediate Errors

The following table lists all incorrect DMA register setups.

Table 46-35. Immediate Errors with Peripheral DMA

Rn[1:0] values	DMA instruction	Comments
0x01 0x11	stf Rn, PSA|SZ16|F stf Rn, PSA|SZ16|I stf Rn, PDA|SZ16|F stf Rn, PDA|SZ16|I	If PSA points to a half-word peripheral or to a half-word address in memory, its value must be 0 modulo 2.
0x01 0x10 0x11	stf Rn, PSA|SZ32|F stf Rn, PSA|SZ32|I stf Rn, PDA|SZ32|F stf Rn, PDA|SZ32|I	If PSA points to a word peripheral or to a word address in memory, its value must be 0 modulo 4.
PSA[1:0]-PDA[1:0]	DMA instruction	Comments
0x01 0x10 0x11	stf Rn, PSA|SZ32|U stf Rn, PDA|SZ32|U	When PDA or PSA is updated and becomes a pointer to a word address in memory, its content must be 0 modulo 4.
0x01 0x11	stf Rn, PSA|SZ16|U stf Rn, PDA|SZ16|U	When PDA or PSA is updated and becomes a pointer to a half-word address in memory, its content must be 0 modulo 2.
Read/Write PD instruction	Comments	
stf Rn,PD ldf Rn,PD	If PDA size (dsize) has never been set up before an stf PD instruction (dsize=0) If PSA size (ssize) has never been set up before an ldf PD instruction (ssize=0)	
ldf Rn,PD|CPY	Copy mode is possible only between two targets whose data path width is identical. It is P8↔P8, P16↔P16, or P32↔P32 regardless of the way the address registers are incremented.	

46.14.2.8.2 Data Transfer Errors

When PSA and PDA are correctly set up, the only error that may arise for an ldf PD or stf PD instruction would be the error of the previous DMA cycle.

Error handling is driven by a single consideration: When an error occurred during a data read on the DMA interface, this error should appear as a transfer error to the core when the core attempts to retrieve the data that was not successfully read from the accessed device (memory or peripheral).

When an error occurred during a write access to the DMA interface, the data is still available in PD and should not be destroyed by subsequent core accesses: The core must be warned about the error issue.

There are three error handling mechanisms for each case: Read Error (First Phase), Write Error and Read Error (Second Phase), and Copy Mode Errors handling.

46.14.2.8.3 Read Error (First Phase)

If an error occurred during a prefetch command, the peripheral DMA enters its ERROR READ mode (PS[9:8]=11). In this mode, the error is reported on the next ldf PD instruction and writing PSA, PDA, or PD will cancel the error flag.

The block returns no error mode and instructions are normally executed (a DMA cycle may be triggered). Similarly, initiating a copy transfer will reset the error flag and start a copy transfer. The following table details which instructions can be executed in this mode.

Table 46-36. Possibilities in ERROR READ Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA (IU IPF) stf rn, PDA ldf rn,PD|COPY	NO	Error mode is reset, PSA or PDA are updated, or a write cycle is started. For the ldf PD|COPY, a copy loop is executed.
stf rn, PS	NO	PS is updated.
ldf rn, PS ldf rn, PSA ldf rn, PDA	NO	PS, PSA, and PDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, PD	YES	Error of the previous read access is reported here and the peripheral DMA enters its ERROR mode.

46.14.2.8.4 Write Error and Read Error (Second Phase)

The peripheral DMA enters its ERROR mode (PS[9:8]=10) when the previous DMA write cycle failed, or, as explained in Read Error (First Phase), when an ldf PD is executed while the block is in ERROR READ mode. When a DMA cycle failed, address registers (PSA, PDA) are not modified and continue to point to the problematic address. In ERROR mode, stf instructions may raise an immediate error, and ldf instructions will not (as detailed in the table below).

Table 46-37. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA stf rn, PDA	YES	Any attempt to modify PD, PSA, or PDA will raise an immediate error, and the peripheral DMA stays in ERROR mode. When address registers are write accessed, an error is returned.
stf rn, PS	NO	This is the only way to exit the ERROR mode. PS[3] must be reset by an stf PS instruction.
ldf rn, PS ldf rn, PSA ldf rn, PDA	NO	PS, PSA, and PDA could be read in ERROR mode without any side effects (for example, no DMA cycle is triggered).
ldf rn, PD	YES	Whatever the DMA direction (read or write), an ldf rn, PD instruction will show an immediate error.

46.14.2.8.5 Copy Mode Errors

Because copy mode is a write access that follows a read access, there are two possible cases of bus error.

When the read access incurs a bus error, the peripheral DMA behaves exactly as described in Read Error (First Phase) and Write Error and Read Error (Second Phase) : It enters its ERROR READ mode, and so on.

When the error occurred during the write access of the copy transfer, the DMA enables the core to retrieve the data that was read because it is assumed the read from the peripheral removed the data from its source device. Therefore, the data to be flushed is still in PD. Any subsequent access to PD triggers an error to the core, which should execute its error handling procedure.

Once the ERROR mode is left (after writing to PS), it is possible for the core to retrieve the data in PD with an ldf instruction or try to flush PD contents once again (for example, when the error was due to a full FIFO and the script waited for the FIFO to be emptied) with another ldf instruction in copy mode. This latter instruction detects that there is valid data in PD, tries to flush it, and thus skips the read phase of the copy instruction. This is a different behavior from the usual stf PD instruction that overwrites PD with the selected General Purpose register contents. The same mechanism can be used any time PD holds data that is not written because of a bus error on the DMA interface; when the data was written via a copy instruction, or via the usual stf PD instruction.

46.14.2.8.6 Error Check Example

The following code illustrates an example checking for both immediate and data transfer errors on a store to the PD register. The first bdf instruction checks for an immediate error, but if a data transfer error occurred it is reported until the next instruction to access the Peripheral DMA. A second check of the error flags is done after the ldf PS

instruction. The value of PS here can be ignored. The act of reading any register in Peripheral DMA while it is in an error mode that returns the error to the core to set either the SF or DF flag. Any error returned on an ldf command sets the SF flag and any error returned on an stf instruction sets the DF flag. This can create a situation as shown in the example where a bus error during a DMA write which would normally be considered as a destination fault is reported as a source fault because the error was reported to the SDMA core during an ldf instruction.

Peripheral DMA Error Check

```
    clrf    0            // Clear SF and DF flags
 stf    R4, PD        // Write data to memory
bdf    error_routine    // Check for immediate error from write to PD.
ldf    r3, PS        // Read PS (PS value in R3 can be ignored)
    bsf    error_routine    // Check for bus error from "stf R4,PD"
            // SF is set because it is a ldf instruction, even though
            // the original error was a destination fault
```

46.14.2.9 Peripheral DMA Unit Prefetch/Flush Management

There is no flush bit because every time data is stored in PD by a stf PD instruction-assuming PDA is correctly programmed-it is automatically flushed to the destination.

An acknowledge is returned in the cycle of the DMA instruction, and the SDMA is only stalled by an instruction that addresses the peripheral DMA when the previous DMA access is not over.

46.14.3 CRC Unit

The Cyclic Redundancy Check (CRC) unit is connected to the SDMA core via the FUBUS. This unit can perform a CRC calculation for a set of given polynomials from degree 8 to 32.

When all CRC unit registers are loaded, the unit can process one byte of data every clock cycle. When loading new data to compute the CRC, the SDMA can perform 32-bit, 16-bit, or 8-bit accesses.

Two 32-bit registers comprise the unit:

- The CRC algorithm CA that describes the polynomial
- The CRC checksum CS to accumulate the data after each processing

46.14.3.1 Polynomial Register (CA)

This register defines the CRC algorithm currently used in the calculation, and the management of data ordering to/from the data register CS.

Before starting any CRC calculation, it must be loaded with the chosen polynomial reference number and data ordering mode as indicated in the following figure and table.

Table 46-38. CA Structure

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

Table 46-39. CA Descriptions

Field	Description
31-5	Reserved
4 ri	Reverse bit order of data in 0 Must be used when the peripheral receives bytes as bit streams in LSB-first order (UART case). 1 Selects the reverse mode, which must be used when the peripheral receives bytes as bit streams in MSB-first order (MMC case).
3 ro	Reverse bit order of data out 0 Must be used when the peripheral transmits bytes as bit streams in LSB-first order (UART case). 1 Selects the reverse mode, which must be used when the peripheral transmits bytes as bit streams in MSB-first order (MMC case).
2-0 p	Polynomial selection 000 CRC32 ($X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$) 001 CRC16 ($X^{16}+X^{15}+X^2+1$) 010 CCITT16 ($X^{16}+X^{12}+X^5+1$) 011 IS136 ($X^{12}+X^{10}+X^8+X^5+X^4+X^3+1$) 100 CRC10 ($X^{10}+X^9+X^5+X^4+X+1$) 101 CRC8 ($X^8+X^2+X+1$) 110 Parity ($X^8+1$) 111 Reserved

46.14.3.2 Accumulator Register (CS)

The accumulator register accumulates the division remainder during the CRC processing.

When writing the accumulator register (process bit is not set) if the *ro* bit is set, the write data has its order reversed whatever the data length (the length is specified by the selected polynomial).

When computing new CRC (writing CS and process bit set) if the *ri* bit is set in CA, any byte of write data will have its bit order reversed before storage and calculation. In the first byte, bit 7 is replaced by bit 0, bit 6 is replaced by bit 1, and so on. In the second byte, bit 7 is replaced by bit 0, bit 6 is replaced by bit 1, and so on, which means in the write data, bit 15 is replaced by bit 8, bit 14 will replaced by bit 9, and so on.

If the *ro* bit is set in CA, any read data will have its bit order reversed whatever the data length. If the valid data length is 16 bits, bit 15 is replaced by bit 0, bit 14 is replaced by bit 1, and so on.

When loading new data to compute the CRC, the SDMA can perform 32-bit, 16-bit, or 8-bit accesses. The CRC is computed in four clock cycles when performing a 32-bit access, in two clock cycles when performing a 16-bit access, and in one clock cycle when performing an 8-bit access. In all cases, an immediate acknowledge is sent back to the SDMA. A wait state is inserted if the SDMA tries to perform a new access before the end of the computation.

When performing a multi-bit access, the first byte used to compute the CRC is the most significant byte of the valid data.

46.14.3.3 Write Instruction (stf)

The following table shows the stf instructions.

Table 46-40. Stf Instructions for CRC

Byte Command								Description
1	0	0	0	0	0	0	0	Write polynomial encoded in the General Register (CA)
1	0	0	0	0	1	0	0	Write accumulator register (right-aligned)
1	0	0	1	0	1	0	0	Compute the CRC with the new incoming byte (b0)
1	0	0	1	0	1	0	1	Compute the CRC with the new incoming byte (b0)
1	0	0	1	0	1	1	0	Compute the CRC with the new incoming halfword (b1 b2) The bytes are used in the following order: b1 and b0.
1	0	0	1	0	1	1	1	Compute the CRC with the new incoming word (b3 b2 b1 b0) The bytes are used in the following order: b3, b2, b1, and b0.

46.14.3.4 Read Instruction (ldf)

The following table shows the ldf instructions.

Table 46-41. Ldf Instructions for CRC

Byte Command								Description
1	0	0	0	0	0	0	0	read the polynomial register encoded
1	0	0	0	0	1	0	0	read the accumulator register, right aligned

46.14.3.5 Operating Mode

The following is the operating mode example:

- Load the polynomial register to select the algorithm and preset the accumulator, if required.
- Data is right-aligned in the general register.
- Input data is fed byte-by-byte into the accumulator through stf instructions.
- When all data is fed into the CRC unit, the CRC checksum is read with ldf ca, sz.

16-bit CCITT CRC with $P(X) = X^{16}+X^{12}+X^5+1$

```
        .equ rL,0 // GReg[0] = rL; loop count register
.equ rA,1 // GReg[1] = rA; CCITT16 polynomial
.equ rP,2 // GReg[2] = rP; initial CRC value
.equ rT,3 // GReg[3] = rT; transferred byte register
.equ aT,4 // GReg[4] = aT; transferred byte address
// (MMC DAT_RX - 0xA003)
.equ CA,8 // CRC unit CA register
.equ CS,9 // CRC unit CS register
ldi rA,2 // init register with CCITT16 polynomial
stf rA,CA // updates the selected polynomial
ldi rP,0xff // initializes register with 0x000000ff
stf rP,CS,0 // presets the accumulator with 0x000000ff
ldi aT,0xA0 // initializes aT with 0xA003: aT = 0x000000A0
revblo aT // initializes aT with 0xA003: aT = 0x0000A000
ori aT,0x03 // initializes aT with 0xA003: aT = 0x0000A003
ldi rL,200 // expects to read 200 bytes from MMC DAT_RX
loop 2 // executes 200 times the next 2 instructions
```

```
ld rT,(aT,0) // loads next byte from MMC DAT_RX

stf rT,CS,1 // process checksum with new incoming byte

ldf rT,CS // read the final checksum at the end
```

46.14.4 OnCE and Real-Time Debug

The On-Chip Emulation block (OnCE) is the debug interface to the SDMA. It supports the access to all core internal devices (registers, memory, and so on), and provides a set of mechanisms that control the core.

The OnCE is accessed by JTAG ports at the chip's board level, or by the host via its peripheral bus.

To reduce the size of the hardware material involved, all tasks supported by the OnCE are performed on the SDMA core. The architecture of the SDMA OnCE is relatively simple and very flexible.

The commands supported by the SDMA OnCE are listed in the following sections.

46.14.4.1 Memory and Register Access

A set of mechanisms is provided to access SDMA memory and register locations. Both reading and writing are allowed. The access is supported if the processor is in debug mode.

Those registers can also be accessed through the ARM platform Control interface when the OnCE is controlled by the ARM platform, as described in the "Using BP" section.

46.14.4.2 Hardware Breakpoints

An event detection unit is implemented to support memory breakpoints. The unit watches the data exchanged between the SDMA memory bus and the core.

A debug request is sent to the core when matching conditions occur. The unit supports mixed conditions based on address range, access type, and data value. Event detection unit configuration registers are memory mapped in the SDMA space (see ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)): You can modify them through a regular memory access or the ARM platform control interface.

46.14.4.3 Watchpoints

One output pin is provided to monitor matching trigger conditions that are defined in the event detection unit.

46.14.4.4 Software Breakpoints

The SDMA instruction set contains a software breakpoint. Upon executing a software breakpoint instruction, the core suspends normal execution and enters debug mode.

No hardware step execution mode is implemented in the OnCE, but this feature may be implemented at the software level with this instruction.

46.14.4.5 Core Control

Commands are provided to monitor and control processor activity. You can halt the core, rerun the core from another address location, and get processor status.

Any hardware breakpoint on the instruction bus is not supported, but this feature may be implemented by inserting a software breakpoints program.

46.15 The OnCE Controller

The OnCE controller receives commands from the ARM platform or from the JTAG controller. Each command is interpreted before being sent to the core.

46.15.1 OnCE Commands

A small set of commands supports the communication between the OnCE and the external world. This command set enables you to perform any of the following tasks: control processor activity, save core context, and execute an SDMA instruction from the OnCE.

Combined together, these tasks perform more complex commands.

A full OnCE command contains a 4-bit instruction (the OnCE command opcode) and a variable length data field (the OnCE data). During command execution, the OnCE data is transferred in a OnCE internal register before being exchanged with the SDMA. Some

data values are also exported. This mechanism creates a link between the processor and the external world. Nine commands are defined: The following table presents their formats.

Table 46-42. OnCE Command Opcode Values

Instruction Opcode	Name	Action	Register	Data Field Size	Mode
0000	rstatus	Reads the OnCE status register	STATUS	16-bit	normal/debug
0001	dmov	Updates general register GReg1	GREG1	32-bit	debug
0010	exec_once	Runs the instruction from the SDMA instruction register	INSTRUCTION	16-bit	debug
0011	run_core	Returns to normal execution	BYPASS	1-bit	debug
0100	exec_core	Returns to normal execution via a jump instruction that specifies the new address	INSTRUCTION	16-bit	debug
0101	debug_rqst	Stops the core after execution of current instruction	BYPASS	1-bit	normal
0110	rbuffer	Reads the real time buffer	RTB	32-bit	normal/debug
0111-1110	reserved	Reserved	BYPASS	1-bit	normal/debug
1111	bypass	Bypasses TARM platform controller	BYPASS	1-bit	normal/debug

Each instruction corresponds to a specific action performed on the OnCE. The nature of the associated data field is clearly identified. The dmov command is followed by a 32-bit data value (which is a data value for the SDMA); the exec_once and the exec_core commands are followed by a 16-bit data value (which is an instruction for the SDMA); the rstatus command is followed by a 16-bit control value (which is the content of the OnCE status register); the rbuffer command is followed by a 32-bit data value. The debug_rqst and the run_core commands are followed by a single bit data field (this is a bypass value). Finally, the bypass instruction enables the SDMA JTAG TAP controller to be daisy-chained with another JTAG TAP controller. This is a JTAG-only feature. The set of commands is simple, but enables you to perform any possible task on the SDMA during a debug process.

46.15.2 Sending Commands to the OnCE Controller

The JTAG access is the standard access to the OnCE, but sometimes the JTAG is not available to fix some bugs (if the chip is in production for instance), an additional access is then required. Therefore, one ARM platform access to the OnCE is provided.

46.15.2.1 Using the JTAG Interface

A serial access is performed through the five JTAG pins TCK, TRST, TMS, TDI, and TDO. A Test Access Port controller is provided to decode the TMS control signal.

It produces shift-enable signals (shift_ir and shift_dr), and updates enable signals (update_ir and update_dr). It is fully compliant with the IEEE 1149.1 testability (JTAG) standard.

During the shift_ir state, the command opcode is shifted into the OnCE controller (for example, the signal from the TDI pin is shifted into the command register and the TDO pin receives the signal shifted out). After transferring the four bits of the command, an update_ir signal is asserted and the command is decoded. The target data register is now clearly identified and the corresponding control signal is produced, as follows: bypass enable signal (bp_en), instruction enable signal (inst_en), data enable (data_en), and status enable signal (stat_en).

During the shift_dr state, the TDI signal is shifted into one of the following target registers: bypass register (1 bit), SDMA instruction register (16 bits), SDMA data register (32 bits), or OnCE status register (16 bits). The TDO pin is connected to the output of the selected register to receive the signals shifted out.

The JTAG access is disabled when the ARM platform access is enabled.

46.15.2.2 Using the ARM platform

The ARM platform access to the OnCE is not the standard access, but it is required if the JTAG is not available.

For example, if the SDMA ROM is out of use on a chip in production, and the ARM platform needs to download new code and restart the SDMA, the OnCE can easily perform this operation. This type of debug operation justifies the use of an ARM platform access to the OnCE.

To drive the OnCE, the ARM platform uses some registers contained in the ARM platform Control block of the SDMA. These registers are accessed through the ARM platform peripheral bus. Most of these registers are connected to another register in the OnCE controller. Thus, accessing one of these registers is equivalent to accessing the associated register in the OnCE controller.

The set of registers in the ARM platform Control block is listed below:

- ONCE_ENB register (1 bit, read/write)-This 1-bit register enables the ARM platform access to the OnCE. When this bit is set, the signals from the JTAG are ignored.

When it is cleared, all writing operations to the following registers through the Host Control interface are ignored. This register is reset on a JTAG reset.

- ONCE_CMD register (4 bits, read/write)-This 4-bit register receives the command opcode. It is connected to the command register in the controller. A write access to this register causes the associated command to be executed on the OnCE. For example, after writing "0001" in this register, a dmov command is executed.

NOTE

On the ARM platform side, the rstatus and bypass commands are not supported. This register is reset on a JTAG reset.

- ONCE_DATA register (32 bits, read/write)-This 32-bit register is connected to the SDMA data register. This register is used when executing a dmov or rbuffer command.

NOTE

Before requesting a dmov command, the 32-bit data to transfer must be written in the ONCE_DATA register. At the end of the execution, the register is updated with GReg1 former value. This register is reset on a JTAG reset.

- ONCE_INSTR register (16 bits, read/write)-This 16-bit register is connected to the SDMA instruction register. This register is used when executing an exec_core or an exec_once command.

NOTE

Before requesting an exec_core or an exec_once command, the appropriate instruction must be written in the ONCE_INSTR register. This register is reset on a JTAG reset.

- ONCE_STAT register (16 bits, read only)-A read access to the ONCE_STAT register returns the content of the OnCE status register (OSTAT). This register is read only.
- The bypass register is not useful when the ARM platform controls the OnCE, therefore no register is defined in the ARM platform Control block to access the bypass register.

46.15.2.3 Conflicts Between the JTAG and the ARM platform Accesses

When ARM platform access to the SDMA OnCE is enabled (that is, when the bit in the ONCE_ENB register is set), the JTAG access is disabled. This guarantees that the block is not accessed at the same time on both sides.

It is possible to check whether the JTAG access to the SDMA OnCE is enabled from the JTAG port. When the JTAG access is disabled, the SDMA TDO always returns 1. The check requires the following steps:

- Execute a dmov command from debug mode (with neither 0xffffffff nor 0x0 as dmov value: 0x5a5a5a5a is good).
- Execute another dmov command (the value here is not important).
- The returned value from the latter dmov command should be the original one if the JTAG access is enabled; if it is 0xffffffff instead of the original input value, this means the JTAG access is disabled.

46.15.3 Executing a Command from the OnCE

All the commands defined in OnCE Commands can be accessed through the JTAG. The ARM platform can access all these commands except the rstatus command.

On the ARM platform side, the OnCE status is directly accessed by reading the ONCE_STAT register.

46.15.3.1 Nature of the Commands

Two types of commands may be distinguished. First, there are two commands that do not interact with the core: rstatus and rbuffer. Those commands may be requested at any time: They do not depend on the core status.

NOTE

Each of these commands exports a data value or a status value from the SDMA.

There are also commands that interact with the core: dmov, run_core, exec_core, exec_once, and debug_rqst. These commands are core status dependent, as follows:

- During user mode only the debug_rqst is taken into account.
- During debug mode, all these commands are taken into account except the debug_rqst. For example, an exec_once command requested while not in debug mode has no effect.

46.15.3.2 Execution Request

The SDMA starts executing a task in debug mode when requested by the OnCE controller. The execution starting time depends on the type of access used to communicate with the OnCE.

If the JTAG is used, the request is send after decoding the update_dr state in the TAP controller. Therefore, always cross this state when sending a command through the JTAG. If the OnCE is driven from the ARM platform side, the request is sent after detecting a write access to the ONCE_CMD register. All the registers involved in this operation must be loaded first.

The following is an example of an exec_core command execution from the ARM platform side: After writing '010' in the ONCE_CMD register, the OnCE controller asks the SDMA to execute the instruction contained in the ONCE_INSTR register. The instruction involved should be available in the ONCE_INSTR register before the beginning of the execution.

46.15.3.3 Command Execution

The following list shows the commands and details how each command is executed:

- rstatus command execution-The rstatus command exports the content of the OnCE status register (OSR). If the JTAG is used, the status information is captured in the OnCE status register during the capture_dr state, and shifted out after 16 TCK clock cycles in the shift_dr state. The rstatus command is not supported on the ARM platform side, but a status register is provided instead. The rstatus may be performed in both debug and user modes.
- dmov command execution-The dmov command accesses SDMA internal registers. Executing a dmov instruction exchanges the 32-bit data values between the SDMA data register and the general register GReg[1].
- If the JTAG is used, the content of GReg1 is captured in the SDMA data register during the capture_dr state, then it is shifted out after 32 TCK clock cycles in the shift_dr state. During the update_dr state, GReg1 is updated with the new, shifted-in 32-bit data value. If the OnCE is driven from the ARM platform side, the data values contained in GReg1 and the SDMA data register are exchanged after detecting a write access to the ONCE_CMD register. The ONCE_DATA register must therefore be loaded first.

- exec_once command execution-The exec_once command executes the instruction loaded in the SDMA instruction register. The command may only be requested from debug mode. The SDMA returns to debug mode at the end of the execution.

- Change of flow instructions as well as instructions that may cause a context switch are not supported: The comprehensive list comprises done/yield/yiedge (except done 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions.

  No other command should be requested before the SDMA returns to debug mode. The SDMA status (for example, whether it is in debug mode or not) can be detected by polling with the rstatus OnCE command, monitoring the debug_mode pin, or checking the OnCE Status Register (SDMAARM_ONCE_STAT) register via the ARM platform control interface.

  NOTE

  Most of the instructions are single-cycle, which omits the step of polling the status. Loads and stores to DMA units are typical instructions that might require this polling.

  If the JTAG is used, the 16-bit instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. A request is sent to the core when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the request is sent to the SDMA when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must be therefore be loaded first.

- run_core command execution-The run_core command leaves debug mode and resume normal program execution. The next instruction executed is the last instruction decoded before entering debug mode. Be sure to restore core context before re-running the core. This procedure is detailed in Restoring the Context.
- If the JTAG is used, a 1-bit bypass value is shifted in the bypass register in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the core is rerun when detecting a write access to the ONCE_CMD register.

- exec_core command execution-The exec_core command resumes program execution from any address. The 16-bit instruction provided with the exec_core overwrites the last instruction decoded before entering debug mode. This command is designed to support change of flow instructions, so that a program execution can be restarted from any address. After executing an exec_core command, the SDMA leaves debug mode. The exec_core command is usually used with a jmp instruction.
- If the JTAG is used, the 16-bit branch instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the SDMA reruns when detecting a write access to the

ONCE_CMD register. The ONCE_INSTR register must therefore be loaded first. For example, to restart the SDMA from the program address 0x100, the instruction loaded should be a jump to address 0x100 instruction.

- debug_rqst command execution-The debug_rqst command puts the SDMA in debug mode. If the JTAG is used, a 1-bit bypass value is shifted in the bypass register during the shift_dr state. A debug request is sent to the SDMA when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the ARM platform side, the debug request is sent when detecting a write access to the ONCE_CMD register. When the SDMA is already in debug mode, this command is simply ignored.
- rbuffer command execution-The rbuffer command exports the content of the real time buffer (RTB). If the JTAG is used, the content of the real time buffer (RTB) is captured in the SDMA data register during the capture_dr state. The register is completely shifted out after maintaining the shift_dr state during 32 TCK clock cycles. If the OnCE is driven from the ARM platform side, the content of the RTB is captured in the ONCE_DATA register after detecting a write access to the ONCE_CMD register.
- bypass command execution-This command is only available from the JTAG interface. It enables daisy-chaining of the SDMA JTAG TAP controller with other JTAG TAP controllers. This command does not change the SDMA state and can be executed in any mode (run, debug, or sleep). It selects the bypass register of the TAP controller.

46.15.4 Registers Descriptions

See SDMACORE, and SDMAARM, for detailed information on each register.

46.15.4.1 Event Cell Counter Register (ECOUNT)

The event cell counter register is a 16-bit register that contains the number of times minus one that an event detection occurs before generating a debug request.

This register should be written before attempting to use the event detection counter during an event detection process. The event cell counter register is cleared on a JTAG reset.

46.15.4.2 Event Cell Address Registers (EAA or EAB)

The event cell contains two address registers-the event cell address register (a), called EAA, and the event cell address register (b), called EAB. Every address register is a 16-bit register that stores a user-defined address value. This value computes one of the following address conditions: addra_cond or addrb_cond. Every address register is cleared on a JTAG reset.

46.15.4.3 Event Cell Address Mask Register (EAM)

The event cell address mask register is a 16-bit register that contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before comparing addresses.

NOTE

There is a common address mask value for the two address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The event cell address mask register is cleared on a JTAG reset.

46.15.4.4 Event Cell Data Register (ED)

The event cell data register is a 32-bit register that contains a user-defined data value. This data value is an input for the data comparator, which generates the data_cond condition.

The event cell data register is cleared on a JTAG reset.

46.15.4.5 Event Cell Data Mask Register (EDM)

The event cell data mask register is a 32-bit register that contains a user-defined data mask value. This mask is applied to the data value latched from the memory bus before comparing data.

Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. The event cell data mask register is cleared on a JTAG reset.

46.15.4.6 Real Time Buffer Register (RTB)

The real Time Buffer register is a 32-bit register that stores and retrieves run-time information without putting the SDMA in debug mode.

Refer to Real Time Buffer for more details.

46.15.4.7 Event Control Register (ECTL)

The event cell control register is a 16-bit register that defines cell event occurrence conditions.

The event cell control register is cleared on a JTAG reset. See also OnCE Event Detection Unit for more details.

46.15.4.8 Trace Buffer (TB)

The Trace Buffer register retrieves the information in the Trace Buffer.

See Trace Buffer for more details.

46.15.4.9 OnCE Status Register (OSTAT)

The OnCE status register is a 16-bit register that contains processor and event detection unit status. The OSTAT is a read-only register.

Refer to OnCE Status Register (SDMAARM_ONCE_STAT) for detailed description of the individual fields in the OSTAT register.

The following figure shows the OSTAT structure.

Table 46-43. OnCE Status Register (OnCE)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PST[3:0]				RCV	EDR	ODR	SWB	MST					ECDR[2:0]		

Where PST[3:0] is the SDMA core state, RCV is set when the real-time buffer (RTB) is modified. EDR, ODR, and SWB are set, respectively, when the SDMA has entered debug mode because of an external debug request, a OnCE debug_rqst command, or a software breakpoint. MST is set when the OnCE is controlled from the ARM platform control interface, and when ECDR is a three-flag set that shows the event cell condition(s) that put the core in debug mode. The OSTAT never provides more than one reason for entering debug mode.

There are two ways of accessing OSTAT content, as follows:

1. Send an rstatus command to the OnCE controller through the JTAG, or read the ONCE_STAT register through the ARM platform access. Executing the rstatus command through the JTAG can be performed in both user and debug modes.
2. Perform an SDMA read access to the location in the SDMA core memory map (OSTAT register) debug mode using the exec_once command. With this method of access, the SDMA state reflected by the PST (processor status bit) is always DATA.

The register may also be accessed by a running application.

46.15.5 JTAG Interface Requirements

Because the signals received from the JTAG (running on TCK) are transferred to the OnCE controller (running on the SDMA clock), a synchronization mechanism is required.

46.15.5.1 TCK Speed Limitation

In the JTAG top-level layer, the TDO signal is always captured on a TCK falling edge. To guarantee a stable TDO signal from the SDMA during this operation, a falling edge detection is performed on TCK.

Before being latched in the *I* flip-flop (see Figure 46-12) on TCK falling edge, the TDO signal must be stable at the input of the flip-flop. This condition is verified if the TCK period is superior to the following delay:

worst-case edge detection delay + negative-edge signal propagation delay + JTAG top-level logic propagation delay

The frequency relationship, TCK < CLK/8, limitation guarantees that all operations are performed as expected.

46.15.5.2 Synchronization Implementation

The following figure shows the synchronization mechanism. Flip-flops tck0, tck1, and tck2 perform falling- and rising-edge detections on TCK. They generate the posedge_detected and negedge_detected nets that are used to sample the TDI and TMS inputs into the respective tdi and tms flip-flops, and update the tdo flip-flop to yield the

TDO output. In the design, the only signal that might go metastable is the output of the tck0 flip-flop. This signal is captured in the tck1 flip-flop and no logical operation is performed on it to minimize a metastability propagation risk.

The TDI and TMS flip-flops also cannot go metastable: The propagation time of the rising-edge detection signal through tck0, tck1, and tck2 guarantees that the TDI and TMS inputs are stable when captured in the TDI and TMS flip-flops.

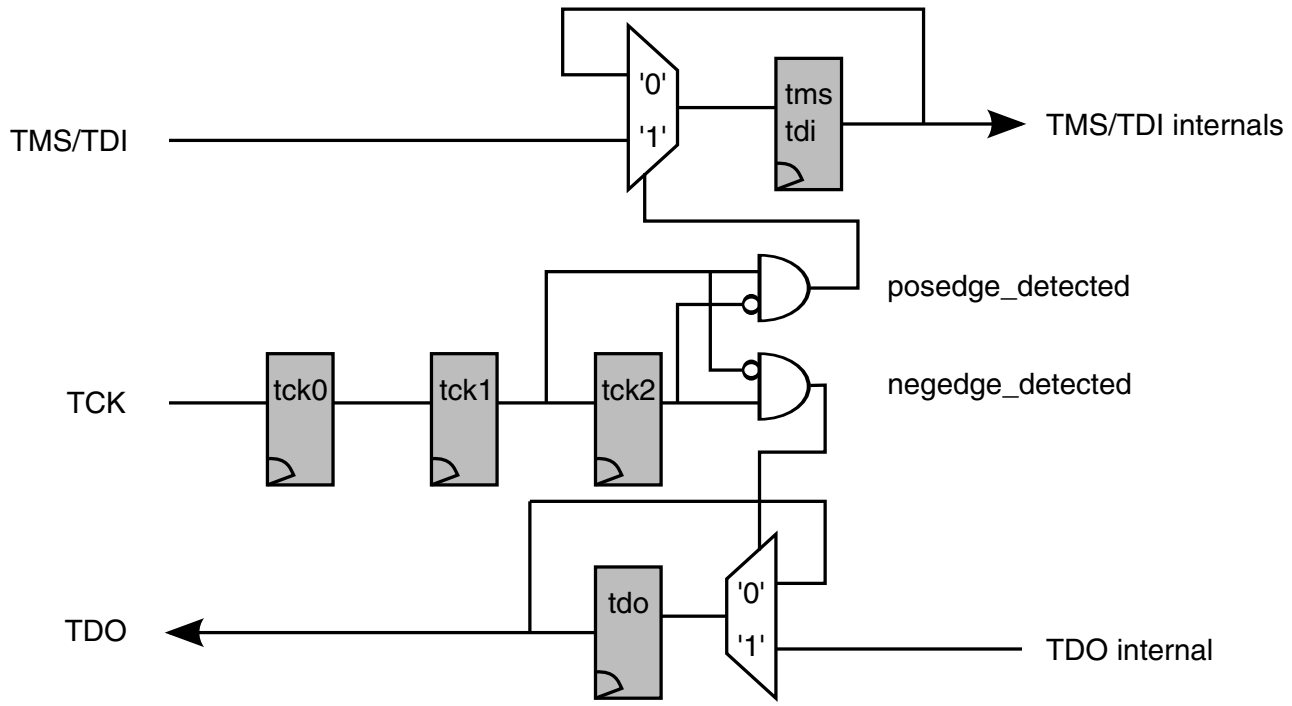


Figure 46-12. OnCE Synchronization Layer

The following figure shows synchronization timings. It takes three CLK clock cycles to synchronize TDI on the SDMA clock.

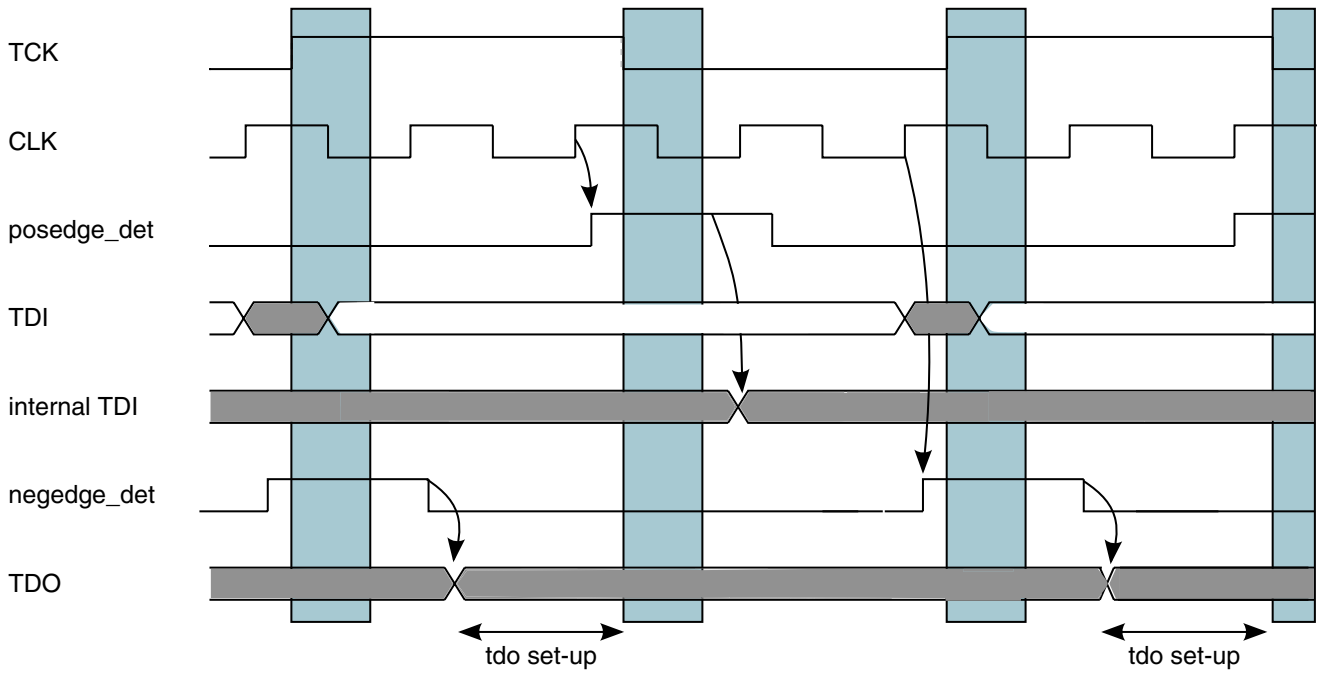


Figure 46-13. Synchronization Timings

46.15.5.3 JTAG Controller Start-Up Recommended Procedure

To ensure correct TAP controller initialization, it is recommended to use the following procedure:

1. Assert JTAG reset TRSTB (for example, set low).
2. Set TMS low.
3. Wait for 1 TCK clock.
4. Release JTAG reset TRSTB (for example, set high).
5. Wait for a minimum of five TCK cycles.

46.16 Using the OnCE

This section provides the elements necessary to run the OnCE during a debug process. In addition to the basic set of commands described in OnCE Commands, more complex commands can be built to meet users' requirements.

46.16.1 Activating Clocks in Debug Mode

For power consumption issues, some clocks in the SDMA are disabled when not needed. This is the case for instances when the SDMA is in sleep mode. Clock gating management depends on the interface used to control the OnCE.

- For the JTAG access, the SDMA clock gating must be turned off via the clk_gating_off input.
- For the ARM platform access, the SDMA clock gating is automatically turned off when the ARM platform access is enabled (see OnCE Enable (SDMAARM_ONCE_ENB)).

46.16.2 Getting the Current Status

Most of the commands the OnCE supports have an impact on the status of the SDMA.

It is not permissible to request the execution of an instruction on the SDMA from the OnCE while the SDMA is not in debug mode. Such a violation may cause unpredictable behavior, and it might be necessary to reset the SDMA.

Therefore, the value of the PST bits provided in the OnCE status register should always be checked before sending any request to the SDMA.

46.16.3 Methods of Entering Debug Mode

A debug request may be asserted at any time, but it is not always taken into account immediately. Debug mode cannot be entered in the middle of an instruction, or during the save or restore states of a context switch.

The request is ignored when the core is already in debug mode. Refer to Figure 46-4, which shows all possible transitions to the debug state, as there are several ways to enter debug mode.

46.16.3.1 External Debug Request During Reset

To enter debug mode after exiting reset, the external debug line has to be maintained high. This line is handled by the JTAG top-level block.

NOTE

The SDMA detects the debug requests only if the SDMA clock is running (see Activating Clocks in Debug Mode). The debug

request line should be not be maintained high when the SDMA is in debug mode.

NOTE

The debug_rqst command (from the OnCE command set) is not supported during system reset.

46.16.3.2 Debug Request During Normal Activity

During normal activity, the SDMA enters debug mode when the following is true:

1. If the debug request line from the JTAG top-level is asserted, or
2. If the OnCE controller receives a debug_rqst command.

   The debug_rqst command can be sent by the JTAG access or by an access on the ARM platform side (if the ARM platform access is enabled).

46.16.3.3 Software Breakpoint Instruction

The SDMA enters debug mode at the end of the execution of a software breakpoint instruction. This instruction must be inserted in program flow executed by the core.

46.16.3.4 Event Detection Unit Matching Condition

If the event detection is enabled, a debug request is sent to the core after detecting a matching condition on the SDMA memory bus.

See OnCE Event Detection Unit for more details.

46.16.4 Executing Instructions in Debug Mode

The OnCE supports a mechanism to execute instructions in debug mode. If the SDMA is in debug mode, then the exec_once command can be used to execute an SDMA instruction from the OnCE controller. The SDMA returns to debug mode at the end of each execution.

Some instructions are not supported by the exec_once command: done/yield/yiedge (except done 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions are not supported.

NOTE

While instructions are executed in debug mode from the OnCE, the program counter of the SDMA is not incremented.

46.16.5 Command Sequences Examples

This section provides examples of command sequences that run the SDMA in debug mode. These sequences are available for both the ARM platform and JTAG accesses.

The following presents the syntax used in this section. The data field provided with each command is put in parenthesis with the command name. A '-' is used if the data field provided is a *don't care* value.

```
my_command(data_field);                 // executing my_command with a data field
my_command(-);                        // executing my_command with a don't care data field
```

The value returned by the command (if there is one) is referred by an assignment. In case the value returned by the command is not used, the assignment is omitted. For an ARM platform access, the value returned (it is always a data value) is obtained by reading back into the SDMA data register.

```
data_out = my_command(data_in); // returning a data value
```

To clarify the syntax, the instructions' opcodes are referred to by their names. In practice, use the corresponding 16-bit encoding.

46.16.5.1 Getting the SDMA Status

NOTE

Before executing any command that affects the SDMA (like dmov or exec_once), check that the SDMA is in debug mode.

Use the following snippet:

```
rstatus();        // read SDMA status until the SDMA is in debug mode
...
rstatus();
```

If the SDMA is not in debug mode, then a debug request must be generated. In this case, the SDMA enters debug mode at the end of the execution of the current instruction. Use this snippet:

```
debug_rqst(-);   // debug request
```

In the following sections, it is assumed that the SDMA was successfully put into debug mode.

46.16.5.2 Saving the Context

The first debug task is to save the SDMA context, which is the content of the eight general-purpose registers, the loop and PC-related registers, and the flags. Use the general register GReg[1] as an intermediate register to export the entire context of the SDMA.

The following example shows how to save GReg[0], GReg[1], GReg[2] and GReg[3]. The sequence of commands used to export additional general registers is very similar to this.

Save GReg[0], GReg[1], GReg[2], and GReg[3]

```
GReg1_data = dmov(-);                          // the value exported is the content of
GReg[1]
exec_once("mov GReg1,GReg0");                      // puts the content of GReg[0] into
GReg[1]
GReg0_data = dmov(-);                          // the value exported is the content of
GReg[0]
exec_once("mov GReg1, GReg2");                     // puts the content of GReg[2] into
GReg[1]
GReg2_data = dmov(-);                          // the value exported is the content of
GReg[2]
exec_once("mov GReg1, GReg3");                     // puts the content of GReg[3] into
GReg[1]
GReg3_data = dmov(-);                          // the value exported is the content of
GReg[3]
```

Get the value of the internal flags (SF, DF, T, and LM), of the loop related registers (EPC and SPC), and of the PC-related registers (PC and RPC). Use a done 5, which is the formatting instruction dedicated to the debug. This instruction formats the flags and the values contained in the registers. It also writes the resulting values into the channel context memory. It should not be used when entering debug from the IDLE state (for example, with no active channel script running on the SDMA), because it will update a channel context that may belong to any channel.

```
exec_once("done 5");                    // formatting the value of flags and registers
```

At this point, the channel context should be up-to-date in memory, and debug operations should now be possible. However, the context can be exported with the following instructions:

Exporting the Context

```
dmov(ctx_base_addr);                              // loading GReg[1] with the channel
context base address
exec_once("ld GReg0,(GReg1,0)");                      // get RPC-PC into GReg0
exec_once("ld GReg1, (GReg1,1)");                     // get SPC-EPC into GReg1
Loop_data = dmov(-);                           // read back the value of Loop registers
exec_once("mov GReg1, GReg0");                     // puts the PC info into GReg1
PC_data = dmov(-);                             // reads back the content of the PC registers
```

After this sequence of operations, the entire SDMA context is exported via the OnCE.

46.16.5.3 Restoring the Context

At this point in the operation, restore the context of the SDMA. It can be different from the original context located in memory, and the content previously saved into the debugging application via the OnCE.

The following example shows how it is possible to modify the current channel context:

Modifying the Current Channel Context

```
dmov(Loop_data);                                  // put Loop former value into GReg[1]
exec_once("mov GReg0, GReg1");                         // copy to GReg[0]
dmov(PC_data);                                    // put PC former value into GReg[1]
exec_once("mov GReg2, GReg1");                         // copy to GReg[2]
dmov(ctx_base_addr);                                  // put channel context base address into
GReg[1]
exec_once("st GReg0, (GReg1,1)");                          // restore Loop context
exec_once("st GReg2, (GReg1,0)");                          // restore PC context
```

Once the context in memory is the desired context (with or without applying the previous instruction sequence), it can be restored to the *real* PC and loop registers in the SDMA core:

```
exec_once("cpShReg");                    // restore flags and PC & loop related registers
```

After this command, the SDMA core PC, RPC, SPC, EPC registers, as well as the flags contain the same data as what is stored in the context RAM for the current channel.

The following example shows how to restore the context of general registers GReg[0], GReg[1], GReg[2] and GReg[3].

Restoring the General Register Context

```
dmov(GReg3_data);                    // put GReg[3] restore value in GReg[1]
exec_once("mov GReg3, GReg1"); // restore GReg[3]
dmov(GReg2_data);                    // put GReg[2] restore value in GReg[1]
exec_once("mov GReg2, GReg1"); // restore GReg[2]
dmov(GReg0_data);                    // put GReg[0] restore value in GReg[1]
exec_once("mov GReg0, GReg1"); // restore GReg[0]
dmov(GReg1_data);                    // restore GReg[1]
```

At this point, it is possible to restart the normal program execution.

NOTE

Every SDMA core general register value can be modified by a mov instruction, which makes modification of these registers easy during debug. Unfortunately, there is no such instruction as a mov to directly modify the contents of either PCU register or flag (PC, RPC, SPC, EPC, T, LM, SF, or DF). The cpShReg instruction is meant to provide a means for changing these register contents via the context memory.

46.16.5.4 Accessing the Memory

In the following example, it is assumed that the SDMA context is entirely saved. If true, it is permissible to modify the general purpose registers during debugging activity.

To perform a memory read access, the target address is stored via the OnCE in GReg[1], then the load instruction is executed on the SDMA (the data loaded from the memory overwrites the address contained in GReg[1]), and then the result value is read back via the OnCE.

```
macro READ:            dmov(target_addr);                                  // put the target
address in GReg[1]
                    exec_once("ld GReg1,(GReg1,0)");                         // execute the
load instruction
                    res_data = dmov(-);                                 // exports the result
data value
```

For a memory write access, the target address is written in GReg[0], and the value to store is written in GReg[1]. Then the store instruction is executed on the SDMA.

```
macro WRITE:            dmov(target_addr);                                   // puts the
target address in GReg[1]
                    exec_once("mov GReg0,GReg1");                          // puts the target
address in GReg[0]
                    dmov(target_data);                                  // puts the target
data in GReg[1]
                    exec_once("st GReg1,(GReg0,0)");                          // performs the
store operation
```

This sequence is shown as an example; however, many other sequences are possible.

NOTE

This sequence of commands can also be applied to memory-mapped registers.

46.16.5.5 Resuming Program Execution

Before resuming program execution, it is assumed that the SDMA context is properly restored. There are two ways to restart the SDMA.

Start by executing the last instruction fetched before entering debug mode, as follows.

```
run_core(-);                          // resume execution from where we stopped before
```

If necessary, restart the execution from a different address. In this case, use the exec_core command. The data field provided with this command must be the encoding of a jump instruction.

```
exec_core("jmp start_addr"); // rerun the SDMA from another address
```

In these two examples, the SDMA exits debug mode and keeps executing the code fetched from the memory.

46.16.5.6 Single Stepping in RAM

To execute a program step-by-step from the RAM, insert software breakpoints in the program flow at appropriate places so that the SDMA only executes one instruction before returning to debug mode.

First, read the next instruction to execute in the RAM. Then, depending on the value of this instruction, compute the address where a software breakpoint instruction should be inserted. The instruction at the corresponding address must be saved, and, the software breakpoint instruction is inserted. After restarting the SDMA, there is only one instruction executed before meeting the software breakpoint.

The following example shows the macro functions READ and WRITE, which correspond to the sequence of commands (described above) used to access the memory.

NOTE

The data read from the memory are 32-bit values, while the instructions are 16-bit values only. This is why it is best to only use addresses divided by two when accessing the memory.

READ and WRITE Macro Functions

```
next_instr = READ(run_addr/2);                          // read the next instruction to execute
// the tool now has to compute the address where the breakpoint
// instruction should be inserted, this address is the "bkpt_addr"
instr_save = READ(bkpt_addr/2);                          // save the instruction before
overwriting
STORE("bkpt instruction",bkpt_addr/2);                         // store the bkpt instruction
in memory
exec_core("jmp run_addr");                              // rerun the SDMA
rstatus(-);                                           // wait for the SDMA to enter debug mode
...
rstatus(-);
STORE(instr_save,bpkt_addr/2);                           // restore the instruction
overwritten
```

In case of branched conditional instructions, a breakpoint instruction should be written at the two possible target addresses.

46.16.5.7 Single Stepping in ROM

No single-step mechanism is supported in ROM. The program code can be loaded in the RAM, where the single-step mechanism can be executed.

46.16.6 OnCE Event Detection Unit

The event detection unit watches signals from the data memory bus (DMBUS), which the SDMA core uses to access its RAM, ROM, and memory mapped registers.

A debug request is sent to the OnCE controller when user-defined conditions on address and/or data values are true.

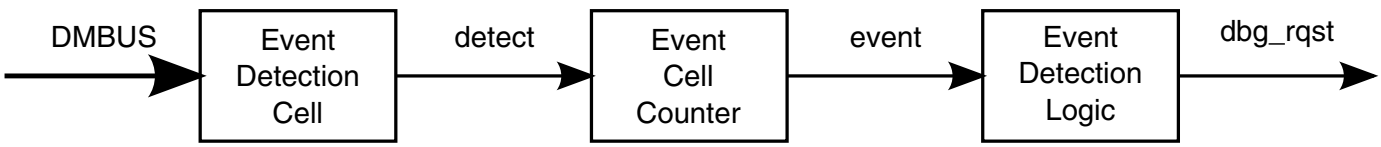


Figure 46-14. Event Detection Unit

A counter, provided with the detection cell, is decreased after an event detection. A debug request is sent to the core only when the counter reaches the value of 0. It is possible to disable the use of the counter if a debug request has to be generated after each event detection.

The event cell is the basic block that supports hardware breakpoints on an address value and/or data values coming from the SDMA memory bus. The trigger condition that generates the debug request is a mixed condition based on those values.

The following figure shows the event cell architecture. The event cell contains the address (stored in the memory address register) and the data (stored in the memory data register) used during the last memory access. There are some user-defined reference values located in memory mapped registers-the event cell addresses, the event cell address mask, the event cell data, and the event cell data mask. These registers are accessed by standard load/store instructions just like regular memory locations.

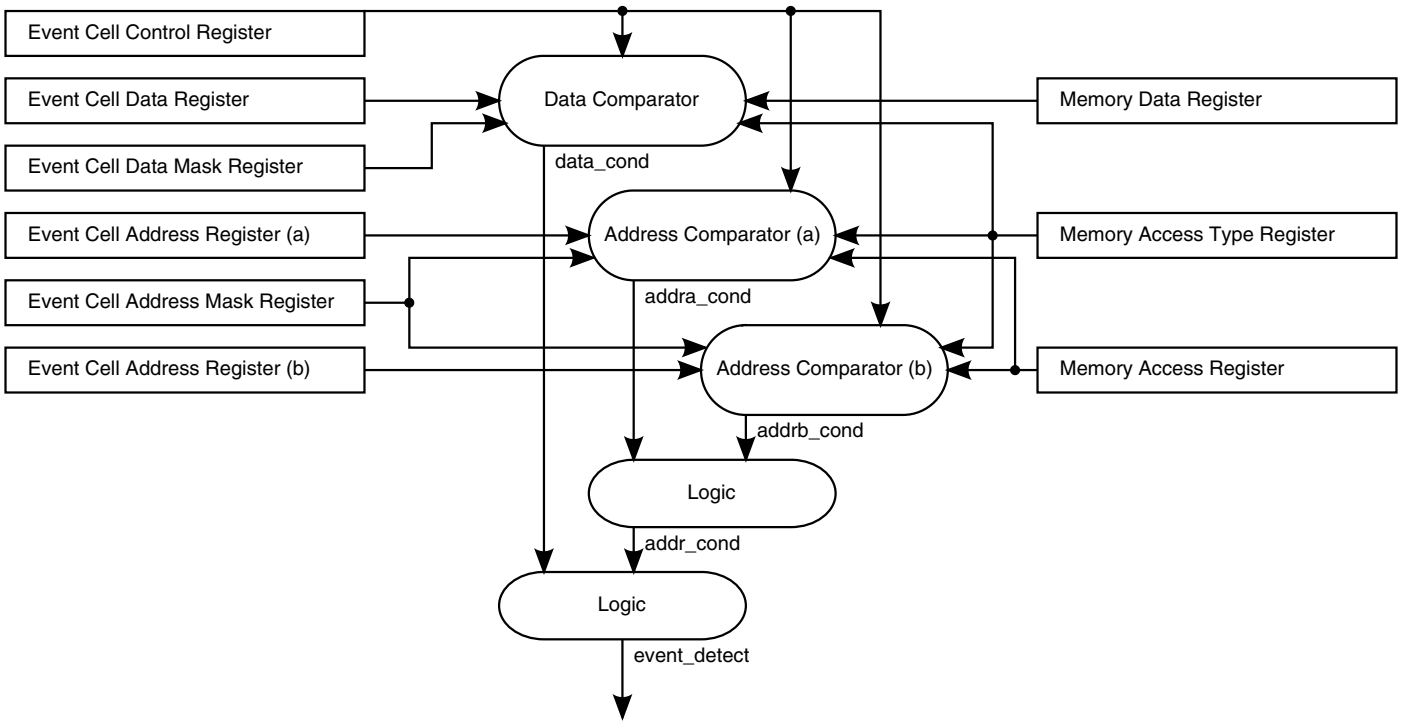


Figure 46-15. Event Cell Architecture

To define a memory breakpoint, three conditions are taken into account: The first two conditions are comparisons of the current memory address with user-defined reference addresses (these conditions are called addressA and addressB). The third condition consists of a comparison between the data received on the DMBUS and a user-defined reference data (this condition is called data). An intermediate address condition is set to express a dependency between addressA and addressB conditions.

46.16.7 Clock Gating and Reset

This section details how to use the clocks and handle the reset signals.

46.16.7.1 Clocks

Because the SDMA uses clock gating to save power, it is necessary to disable the clock gating and force the clocks to be enabled when using the OnCE.

When the OnCE is accessed through its JTAG interface, clock gating must be disabled outside the SDMA via a dedicated SDMA input port clk_gating_off. The reason why detection is not performed automatically by the SDMA internal hardware is that it would cost power to monitor activity on the JTAG interface.

This clk_gating_off input is controlled by a control bit in the System JTAG Controller. Refer to the System JTAG Controller chapter for more information.

When the OnCE is accessed through the ARM platform Control interface, clock gating is automatically turned off. This is done when bit 0 of the ONCE_ENB register (see OnCE Enable (SDMAARM_ONCE_ENB)) is set. A write access to this register is possible even when the OnCE clock is not running. If the ARM platform access is used, the bit in the ONCE_ENB register must be set before any attempt to access any other OnCE register.

46.16.7.2 Resets

The OnCE reset is different from the SDMA main reset. The OnCE reset is connected to the TRST_B pin.

Normally, activating the SDMA reset while keeping the OnCE reset inactive (when possible) enables you to reset the core without having to reprogram the OnCE.

46.16.8 Real Time Features

To rebuild the skeleton of a program execution, it is necessary to store the addresses of the program instructions where jumps are taken: A trace buffer is therefore provided. A real time buffer has also been added to receive data values written during a program execution.

The content of this register may be exported through JTAG ports without stopping the core.

46.16.8.1 Trace Buffer

The Trace Buffer is a 32-stage buffer that contains appropriate information to identify the 32 last changes of flow detected during a program execution.

The following figure shows an overview of the Trace Buffer.

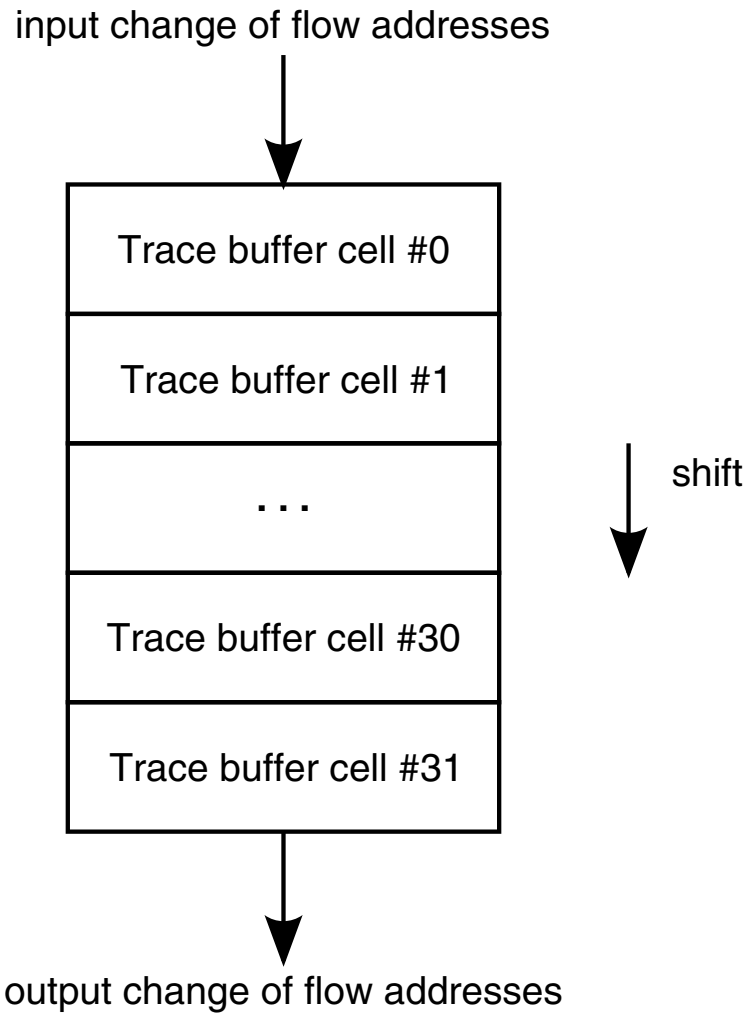


Figure 46-16. Trace Buffer

Each cell of the trace buffer contains two reference addresses and a flag. The flag is set when the addresses stored in the cell correspond to a valid change of flow; otherwise, the flag is cleared. The three most significant bits are unused.

After every change of flow detection, the address of current instruction and the address of the target instruction are stored at the top of the Trace Buffer (cell #0). The flag in the cell is set to indicate that a valid change of flow was detected. Former cell values are shifted one level down. The Trace Buffer contains the 32 last changes of flow. All the flags are reset on a software or a hardware reset, and after each transition from debug mode to user mode.

A memory mapped register of SDMA core, the Trace Buffer register (TB), is provided to read the content of the Trace Buffer. This operation should be done in debug mode. Performing a read access to the Trace Buffer register returns the content of the bottom of the Trace Buffer (cell #31). After every read access, the trace buffer is shifted one level down, and the flag at the top of the trace buffer is cleared.

A typical OnCE command sequence that retrieves the oldest change-of-flow information is a follows:

```
exec_once("mov r1, TB");                              // stores the oldest change-of-flow in
GReg1
dmov(-);                                          // retrieves GReg1 contents
```

This sequence requires the SDMA to be put in debug mode.

46.16.8.2 Real Time Buffer

The Real Time Buffer register (RTB) is a memory mapped register that can be accessed as a regular memory location by the SDMA core during program execution. This register is located in the OnCE.

Executing ar rbuffer command (see The OnCE Controller for further details) exports the content of this register through JTAG ports.

When a write access is performed at the memory location corresponding to the RTB, the receive flag (for example, the RCV bit) is set in the OnCE Status Register (OSR). This flag is cleared at the end of the execution of a rbuffer command.

NOTE

Every write access to the RTB memory location updates the RTB register even if the RCV flag is set. The RTB is cleared on a JTAG reset.

46.16.8.3 Emulation Pin

The debug_matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit.

Since it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

46.16.8.4 Real-Time Debug Outputs

The folloing table shows the debug signals that are available at the SDMA boundaries. Their availability at chip boundaries depends on the project.

Table 46-44. Real-Time Debug Output Pins

Pin	Description
debug_core_state[3:0]	The core_state bits reflect the state of the SDMA core. • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the core idle state); the "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 is executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel: they are used when entering debug mode after reset; the reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Context Switch Saving Channel 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change of Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Context Switch Restoring Channel
debug_yield	Pulse that is active when a yield (done 0) or a yieldge (done 1) instruction is executed. 0 - 1 yield/yieldge executed

Table continues on the next page...

Table 46-44. Real-Time Debug Output Pins (continued)

Pin	Description
debug_core_run	Active when the SDMA core is executing instructions. 0 Debug or sleep mode 1 Run mode
debug_event_channel_sel	Indicates if debug_event_channel displays current channel or last received event 0- debug_event_channel[5:0] gives the number of the current channel 1- debug_event_channel[5:0] gives the number of the last received event
debug_event_channel[5:0]	Gives the number of any DMA request as soon as it is received or the number of the current channel. The value of debug_event_channel_sel indicates if debug_event_channel displays the current channel or last received event. The signal debug_event_channel_sel must be observed to determine what information is provided on debug_event_chanel at any given time.
debug_pc[13:0]	Program Counter value; it has a meaning when the core is in run mode.
debug_mode	Set when the core is in debug. 0 - 1 Core is in debug
debug_bus_error	Set when an error was received during a load or a store (ld, st, ldf, or stf instruction) and registered in SF or DF flag. 0 No error during last load/store 1 Error during last load/store

Table continues on the next page...

Table 46-44. Real-Time Debug Output Pins (continued)

Pin	Description
debug_bus_device[4:0]	Indicates the device or functional unit that is accessed by the current instruction. The debug_bus_device output is always valid when in sleep mode, debug mode, or executing any instruction that does not access the functional units or the memory mapped devices, "no access" is output. 0 No access 1 MSA 2 MDA 3 MD 4 MS 5 PSA 6 PDA 7 PD 8 PS 9 RESERVED 10 RESERVED 11 RESERVED 12 RESERVED 13 CA 14 CS 15 Reserved 16 Memory (RAM or ROM) 17 Memory mapped register 18 Peripheral #1 19 Peripheral #2 20 Peripheral #3 21 Peripheral #4 22 Peripheral #5 23 Peripheral #6 24 Peripheral #7 25 Peripheral #8 26 Peripheral #9 27 Peripheral #10 28 Peripheral #11 29 Peripheral #12 30 Peripheral #13 31 Peripheral #14

Table continues on the next page...

Table 46-44. Real-Time Debug Output Pins (continued)

Pin	Description
debug_bus_rwb	Indicates the direction of the access given by debug_bus_device 0 Write access (st or stf) 1 Read access (ld or ldf)
debug_matched_dmbus	Pulse indicating the OnCE event detection unit has detected a match on the data bus during an access to memory (RAM or ROM), a memory mapped register or a peripheral that is hooked to the SDMA. 0 - 1 data bus match detected
debug_rtbuffer_write	Pulse indicating when the real-time buffer is written by the core. 0 - 1 RTB was modified
debug_evt_chn_lines[7:0]	Eight lines that generate short pulses when DMA requests are received or channels are (re)started. Every line is controlled through two parameters defined in registers Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1) (as described in SDMAARM). The following two parameters are available for every line: • CNF-Indicates what is monitored on the line: 0 for a channel start, 1 for a DMA request reception • NUM[5:0]-Gives the number of the DMA request or channel to monitor

The matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit. Because it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

All real-time debug outputs are disabled by default (for example, they are stuck to 0) to avoid power consumption when they are not used. They are enabled when bit 11 (RTDOBS) of the Configuration Register (SDMAARM_CONFIG) is set. Signals provided to the system JTAG controller for SDMA debug mode status will also be enabled when the *clk_gating_off* input is asserted.

46.17 Instruction Set

46.17.1 Instruction Encoding

This section presents a short summary of the instruction codes. All context switch instructions are listed for information only; they cannot function properly out of the context switch routine.

```
x...x - don't care

rrr - destination/source general register

sss - additional source general register

bbb - general register used as address base register

ddddd - address displacement

nnnnn - bit number
uuuuuuuu - function unit command bits

pppppppp - branch displacement (signed)

iiiiiiii - 8-bit immediate

jjj - control bit to clear

ff - flag to clear
00000jjj00000000        - done (done,yield,wait)
00000jjj00000001        - notify
00000xxx00000010        - reserved
00000xxx00000011        - reserved
00000xxx00000100        - reserved
0000000000000101     - softBkpt
0000000100000101     - reserved
0000001000000101     - reserved
0000001100000101     - reserved
0000010000000101     - reserved
0000010100000101     - reserved
0000011000000101     - reserved
0000011100000101     - reserved
0000000000000110     - ret
0000000100000110     - reserved
0000001000000110     - reserved
0000001100000110     - reserved
0000010000000110     - reserved
0000010100000110     - reserved
0000011000000110     - reserved
0000011100000110     - reserved
000000ff00000111       - clrf ff
0000010000000111     - reserved
0000010100000111     - reserved
0000011000000111     - reserved
0000011100000111     - illegal
00000rrr00001000        - jmpr r
00000rrr00001001        - jsrr
00000rrr00001010        - ldrpc r
00000rrr00001011        - reserved
00000rrr000011xx          - reserved
00000rrr00010000        - revb
00000rrr00010001        - revblo
00000rrr00010010        - rorb
00000rrr00010011        - reserved
00000rrr00010100        - ror1
00000rrr00010101        - lsr1
00000rrr00010110        - asr1
00000rrr00010111        - lsl1
00000rrr001nnnnn        - bclri r,n
00000rrr010nnnnn        - bseti r,n
00000rrr011nnnnn        - btsti r,n
00000xxx10000xxx           - reserved
00000rrr10001sss           - mov
00000rrr10010sss           - xor
00000rrr10011sss           - add
00000rrr10100sss           - sub
00000rrr10101sss           - or
00000rrr10110sss           - andn
```

```
00000rrr10111sss           - and
00000rrr11000sss           - tst
00000rrr11001sss           - cmpeq
00000rrr11010sss           - cmplt
00000rrr11011sss           - cmphs
0000011011100000     - reserved
0000011011100001     - reserved
0000011011100010     - cpShReg
0000011011100011     - reserved
0000011011100100     - reserved
0000011011100101     - reserved
0000011011100110     - reserved
0000011011100111     - reserved
00000xxx11101xxx           - reserved
00000xxx11110xxx           - reserved
00000xxx11111xxx           - reserved
00001rrriiiiiiii                - ldi r,i
00010rrriiiiiiii                - xori r,i
00011rrriiiiiiii                - addi r,i
00100rrriiiiiiii                - subi r,i
00101rrriiiiiiii                - ori r,i
00110rrriiiiiiii                - andni r,i
00111rrriiiiiiii                - andi r,i
01000rrriiiiiiii                - tsti r,i
01001rrriiiiiiii                - cmpeqi r,i
01010rrrddddddbbb         - ld r,(d,b)
01011rrrddddddbbb         - st r,u
01100rrruuuuuuuuu         - ldf r,u
01101rrruuuuuuuuu         - stf r,u
011100xxxxxxxxxx                - reserved
011101xxxxxxxxxx                - reserved
011110ffnnnnnnnn        - Loop ff flags are reset
01111100pppppppp      - bf pc=pc+signed(pppppppp)+1
01111101pppppppp      - bt pc=pc+signed(pppppppp)+1
01111110pppppppp      - bsf pc=pc+signed(pppppppp)+1
01111111pppppppp      - bdf pc=pc+signed(pppppppp)+1
10aaaaaaaaaaaaaa      - jmp absolute
11aaaaaaaaaaaaaa      - jsr absolute
```

46.17.2 SDMA Instruction Set

This section describes all the useful instructions from the SDMA set.

Table 46-45. SDMA Instruction List

Instruction	Description	Page
ADD	Addition	ADD (Addition)
ADDI	Add with Immediate Value	ADDI (Add with Immediate Value)
AND	Logical AND	AND (Logical AND)
ANDI	Logical AND with Immediate Value	ANDI (Logical AND with Immediate Value)
ANDN	Logical AND NOT	ANDN (Logical AND NOT)

Table continues on the next page...

Table 46-45. SDMA Instruction List (continued)

Instruction	Description	Page
ANDNI	Logical AND with Negated Immediate Value	ANDNI (Logical AND with Negated Immediate Value)
ASR1	Arithmetic Shift Right by 1 Bit	ASR1 (Arithmetic Shift Right by 1 Bit)
BCLRI	Bit Clear Immediate	BCLRI1 (Bit Clear Immediate)
BDF	Conditional Branch if Destination Fault	BDF (Conditional Branch if Destination Fault)
BF	Conditional Branch if False	Functional Units Programming Model
BSETI	Bit Set Immediate	BSETI (Bit Set Immediate)
BSF	Conditional Branch if Source Fault	BSF (Conditional Branch if Source Fault)
BT	Conditional Branch if True	BT (Conditional Branch if True)
BTSTI	Bit Test immediate	BTSTI (Bit Test immediate)
CLRF	Clear ARM platform flags	CLRF (Clear ARM platform flags)
CMPEQ	Compare for Equal	CMPEQ (Compare for Equal)
CMPEQI	Compare with Immediate for Equal	CMPEQI (Compare with Immediate for Equal)
CMPHS	Compare for Higher or Same	CMPHS (Compare for Higher or Same)
CMPLT	Compare for Less Than	CMPLT (Compare for Less Than)
cpShReg	Update Context of PCU Registers and Flags	cpShReg (Update Context of PCU Registers and Flag)
DONE	DONE, Yield	DONE (DONE, Yield)
ILLEGAL	ILLEGAL Instruction	ILLEGAL (ILLEGAL Instruction)
JMP	Unconditional Jump Immediate	JMP (Unconditional Jump Immediate)
JMPR	Unconditional Jump	JMPR (Unconditional Jump)
JSR	Unconditional Jump to Subroutine Immediate	JSR (Unconditional Jump to Subroutine Immediate)
JSRR	Unconditional Jump to Subroutine	JSRR (Unconditional Jump to Subroutine)

Table continues on the next page...

Table 46-45. SDMA Instruction List (continued)

Instruction	Description	Page
LD	Load Register	LD (Load Register)
LDF	Load Register from Functional Unit	LDF (Load Register from Functional Unit)
LDI	Load Register with Immediate Value	LDI (Load Register with Immediate Value)
LDRPC	Load from RPC to Register	LDRPC (Load from RPC to Register)
LOOP	Hardware Loop	LOOP (Hardware Loop)
LSL1	Logical Shift Left by 1 Bit	LSL1 (Logical Shift Left by 1 Bit)
LSR1	Logical Shift Right by 1 Bit	LSR1 (Logical Shift Right by 1 Bit)
MOV	Logical Move	MOV (Logical Move)
NOTIFY	Notify to ARM platform	NOTIFY (Notify to ARM platform)
OR	Logical OR	OR (Logical OR)
ORI	Logical OR with Immediate Value	ORI (Logical OR with Immediate Value)
RET	Return from Subroutine	RET (Return from Subroutine)
REVB	Reverse Byte Order	REVB (Reverse Byte Order)
REVBLO	Reverse Low Order Bytes	Reverse Low Order Bytes(REVBLO)
ROR1	Rotate Right by 1 Bit	ROR1 (Rotate Right by 1 Bit)
RORB	Rotate Right by 1 Byte	RORB (Rotate Right by 1 Byte)
SOFTBKPT	Software Breakpoint	SOFTBKPT (Software Breakpoint)
ST	Store Register	ST (Store Register)
STF	Store Register in Functional Unit	STF (Store Register in Functional Unit)
SUB	Subtract	SUB (Subtract)
SUBI	Subtract with Immediate	SUBI (Subtract with Immediate)
TST	Test with Zero	TST (Test with Zero)
TSTI	Test Immediate	TSTI (Test Immediate)

Table continues on the next page...

Table 46-45. SDMA Instruction List (continued)

Instruction	Description	Page
XOR	Logical Exclusive OR	XOR (Logical Exclusive OR)
XORI	Exclusive OR with Immediate	XORI (Exclusive OR with Immediate)

46.17.2.1 ADD (Addition)

Operation:

```
GReg[r] ← GReg[s] + GReg[r]

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: add r,s

Example: add 0,3
```

ADD GReg[3] and GReg[0] and store the result in GReg[0]

CPU Flags: T

Cycles: 1

Description: Performs the ADDition of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*. The T flag is set if the result of the operation is 0. It is cleared if the result is not 0.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]
```

```
100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.2 ADDI (Add with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] + immediate

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: addi r,immediate

Example: add 6,112
```

ADD GReg[6] and decimal value 112 and store the result in GReg[6]

CPU Flags: T

Cycles: 1

Description: Adds a 0-extended immediate value to a general register; stores the result in the general register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]
```

```
111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

 ...

11111110 - 254

11111111 - 255
```

46.17.2.3 AND (Logical AND)

Operation:

```
GReg[r] ← GReg[s] & GReg[r]
```

Assembler:

```
Syntax: and r,s

Example: and 1,2
```

AND GReg[1] and GReg[2] and store the result in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]
```

110 - GReg[6]

111 - GReg[7]

46.17.2.4 ANDI (Logical AND with Immediate Value)

Operation:

GReg[r] ← GReg[r] & immediate

Assembler:

Syntax: andi r,immediate

Example: andi 7,45

AND GReg[7] and decimal value 45 and store the result in GReg[7]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between a 0-extended immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]

iiiiiiii - immediate value:

00000000 - 0

```
00000001 - 1

...

11111110 - 254

11111111 - 255
```

46.17.2.5 ANDN (Logical AND NOT)

Operation:

```
GReg[r] ← ~GReg[s] & GReg[r]
```

Assembler:

```
Syntax:andn r,s

Example: andn 3,4
```

AND GReg[3] and NOT GReg[4] (bit inverted) and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the AND of the negation of the source general register *s* and the destination general register *r*, and stores the result in the destination general register *r*.

Instruction Format:

Table 46-46. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	1	0	s	s	s

Instruction Fields:

rrr /sss - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.6 ANDNI (Logical AND with Negated Immediate Value)

Operation:

```
GReg[r] ← GReg[r] & ~immediate
```

Assembler:

```
Syntax: andni r,immediate

Example: andni 0,2
```

AND GReg[0] and decimal value -3 (inverted 32-bit value 2) and store the result in GReg[0]

CPU Flags: unaffected

Cycles: 1

Description: Performs an AND between the negation of a 0-extended 8-bit immediate value and a general register; stores the result in the general register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1
```

```
...

11111110 - 254

11111111 - 255
```

46.17.2.7 ASR1 (Arithmetic Shift Right by 1 Bit)

Operation:

```
GReg[r]:{b31,b30,...,b1,b0} ← GReg[r]:{b31,b31,b30,...,b1}
```

Assembler:

```
Syntax: asr1 r

Example: asr1 3
```

divide by 2 the signed value of GReg[3] and store the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any general register to the right and keep the same sign: The left bit (bit 31) is kept untouched.

Instruction Format:

Table 46-47. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.8 BCLRI1 (Bit Clear Immediate)

Operation:

```
GReg[r]:{b31,...,b(i+1),0,b(i-1),...,b0} ← GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

```
Syntax: bclri r,i

Example: bclri 1,12
```

clear bit 12 in GReg[1]

CPU Flags: Unaffected

Cycles: 1

Description: Clear the bit of register r specified by the 5-bit immediate field

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	1	i	i	i	i	i

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - immediate value:

```
00000 - 0

00001 - 1

...

11110 - 30

11111 - 31
```

46.17.2.9 BDF (Conditional Branch if Destination Fault)

Operation:

```
if (DF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

```
Syntax:bdf label

Example: bdf LLL
```

Jump to LLL if DF is set, or go to the next instruction if DF is cleared; the displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: If flag DF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag DF is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

46.17.2.10 BF (Conditional Branch if False)

Operation:

```
if (T == 0)

PC ← PC + 1 + displacement

else

PC ← PC + 1
```

Assembler:

```
Syntax: bf label

Example: bf LLL
```

Jump to LLL if T is cleared, or go to the next instruction if T is set. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is cleared, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is set, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	0	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

46.17.2.11 BSETI (Bit Set Immediate)

Operation:

```
GReg[r]:{b31,...,b(i+1),1,b(i-1),...,b0} ← GReg[r]:{b31,...,b(i+1),b(i),b(i-1),...,b0}
```

Assembler:

```
Syntax: bseti r,i

Example: bseti 6,5
```

Set bit 5 in GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Sets bit number i in the selected General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	0	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - bit number field:

```
00000 - 0

00001 - 1

...

11110 - 30

11111 - 31
```

46.17.2.12 BSF (Conditional Branch if Source Fault)

Operation:

```
if (SF == 1) PC ← PC + 1 + displacement else PC ← PC + 1
```

Assembler:

```
Syntax: bsf label

Example: bsf LLL
```

Jump to LLL if SF is set, or go to the next instruction if SF is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag SF is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag SF is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	0	p	p	p	p	p	p	p	p

Instruction Fields:

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110 - 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 - (-2)

11111111 - (-1)
```

46.17.2.13 BT (Conditional Branch if True)

Operation

```
if (T == 1)

PC ← PC + 1 + displacement

else

PC ← PC + 1
```

Assembler

```
Syntax: bt label

bt LLL
```

Jump to LLL if T is set, or go to the next instruction if T is cleared. The displacement value is calculated by the assembler.

CPU Flags: Unaffected

Cycles: 2 when the branch is done, 1 otherwise

Description: Conditional branch: If flag T is set, jump to the new address that is calculated by adding the sign-extended 8-bit displacement to the next PC address. If flag T is cleared, no jump is performed: The next instruction is located at the next PC address.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	1	p	p	p	p	p	p	p	p

pppppppp - signed displacement field:

```
00000000 - 0

00000001 - 1

...

01111110- 126

01111111 - 127

10000000 - (-128)

10000001 - (-127)

...

11111110 -  (-2)

11111111 - (-1)
```

46.17.2.14 BTSTI (Bit Test immediate)

Operation:

```
T ← GReg[r]:b(i)
```

Assembler:

```
Syntax: btsti r,i

Example: btsti 2,29
```

Test bit 29 in GReg[2] and copy its value in flag T

CPU flags: T

Cycles: 1

Description: T is loaded with the value of bit number i from the selected general register.

Instruction Format:

Table 46-48. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	1	1	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiii - bit number field:

```
0000 - 0

0001 - 1

...

11110 - 30
```

```
11111 - 31
```

46.17.2.15 CLRF (Clear ARM platform flags)

Operation:

```
 if (ff%2 == 0)

SF ← 0

if (ff/2 == 0)

DF ← 0
```

Assembler:

```
Syntax: clrf ff

Example: clrf 2
```

Clear flag SF and keep flag DF unchanged

CPU Flags: SF, DF

Cycles: 1

Description: Clears a selection of the ARM platform fault flags: SF, DF, both SF and DF or none can be cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	f	f	0	0	0	0	0	1	1	1

Instruction Fields:

ff - flags field:

```
00 - clear SF and clear DF

01 - clear DF

10 - clear

SF 11 - no clear
```

46.17.2.16 CMPEQ (Compare for Equal)

Operation:

```
T ← (GReg[s] == GReg[r])
```

Assembler:

```
Syntax: cmpeq r,s

Example: cmpeq 7,5
```

Compare GReg[7] and GReg[5] and set flag T if they are equal

CPU flags: T

Cycles: 1

Description: Subtracts the destination general register *r* from the source general register *s*, and sets T if the result is 0, clears T if the result is not 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.17 CMPEQI (Compare with Immediate for Equal)

Operation:

```
T ← (GReg[r] == immediate)
```

Assembler:

```
Syntax: cmpeqi r,immediate

Example: cmpeqi 2,13
```

Compare GReg[2] and decimal value 13 and set flag T if they are equal

CPU Flags: T

Cycles: 1

Description: Subtracts the 0-extended 8-bit immediate value from the general register, and sets T if the result is 0, clears T if the result is not 0. The immediate value is the low-order byte of the instruction.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

46.17.2.18 CMPHS (Compare for Higher or Same)

Operation:

```
T ← (GReg[r] ≥ GReg[s])
```

Assembler:

```
Syntax: cmphs r,s

Example: cmphs 0,1
```

Compare GReg[0] and GReg[1] and set flag T if GReg[0] is higher than or equal to GReg[1]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is higher than or equal to the source general register *s*, clears T otherwise. The comparison is unsigned.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.19 CMPLT (Compare for Less Than)

Operation:

```
T ← (GReg[r] < GReg[s])
```

Assembler:

```
Syntax: cmplt r,s

Example: cmplt 7,4
```

Compare GReg[7] and GReg[4] and set flag T if GReg[7] is lower than GReg[4]

CPU Flags: T

Cycles: 1

Description: Compares the destination general register *r* and the source general register *s*, and sets T if the destination general register *r* is lower than the source general register *s*, clears T otherwise. The comparison is signed.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	1	0	s	s	s

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.20 cpShReg (Update Context of PCU Registers and Flag)

Assembler:

```
Syntax: cpShReg
```

CPU Flags: none

Cycles: 1

Description: SF, RPC, T, PC,LM, EPC, DF, and SPC registers are updated according to the value of their corresponding bits in the context memory. This instruction must only be used in debug mode via the OnCE. It reverses the done *5* operation.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0	1	1	1	0	0	0	1	0

46.17.2.21 DONE (DONE, Yield)

Operation:

```
if (jjj&6 == 2) HE[CCR] ← 0

if (jjj == 3) HI[CCR] ← 1

if (jjj == 4) EP[CCR] ← 0


if ((jjj == 0) && (NCP > CCP)) CCR ← NCR

else if ((jjj == 1) && (NCP >= CCP))

CCR ← NCR

else

CCR ← NCR
```

(CCR stands for Current Channel Register; NCR stands for Next Channel Register)

Assembler:

```
Syntax: done jjj

Example: done 3
```

Clear HE bit for the current channel, send an interrupt to the ARM platform for the current channel and reschedule.

CPU Flags: Unaffected

Cycles: Variable if a context switch is done, 1 otherwise

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required. Sends an interrupt to the corresponding ARM platform by setting the appropriate flag, if required (HI for the corresponding channel number). Reschedules according to the mode and the NCP (Next Channel Priority) and CCP (Current Channel Priority) values. According to the scheduling decision, the NCR (Next Channel Register) is copied to the CCR (Current Channel Register) and channel contexts are switched. If several channels with the same highest priority are pending, they are ordered by their number from 31 down to 0. The higher number is selected (for example, channel 26 is selected if channels 3, 12, 14, and 26 with the same highest priority are pending). If no flag is modified, the reschedule can allow the replacement of the current channel by another channel with a priority strictly greater than the current channel priority (yield). Or, it can allow the replacement of the current channel by another channel with a priority greater than or equal to the current channel priority (yieldge). In the latter case, the selected channel will always be the first one with the same priority, starting from channel number 31 down to channel 0 (the current channel does not belong to the set of selectable channels).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	0

jjj - Channel Flags field:

000 - No channel flags affected: Reschedule only if the next channel priority is greater than current channel priority (yield)

001 - No channel flags affected: Reschedule only if the next channel priority is greater than or equal to the current channel priority (yieldge)

010 - Clear HE for the current channel and reschedule 011 - Clear HE, set HI for the current channel and reschedule 100 - Clear EP for the current channel and reschedule

101 - Reserved for debug to copy relevant registers into context memory

110 - RESERVED

111 - RESERVED

For the scheduling rules, refer to Scheduler Functional Description. Every possible done instruction is further described as follows:

- done 0/yield is executed by a channel script when it accepts preemption by a higher priority channel;
- done 1/yieldge is executed by a channel script when it accepts preemption by a higher priority channel and it also accepts a roll-up with other channels that have the same priority;
- done 2 is executed by a channel script that was triggered by a ARM platform start via the Channel Start (SDMAARM_HSTART) register, when its task is completed and it requires termination;
- done 3 is executed by a channel script that was triggered by a ARM platform start via the Channel Start (SDMAARM_HSTART) register, when its task is completed, it requires termination and it needs to trigger an interrupt to the ARM platform upon closure;
- done 4 is executed by a channel script that was triggered by a DMA request, when its task is completed and it requires termination;
- done 5 is used in debug mode only; it copies the PCU registers and flags to the context memory of the current channel;

46.17.2.22 ILLEGAL (ILLEGAL Instruction)

Operation:

```
PC ← 0001
```

Assembler:

```
Syntax: illegal
```

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the Illegal instruction routine located at address 0001. All unauthorized instructions result in an Illegal instruction behavior; however, the ILLEGAL instruction must be used to guarantee software compatibility with future versions of the SDMA.

Instruction Format

Table 46-49. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

46.17.2.23 JMP (Unconditional Jump Immediate)

Operation:

```
PC ← absolute_address
```

Assembler:

```
Syntax: jmp label
```

```
Example: jmp LLL
```

The assembler translates the label to the exact address

CPU Flags:Unaffected

Cycles: 2

Description: Jumps to the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	a	a	a	a	a	a	a	a	a	a	a	a	a	a

aaaaaaaaaaaaa - address field:

```
00000000000000 - 0
```

```
00000000000001 - 1
```

```
...

11111111111110 - 16382

11111111111111 - 16383
```

46.17.2.24 JMPR (Unconditional Jump)

Operation:

```
PC ← GReg[r]
```

Assembler:

```
Syntax: jmpr r

Example: jmpr 0
```

Jump to address stored in GReg[0]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the absolute address contained in a General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.25 JSR (Unconditional Jump to Subroutine Immediate)

Operation:

```
RPC ← PC + 1

PC ← absolute_address
```

Assembler:

```
Syntax: jsr r

Example:jsr LLL
```

Jumps to subroutine starting at LLL; the assembler translates the label to exact address

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine located at the absolute address contained the lower 14 bits of the instruction (the PC is a 14-bit register).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	a	a	a	a	a	a	a	a	a	a	a	a	a	a

aaaaaaaaaaaaaa - address field:

```
00000000000000 - 0

00000000000001 - 1

...

11111111111110 - 16382

11111111111111 - 16383
```

46.17.2.26 JSRR (Unconditional Jump to Subroutine)

Operation:

```
RPC ← PC + 1

PC ← GReg[r]
```

Assembler:

```
Syntax: jsrr r

Example:jsrr 5
```

Jumps to subroutine located at address stored in GReg[5]

CPU Flags: Unaffected

Cycles: 2

Description: Jumps to the subroutine at address contained in a General Register

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	0	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.27 LD (Load Register)

Operation:

```
GReg[r] ← [GReg[b] + displacement]

if (transfer_error)

SF ← 1

else

SF ← 0
```

Assembler:

```
Syntax: ld r,(b,displacement)

Example: ld 1,(2,23)
```

Loads data into GReg[1]; the data is located at address obtained by adding decimal value 23 to GReg[2]

CPU Flags: SF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to fetch on the DM bus. The data received from the bus is stored in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	r	r	r	d	d	d	d	d	b	b	b

rrr / bbb - register field:

```
000 - GReg[0]

001 - GReg[1]

...

111 - GReg[7]
```

ddddd - displacement value:

```
00000 - 0

00001 - 1

...

11111 - 31
```

46.17.2.28 LDF (Load Register from Functional Unit)

Operation:

```
GReg[r] ← [fu_address]

if (transfer_error)

SF ← 1

else

SF ←0
```

fu_address is an 8-bit field and depends on addressed functional unit

Assembler:

```
Syntax: ldf r,fu_address

Example: ldf 0,13
```

Loads data coming from the Burst DMA register MD into GReg[0]; it is a 32-bit access with no prefetch

CPU Flags: SF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and stores the data received from the bus in the destination General Register r. If an error occurs during the transfer, the flag SF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the LDF instruction usage with each functional unit:

- Burst DMA Read (ldf) for Burst DMA
- Peripheral DMA Read (ldf)-Read Mode for Peripheral DMA
- Read Instruction (ldf) for CRC unit

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ffffffff - functional unit source register and action (unspecified values are reserved):

```
00000000 - MSA

00000100 - MDA

00001001 - MD byte

00001010 - MD halfword

00001011 - MD word
```

```
00001100 - MS

00101001 - MD byte - prefetch

00101010 - MD halfword - prefetch

00101011 - MD word - prefetch

01000000 - DSA


10000000 - CA

10000100 - CS (CRC checksum)

11000000 - PSA

11001000 - PD

11010000 - PDA

11011000 - PD in copy mode (rrr contents are lost)

11101000 - PD - prefetch next data

11111111 - PS
```

46.17.2.29 LDI (Load Register with Immediate Value)

Operation:

```
GReg[r] ← immediate
```

Assembler:

```
Syntax: ldi r,immediate

Example: ldi 6,1
```

loads decimal value 1 into GReg[6]

CPU Flags: Unaffected

Cycles: 1

Description: Stores a 0-extended immediate value in a General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

46.17.2.30 LDRPC (Load from RPC to Register)

Operation:

```
GReg[r] ← RPC
```

Assembler:

```
Syntax: ldrpc r

Example: ldrpc 3
```

copies RPC to GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Stores the contents of the RPC in a General Register. That instruction may be used to have more than one level of subroutines.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	0	1	0	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.31 LOOP (Hardware Loop)

Operation:

```
if (ff%2 == 0)
   SF ← 0
if (ff/2 == 0)
  DF ← 0
if ((GReg[0] == 0) || (SF == 1) || (DF == 1))
   PC ← PC + loop_size + 1
else
   {
      SPC ← PC + 1
      EPC ← PC + loop_size + 1
      LM ← 1
      PC ← PC + 1
   }
```

during every instruction execution in the loop:

```
if ((SF == 1) || (DF == 1))
 {
    LM ← 0
    PC ← EPC
  }
else if ((PC + 1) == EPC)
```

```
 {
    GReg[0] ← GReg[0] - 1
    if (GReg[0] == 0)
      {
         LM ← 0
         PC ← EPC
       }
     else
         PC ← SPC
   }
else
  PC ← nextPC(instruction)
```

after the execution of the last instruction of the loop body:

```
if (GReg[0] == 0)
   T ← 1
else
   T ← 0
```

Assembler:

```
Syntax: loop n{,ff}
Example: loop 3,1
```

Executes GReg[0] times the instructions comprised between PC+1 and PC+3 (included); ff=1 clears the DF flag before starting the loop. When omitted, the ff field is set to 0 (clearing both SF and DF).

CPU Flags: LM[1:0], T

Cycles: 2 when the loop count (GReg[0]) is 0 or SF or DF is set at loop start, 1+1 when the loop starts but exits abnormally (SF or DF set inside the loop which adds 1 cycle to the offending load or store to jump to EPC), 1 when the loop is executed normally

Description: The loop instruction executes a sequence of instructions several times. The number of times is given by the contents of GReg[0], the loop counter. SDMA will jump to the first instruction after the end of the loop if the value in GReg[0] is 0. Otherwise the SDMA enters loop mode. It sets the most significant bit of the LM flag that will only be reset once the last instruction of the last loop is executed. The instructions in the loop are executed GReg[0] times. The management of fault flags (SF and DF) is as follows. When entering the hardware loop, SF and DF can be cleared according to the ff field of the

instruction. After that operation, if any flag is still set the loop will not be executed. The SDMA will jump to the first instruction after the end of the loop without entering loop mode. During the execution of the loop, if any fault flag is set by a LD, LDF, ST, or STF instruction, the SDMA will immediately exit loop mode and jump to the first instruction after the end of the loop. In that case, GReg0 is not decremented for that last piece of the loop body execution (even if the SF or DF flag is set at the last instruction of the loop body). The T flag reflects the state of GReg[0] after the end of the loop, which is an indicator of the complete execution of the loop. If the loop exited because of an error (SF or DF set), GReg[0] will not be 0 at the end of the loop, hence T will be cleared. If the loop executes without fault, GReg[0] will be 0 at the end of the loop, hence T will be set. The boundary case when a source or destination fault occurs at the last instruction of the last loop is considered as an anticipated exit of the loop, which causes the T flag to be cleared. If the last instruction executed before leaving the hardware loop also tries to modify the T flag, the flag is updated according to the value of GReg[0], NOT according to the result of the last executed instruction.

Limitations:

1. 1. Jump instructions (JMP, JMPR, JSR, JSRR, BF, BT, BSF, BDF) are not allowed inside the hardware loop.
2. 2. GReg[0] cannot be written to inside the hardware loop (it can be read).
3. 3. The empty loop (0 instruction in the body) is forbidden.
4. 4. If GReg[0] == 0 at the start of the loop, which causes a jump to EPC, the T flag is not updated.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	f	f	n	n	n	n	n	n	n	n

Instruction Fields:

```
ff - flags field:

00 - clear SF and clear DF

01 - clear DF

10 - clear SF

11 - no clear
```

nnnnnnnn - loop size

```
00000000 - empty loop: forbidden value

00000001 - 1 instruction in the loop

00000010 - 2 instructions in the loop
```

```
...

11111111 - 255 instructions in the loop
```

46.17.2.32 LSL1 (Logical Shift Left by 1 Bit)

Operation:

```
GReg[r]:{b30,...,b1,b0,0} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: lsl1 r

Example: lsl1 2
```

multiplies by 2 the value in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the left. The right bit (bit 0) is set to 0. No overflow is detected by the hardware.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	1	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.33 LSR1 (Logical Shift Right by 1 Bit)

Operation:

```
GReg[r]:{0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: lsr1 r

Example: lsr1 4
```

divides by 2 the unsigned value contained in GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Shift the bits of any General Register to the right. The left bit (bit 31) is set to 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	1

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.34 MOV (Logical Move)

Operation:

```
GReg[r] ← GReg[s]
```

Assembler:

```
Syntax: mov r,s

Example: mov 4,0
```

copies GReg[0] to GReg[4]

CPU Flags: Unaffected

Cycles: 1

Description: Move the contents of the source General Register s to the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.35 NOTIFY (Notify to ARM platform)

Operation:

```
 if (jjj & 4 == 0)
 {
   if (jjj&2 == 2)
      HE[CCR] ← 0
   if (jjj&1== 1)
       HI[CCR] ← 1
  }
else if (jjj == 4)
   EP[CCR] ← 0
else
```

(CCR stands for Current Channel Register)

Assembler:

```
Syntax: notify jjj

Example: notify 3
```

clears the HE bit for the current channel and sends an interrupt to the Host for the current channel

CPU Flags: Unaffected

Cycles: 1

Description: Clears one of the channel enabling bits (HE or EP for the corresponding channel number) if required, sends an interrupt to the corresponding ARM platform by setting the appropriate flag if required (HI for the corresponding channel number).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	j	j	j	0	0	0	0	0	0	0	1

jjj - Channel Flags field:

```
000 - unused

001 - set HI for the current channel

010 - clear HE for the current channel

011 - clear HE, set HI for the current channel

100 - clear EP for the current channel

101 - RESERVED

110 - RESERVED

111 - RESERVED
```

46.17.2.36 OR (Logical OR)

Operation:

```
GReg[r] ← GReg[s] | GReg[r]
```

Assembler:

```
Syntax: or r,s

Example: or 3,6
```

ORs GReg[3] and GReg[6] and stores the result in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	1	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.37 ORI (Logical OR with Immediate Value)

Operation:

```
GReg[r] ← GReg[r] | immediate
```

Assembler:

```
Syntax: ori r,immediate

Example: ori 1,56
```

ORs GReg[1] and the decimal value 56 and stores the result in GReg[1]

CPU Flags: unaffected

Cycles: 1

Description: Performs an OR between a 0-extended 8-bit immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

46.17.2.38 RET (Return from Subroutine)

Operation:

```
PC ← RPC
```

Assembler:

```
Syntax: ret
```

CPU Flags: Unaffected

Cycles: 2

Description: Return from subroutine.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

46.17.2.39 REVB (Reverse Byte Order)

Operation:

```
GReg[r]:{B3,B2,B1,B0} ← GReg[r]:{B0,B1,B2,B3}
```

Assembler:

```
Syntax: revb r

Example:  revb 5
```

reverses bytes order in GReg[5]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse the byte order of any General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.40 Reverse Low Order Bytes(REVBLO)

Operation:

```
GReg[r]:{B3,B2,B0,B1} ← GReg[r]:{B3,B2,B1,B0}
```

Assembler:

```
Syntax: revblo r

Example: revblo 0
```

reverses low order bytes in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Reverse both low order bytes of any General Register.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	0	1

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.41 ROR1 (Rotate Right by 1 Bit)

Operation:

```
GReg[r]:{b0,b31,b30,...,b1} ← GReg[r]:{b31,b30,...,b1,b0}
```

Assembler:

```
Syntax: ror1 r

Example: ror1 3
```

rotates bits to the right in GReg[3]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bits of any General Register to the right.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	1	0	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.42 RORB (Rotate Right by 1 Byte)

Operation:

```
GReg[r]:{B0,B3,B2,B1} ← GReg[r]:{B3,B2,B1,B0}
```

Assembler:

```
Syntax: rorb r

Example: rorb 2
```

rotates bytes to the right in GReg[2]

CPU Flags: Unaffected

Cycles: 1

Description: Rotate the bytes of any General Register to the right.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	0	0	0	1	0	0	1	0

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.43 SOFTBKPT (Software Breakpoint)

Operation:

Stops the current script and enters debug mode

Assembler:

```
softbkpt
```

CPU Flags: Unaffected

Description: When the core executes this instruction, it has the same effect as receiving a debug request from the OnCE or via the external debug request input: the script execution halts, the PCU enters its debug state and waits for the OnCE commands that are described in OnCE and Real-Time Debug.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

46.17.2.44 ST (Store Register)

Operation:

```
[GReg[b] + displacement] ← GReg[r]

if (transfer_error)

 DF ← 1

 else

DF ← 0
```

Assembler:

```
Syntax: st r,(b,displacement)

Example: st 7,(0,9)
```

stores the value from GReg[7] into memory at address obtained by adding decimal value 9 to GReg[0]

CPU Flags: DF

Cycles: 2+n where n is 0 for ROM, RAM or memory mapped registers, and n is the number of wait-states of the peripheral for a peripheral access

Description: Adds a 5-bit 0-extended displacement to a base address in General Register b; the result is the address of the data to store on the DM bus. The data sent on the bus comes from the source General Register r. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	r	r	r	d	d	d	d	d	b	b	b

Instruction Fields:

rrr / bbb - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]
```

```
111 - GReg[7]
```

ddddd - displacement value:

```
00000 - 0

00001 - 1

...

11111 - 31
```

46.17.2.45 STF (Store Register in Functional Unit)

Operation:

```
[fu_address] ← GReg[r] 0

if (transfer_error) 0

DF ← 1 0

else 0

DF ← 0
```

fu_address is an 8-bit field

Assembler:

```
Syntax: stf r,fu_address

Example: stf 3,0x2B
```

stores the 32-bit contents of GReg[3] to the Burst DMA register MD; waits until the flush to external memory is completed

CPU Flags: DF

Cycles: 1+n where n is the number of wait-states that may be inserted by the functional unit

Description: Sends an 8-bit address on the Functional Unit Bus (FU bus) and sends the contents of the source General Register r on the bus. If an error occurs during the transfer, the flag DF is set, else it is cleared.

Table 46-50. Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	r	r	r	f	f	f	f	f	f	f	f

See the following sections for more details of the STF instruction usage with each functional unit:

- Burst DMA Write (stf) for Burst DMA
- Peripheral DMA Write (stf)-Write Mode for Peripheral DMA
- Write Instruction (stf) for CRC

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

ffffffff - functional unit destination register and action (unspecified values are reserved):

00000000 - MSA in incremented mode

00000100 - MDA in incremented mode

00001001 - MD byte

00001010 - MD halfword

00001011 - MD word

00001100 - clear MS error flag

00001111 - MS

00010000 - MSA in frozen mode

00010100 - MDA in frozen mode

00011000 - MD in copy mode - number of words in rrr

00100000 - MSA in incremented mode - start prefetch

00101000 - MD no data - flush

00101001 - MD byte - flush

00101010 - MD halfword - flush

00101011 - MD word - flush

00110000 - MSA in frozen mode - start prefetch

10000000 - CA

10000100 - init CRC accumulator

10010100 - CS byte - compute CRC

10010101 - CS byte - compute CRC

10010110 - CS halfword - compute CRC

10010111 - CS word - compute CRC

11000001 - PSA in frozen mode - 8-bit data width

11000010 - PSA in frozen mode - 16-bit data width

11000011 - PSA in frozen mode - 32-bit data width

11000101 - PSA in incremented mode - 8-bit data width

11000110 - PSA in incremented mode - 16-bit data width

11000111 - PSA in incremented mode - 32-bit data width

11001000 - PD

11001001 - PSA in decremented mode - 8-bit data width

11001010 - PSA in decremented mode - 16-bit data width

11001011 - PSA in decremented mode - 32-bit data width

11001100 - clear PS error flag

11001101 - PSA data width becomes 8-bit

11001110 - PSA data width becomes 16-bit

11001111 - PSA data width becomes 32-bit

11010001 - PDA in frozen mode - 8-bit data width

11010010 - PDA in frozen mode - 16-bit data width

11010011 - PDA in frozen mode - 32-bit data width

11010101 - PDA in incremented mode - 8-bit data width

11010110 - PDA in incremented mode - 16-bit data width

11010111 - PDA in incremented mode - 32-bit data width

11011001 - PDA in decremented mode - 8-bit data width

11011010 - PDA in decremented mode - 16-bit data width

11011011 - PDA in decremented mode - 32-bit data width

11011101 - PDA data width becomes 8-bit

11011110 - PDA data width becomes 16-bit

11011111 - PDA data width becomes 32-bit

11100001 - PSA in frozen mode - 8-bit data width - prefetch data

11100010 - PSA in frozen mode - 16-bit data width - prefetch data

11100011 - PSA in frozen mode - 32-bit data width - prefetch data

11100101 - PSA in incremented mode - 8-bit data width - prefetch data

11100110 - PSA in incremented mode - 16-bit data width - prefetch data

11100111 - PSA in incremented mode - 32-bit data width - prefetch data

11101001 - PSA in decremented mode - 8-bit data width - prefetch data

11101010 - PSA in decremented mode - 16-bit data width - prefetch data

11101011 - PSA in decremented mode - 32-bit data width - prefetch data

11101101 - PSA data width becomes 8-bit - prefetch data

11101110 - PSA data width becomes 16-bit - prefetch data

11101111 - PSA data width becomes 32-bit - prefetch data

11111111- PS

46.17.2.46 SUB (Subtract)

Operation:

```
GReg[r] ← GReg[r] - GReg[s]

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: sub r,s

Example: sub 4,7
```

SUBtracts GReg[7] from GReg[4] and stores the result in GReg[4]

CPU Flags: T

Cycles: 1

Description: Subtracts the source General Register s from the destination General Register r, and stores the result in the destination General Register r. The T flag is set if the result of the operation is 0; it is cleared if the result is not 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	1	0	0	s	s	s

Instruction Fields:

rrr / sss - register fields:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.47 SUBI (Subtract with Immediate)

Operation:

```
GReg[r] ← GReg[r] - immediate

T ← (GReg[r] == 0)
```

Assembler:

```
Syntax: sub r,immediate

Example: sub 1,255
```

SUBtracts decimal value 255 from GReg[1] and stores the result in GReg[1]

CPU Flags: T

Cycles: 1

Description: Subtracts a 0-extended 8-bit immediate value from a General Register; stores the result in the General Register. The flag T is set when the result of the operation is 0; otherwise, it is cleared. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

46.17.2.48 TST (Test with Zero)

Operation:

```
T ← ((GReg[s] & GReg[r]) != 0)
```

Assembler:

```
Syntax: tst r,s

Example: tst 2,3
```

ANDs GReg[2] and GReg[3] and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of the source General Register s and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	1	0	0	0	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.49 TSTI (Test Immediate)

Operation:

```
T ← ((GReg[r] & immediate) != 0)
```

Assembler:

```
Syntax: tsti r,immediate

Example: tsti 5,13
```

ANDs GReg[5] and decimal value 13 and sets T if the result is non-null

CPU Flags: T

Cycles: 1

Description: Performs the AND of a 0-extended 8-bit immediate value and the destination General Register r, and sets T if the result is not 0, clears T if the result is 0. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - destination register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

46.17.2.50 XOR (Logical Exclusive OR)

Operation:

```
GReg[r] ← GReg[s] ^ GReg[r]
```

Assembler:

```
Syntax: xor r,s

Example: xor 0,3
```

XORs GReg[0] and GReg[3] and stores the result in GReg[0]

CPU Flags: Unaffected

Cycles: 1

Description: Performs the eXclusive OR of the source General Register s and the destination General Register r, and stores the result in the destination General Register r.

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	r	r	r	1	0	0	1	0	s	s	s

Instruction Fields:

rrr / sss - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

46.17.2.51 XORI (Exclusive OR with Immediate)

Operation:

```
GReg[r] ← GReg[r] ^ immediate
```

Assembler:

```
Syntax: xori r,immediate

Example: xor 7,5
```

XORs GReg[5] and decimal value 5 and stores the result in GReg[7]

CPU Flags: Unaffected

Cycles: 1

Description: Performs an eXclusive OR between a 0-extended 8-bit immediate value and a General Register; stores the result in the General Register. The immediate value is the low-order byte of the instruction and has a maximum value of 255 (0xFF).

Instruction Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	r	r	r	i	i	i	i	i	i	i	i

Instruction Fields:

rrr - register field:

```
000 - GReg[0]

001 - GReg[1]

010 - GReg[2]

011 - GReg[3]

100 - GReg[4]

101 - GReg[5]

110 - GReg[6]

111 - GReg[7]
```

iiiiiiii - immediate value:

```
00000000 - 0

00000001 - 1

...

11111110 - 254

11111111 - 255
```

46.17.2.52 YIELD, YIELDGE (DONE, Yield)

By default, unsupported assembler syntax. Can be aliased to the corresponding done instructions (yield = done 0; yieldge = done 1). Refer to the done instruction description DONE (DONE, Yield) .

46.18 Software Restrictions

46.18.1 Unsupported Burst DMA Access Sequence

The SDMA does not support triggering a pre-fetch followed by a flush of the Burst DMA without reading or writing any data. If the flush occurs while the background pre-fetch DMA operation is still in progress, it could result in un-defined behavior.

An example of the sequence which could result in undefined results is shown in the following example:

Instruction sequence not supported

```
          stf r1, MSA|PF              ; Update source address, triggers data pre-fetch in the
background
          mov R0,R0                ; Execute multiple assembly instructions, none of which
read
          mov R0,R0                ; or write data to/from MD
          stf MD|SZ0|FL               ; Flush FIFO without writing data. If the pre-fetch is
still in
                                  ; progress when this instruction is executed, there could be
```

; undefined operation.

A work-around to avoid any undesirable results is to first read MD to ensure the pre-fetch is complete before the flush is attempted.

Work-Around to previous example

```
          stf r1, MSA|PF              ; Update source address, triggers data pre-fetch
          mov R0,R0                ; Execute multiple assembly instructions, none of which
read
          mov R0,R0                ; or write data to/from MD
          ldf r2, MD                 ; dummy read of MD to ensure pre-fetch is complete before
the
                                  ; next instruction
```

stf MD|SZ0|FL ; Flush FIFO without writing data.

46.19 Application Notes

46.19.1 Data Structures for Boot Code and Channel Scripts

SDMA boot code downloads the different channel contexts and the scripts that will be executed on SDMA channels during the application. The boot code is run after reset when channel 0 is started by the ARM platform.

The boot code is also known as channel 0 script.

The boot code is based on the Channel Control Block (CCB) and Buffer Descriptor (BD) mechanisms that are data structures located into the ARM platform memory space. With these data structures, it is possible to instruct SDMA to download scripts and contexts but also to dump a context or a script to a destination data buffer. Channel scripts also use the CCB and BD data structures to pass instructions and/or pointers to data to be copied.

The format, processing, and field definition of the CCB and BD are defined and performed entirely by the software script rather than the SDMA hardware. An overview of the format and structure is provided here, but for complete details refer to the SDMA software documentation (see SDMA Scripts).

The CCB and BD data structures are accessed by SDMA using DMA and processed by the SDMA scripts. The ROM contains common sub-routines for processing these data structures which may be called by the bootload and channel scripts.

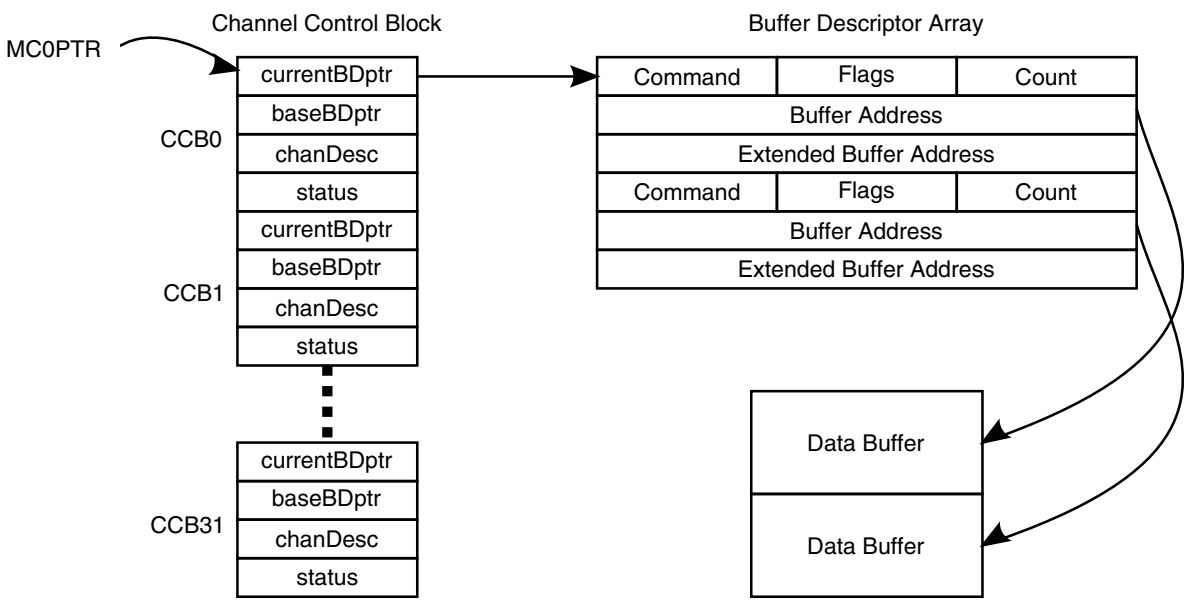


Figure 46-17. Data Structures Layout

The previous figure shows an example how these data structures are linked to pass command and pointers to data buffers. The SDMA's MC0PTR register holds the base address of the Channel 0 Control Block (CCB0). The Channel 0 control block holds a pointer to the array of buffer descriptors. The buffer descriptors are used to tell the channel 0 (boot channel) what to do as described Buffer Descriptor Format.

46.19.1.1 Buffer Descriptor Format

Buffer descriptors are three longs (32-bit words) in size as, shown in the following figure. A buffer descriptor describes the properties of the data buffer it points to. The buffer descriptors can be used for linear or circular data buffers in the ARM platform processor memory. The CCB contains a pointer to the base BD as well as the current BD.

Table 46-51. Buffer Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command								-	-	L	R	I	C	W	D	Count															
Buffer Address																															
Extended Buffer Address																															

Table 46-52. Buffer Descriptor Field Descriptions

Field	Description
31-24 Command	Command. The command field is used to differentiate operations performed within a script when the script accesses this particular buffer descriptor. The use of this field can be defined by the script. The command values defined for the bootload script are defined in Buffer Descriptor Commands for Bootload scripts. Refer to the individual script definition in script library documents in SDMA Scripts for command field definitions for other scripts.
23	Reserved
22	Reserved
21 L	Last Buffer Descriptor: This bit is set in SDMA IPC scripts to indicate to the receiving Core that the transfer has ended. Whenever the source finishes transferring the count it wanted to transfer, it sets LAST_BIT in the destination BD, to let the destination know that transfer is over.This bit also tells the destination software that when it processes the destination BDs, they need not process any BD after the BD with the LAST_BIT set.For example, when the DSP prepares a single buffer descriptor with count equals to 25 and ARM platform prepares a single buffer descriptor with count equals 100. When 25 bytes have been transferred from DSP to ARM platform, the DSP buffer descriptor is normally closed while the ARM platform buffer descriptor will have the L bit set and the byte count updated to 25.
20 R	erroR. Indicates an error occurred on the channel's buffer descriptor requested command. Some scripts may overwrite the command field with an error code indicating the source of the error. 0 No Error 1 Error
19 I	Interrupt. When SDMA has finished to process data transfer attached to this buffer descriptor, send an interrupt to the ARM platform. 0 No Interrupt 1 Interrupt the processor when BD is complete
18 C	CONTinuous. This buffer is allowed to receive multiple transmit buffers or is allowed to transmit to multiple receive buffers.The Continuos bit is decoded at the end of the processing of a BD to determine if the SDMA script must open a new BD to potentially continue the data transfer. 0 No further buffer descriptors 1 SDMA should move to the next Buffer descriptor after this one

Table continues on the next page...

Table 46-52. Buffer Descriptor Field Descriptions (continued)

Field	Description
17 W	Wrap. Indicates if this buffer descriptor is the last one for the channel control block. When encountering this bit set, the SDMA scripts updates the CurrentBD pointer to point to the first Buffer Descriptor of the array. This bit is set if the ARM platform wants to organize the array of BD in a circular way (like a ring). When all BD have been processed and if Wrap bit and CONtinuous bit are set in the last BD, the SDMA script will wrap around and it will try to re-open the first BD. 0 No Error 1 Wrap to first buffer descriptor after this one is processed.
16 D	D - "Done": bit 16: indicates the "ownership" of the buffer descriptor. When D=0 the host owns the buffer descriptor; when D=1 SDMA owns the buffer descriptor. In the case of the channel 0, D=1 indicates the SDMA has not yet processed this buffer, D=0 indicates the SDMA has processed this buffer. 0 ARM platform owns the buffer. 1 SDMA owns the buffer
15-0 Count	Count. the count field (bit 15-0) indicates the size of the data to be transmitted, the size of the data buffer pointed to by the buffer descriptor. The SDMA memory structure is different for program memory (16-bits shorts/ half-words) and data memory (32-bits long). For channel 0 buffer descriptors, Count is expressed in 16-bit half-words when PM is addressed and in 32-bit words when DM is addressed. Count is typically expressed in bytes for other channel scripts, but the unit is dependant on the script.
31-0	Buffer address. Address pointer to the data buffer.
31-0	Extended buffer address. Additional pointer or other information required by some scripts.

The buffer descriptors form an array of programmable size. If the last buffer descriptor is marked by the Wrap flag-bit W=1, the array of buffer descriptor is treated as a ring with some logically continuous portion owned by the ARM platform with D=0, and the remainder owned by the SDMA with D=1. The count field of the buffer descriptor indicates how much data has been transmitted.

If ARM platform has prepared 3 buffers to be filled by the SDMA script, it has also prepared 3 BD, one for each buffer. The *Cont* and *Wrap* bits are used to organize the buffers in a circular way. For example, *CONT*inous bit is set to 1 in the 2 first BDs and *Wrap* is set in the 3rd BD. The SDMA script opens and processes BD#1. Since *CONT*inous bit is set for this BD, the SDMA will open the second BD and it will process it. Each time a BD is processed, its *Done* bit is reset by the SDMA. After the 3rd BD, if *CONT*inous is not set but if *Wrap* is set, the SDMA script stops here and the next time the channel will be triggered, the script will open the BD pointed by the currentBDptr pointer of the CCB and it will correspond to the first buffer descriptor.

If the *CONT*inous bit and *Wrap* bits are both set in the 3rd BD, the script will close it and it will try to open the first BD. An error may occur at this point if the BD#1 has already been processed and its *Done* bit is 0. The SDMA script cannot process a BD with a *Done* bit to 0. It means the BD is not ready to be processed. To avoid this situation, the *CONT*inous bit should not be set for the last BD if *Wrap* is set, and the Interrupt flag must set for the last BD. It will warn the owner of the BD that all the BDs have been processed and it has to re-set to 1 the *Done* bit of all the BD's if it desires the SDMA to

fill them again. Basically, if the ARM platform expects the SDMA to fill up the buffers in a circular fashion, then it's the responsibility of the ARM platform to set the *Done* bit of a buffer descriptor at an appropriate time.

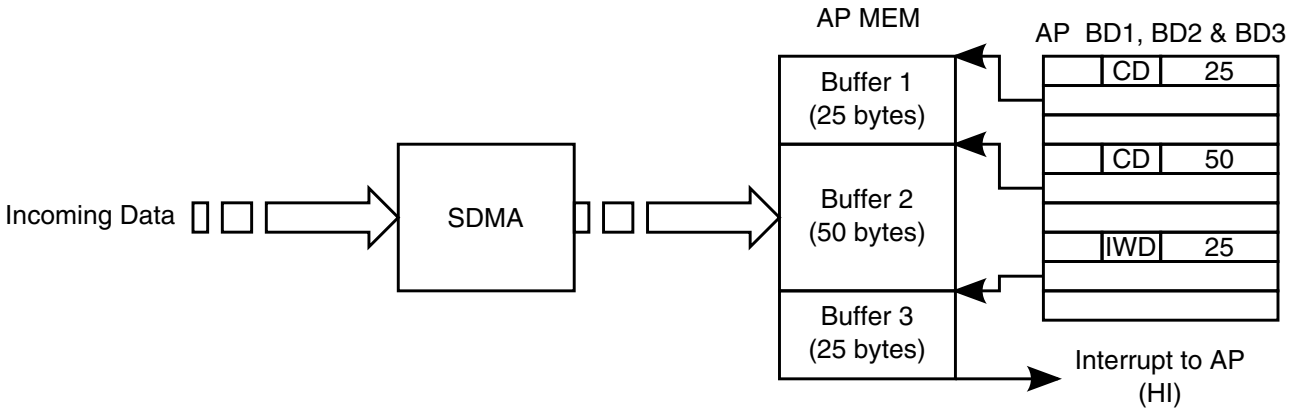


Figure 46-18. Buffer Descriptor Flow

The previous figure shows an example buffer descriptor flow. When the incoming data is stored and fills the first buffer of 25 bytes, the SDMA script opens the second BD because the CONTinuous bit was set. Then next incoming data is put in the second buffer. After receiving 50 bytes, the second buffer descriptor is also closed. The Done bit is reset and the third BD is opened. After receiving another 25 bytes, the third buffer is full and an interrupt is sent to the ARM platform because the Interrupt flag is set in the 3rd BD. The CONTinuous flag is not present the transfer is over. The next time the script will be triggered, the BD to be opened will be the first buffer descriptor since the Wrap flag was set in the 3rd BD. It is the ARM platform responsibility to set the Done bit of all the BD if it wants to use the same buffers.

46.19.1.2 Buffer Descriptor Commands for Bootload scripts

The command field of the buffer descriptor is defined separately for each script.

The following table lists the buffer descriptor commands defined for the channel 0 bootload script.

Table 46-53. Channel Zero Buffer Descriptor Commands

Command Field (binary)	Command	Description	Buffer Address	Extended Buffer Address
0000_0001 (0x01)	C0_SET_DM	Load SDMA data memory (RAM) from ARM platform memory buffer	ARM platform memory source address	SDMA memory destination address
0000_0010 (0x02)	C0_GET_DM	Copy SDMA data memory (RAM) to ARM platform memory buffer	ARM platform memory destination address	SDMA memory source address
0000_0100 (0x04)	C0_SET_PM	Load SDMA program memory (RAM) from ARM platform memory buffer	ARM platform memory source address	SDMA memory destination address
0000_0110 (0x06)	C0_GET_PM	Copy SDMA program memory (RAM) to ARM platform memory buffer	ARM platform memory destination address	SDMA memory source address
cccc_c111 (0x07 | CHN)	C0_SETCTX	Load Context for channel cccc into SDMA RAM from ARM platform memory buffer	ARM Platform memory source address	-
cccc_c011 (0x03 | CHN)	C0_GETCTXT	Copy Context for channel ccccc from SDMA RAM to ARM platform memory buffer	ARM platform memory destination address	-

The Channel 0 bootload commands are summarized as follows:

- C0_SET_[PM-DM]: load the buffer descriptor data in the SDMA local memory at the address pointed to by the "extended buffer address" field. The SDMA RAM can be seen as a Program Memory (PM, 16-bit address) or Data Memory (DM 32-bit address). When C0_SET_PM is used, the count field is expressed in "shorts" (16-bit half words), this command can be used to download scripts. When C0_SET_DM is used, the count field is expressed in "long" (32-bit words), this command can be used to download channel contexts to the context channel area in RAM.
- C0_GET_[PM-DM]: write to the buffer descriptor's data buffer the content of the SDMA local memory from the address pointed to by the "extended buffer address" field for the length defined by the count in the buffer descriptor. C0_GET_PM is used to dump some part of the Program Memory (may be used to dump context of a channel), therefore count is expressed in "shorts"; while C0_GET_DM is used to dump to the buffer descriptor's data buffer, so the count field is in "longs."
- C0_SETCTX: load a context into the SDMA context page area. The handling script decodes the channel number from the 5 MSB of the command field of the buffer descriptor. Using the channel number the script computes the offset of the context data pointer for the channel relative to the context page base to use as the destination

address in SDMA memory. Then the C0_SET_DM command explained above is invoked to load SDMA RAM from memory. The counter indicates the size in words of the context structure.

- Command value: (in binary) cccc c111, where ccccc is the channel number (5 bits). For instance, 0x0F means set context for channel 1, 0xFF means set context for channel 31.

- C0_GETCTX: write to the buffer descriptor's data buffer the content of the SDMA context page area. The handling script decodes the channel number from the 5 MSB of the command field of the buffer descriptor. Using this channel number, the script computes the offset of the context data pointer for the channel relative to the context page base to use as the source address for the copy. Then the C0_GET_DM command explained above is invoked to copy the context to memory. The counter indicates the size in words of the context structure.
- Command value: (in binary): cccc c011, where ccccc is the channel number (5 bits). For instance, 0x03 means get context of channel 1, 0xFB means get context of channel 31.

NOTE

To download channel context, C0_SETDM and C0_SETCTXT command can be used but the second one is easier because the channel number is embedded into the command field, whereas with the C0_SETDM, the pointer to the channel context area must be written into the extended buffer address field of the buffer descriptor.

46.19.1.3 Example of Buffer Descriptors for Channel 0.

Figure 46-20 illustrates the buffer descriptors that must be set in ARM platform memory space, before execution of boot code, to download contexts and scripts of channels 1, 4, and 10. After boot code execution, SDMA memory will be populated with the different contexts and scripts as presented in the following figure.

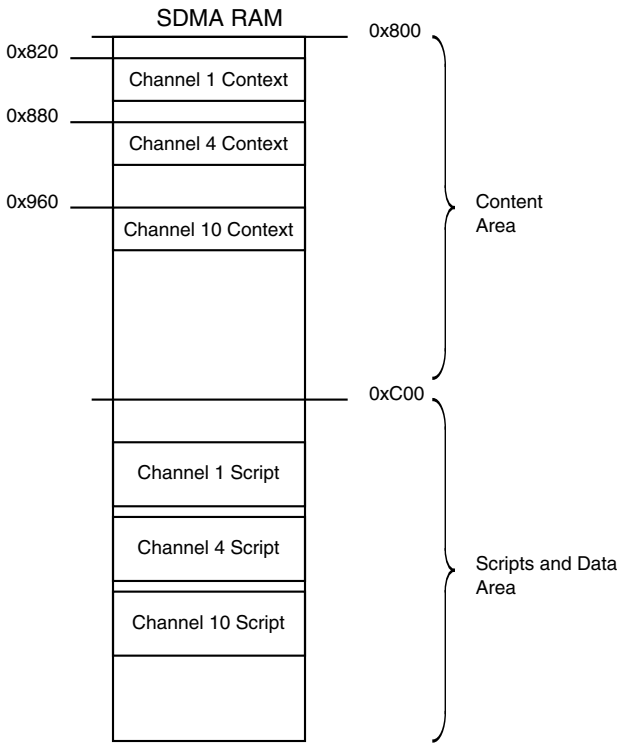
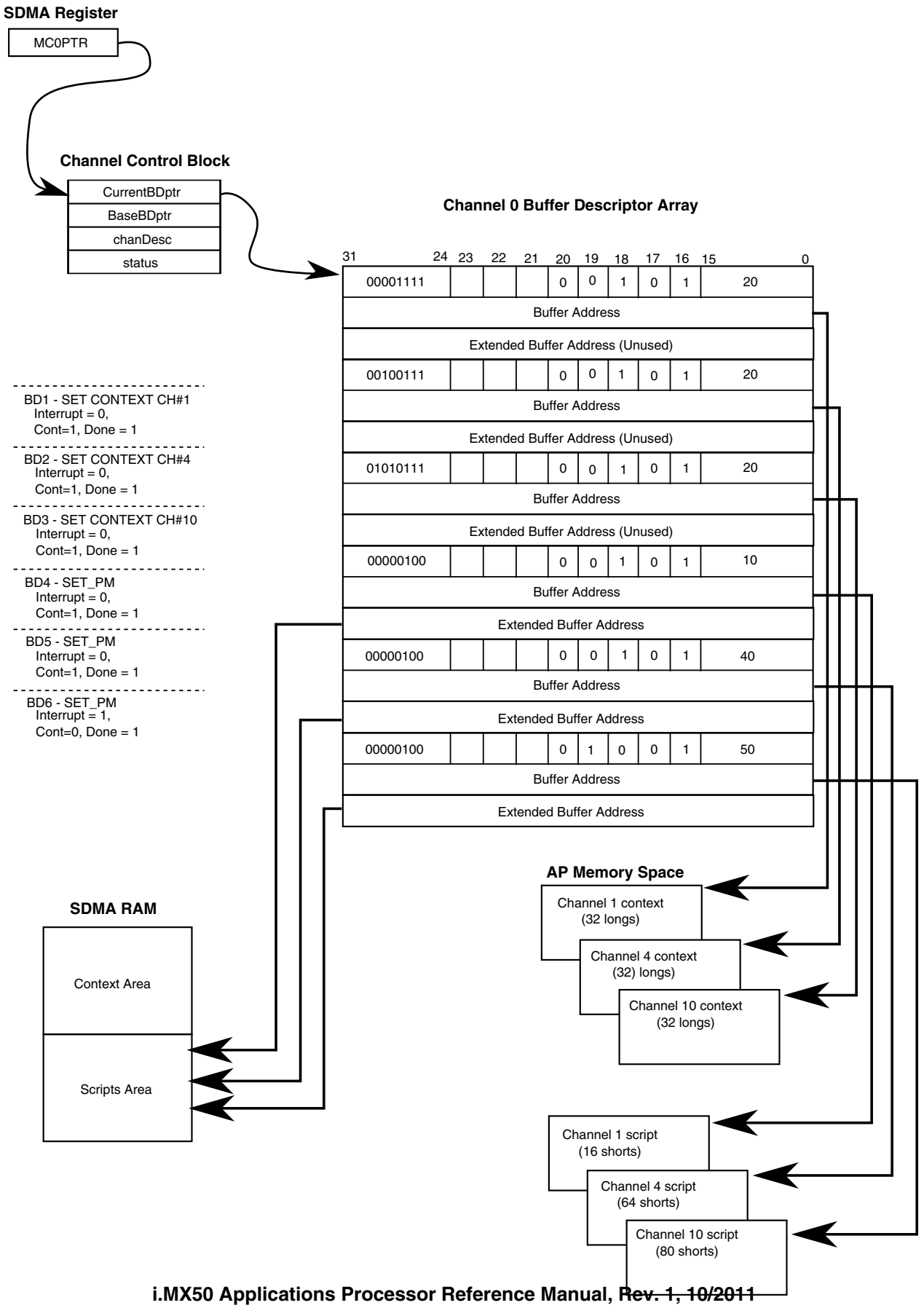


Figure 46-19. Example of SDMA RAM After Boot Session


SDMA Register
MC0PTR
Channel Control Block
CurrentBDptr
BaseBDptr
chanDesc
status
Channel 0 Buffer Descriptor Array
31 24 23 22 21 20 19 18 17 16 15 0
00001111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00100111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
01010111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00000100 0 0 1 0 1 10
Buffer Address
Extended Buffer Address
00000100 0 0 1 0 1 40
Buffer Address
Extended Buffer Address
00000100 0 1 0 0 1 50
Buffer Address
Extended Buffer Address
BD1 - SET CONTEXT CH#1
Interrupt = 0,
Cont=1, Done = 1
BD2 - SET CONTEXT CH#4
Interrupt = 0,
Cont=1, Done = 1
BD3 - SET CONTEXT CH#10
Interrupt = 0,
Cont=1, Done = 1
BD4 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD5 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD6 - SET_PM
Interrupt = 1,
Cont=0, Done = 1
SDMA RAM
Context Area
Scripts Area
AP Memory Space
Channel 1 context
(32 longs)
Channel 4 context
(32) longs)
Channel 10 context
(32 longs)
Channel 1 script
(16 shorts)
Channel 4 script
(64 shorts)
Channel 10 script
(80 shorts)

46.19.1.4 Channel Context

There are 32 channel context memory structures pointed to by the local save area pointer. These channel context memory structures are fixed.

The script in the SDMA computes the memory offset for a given channel based on the structure length and channel number. Figure below shows the structure of the channel context as it is saved in the SDMA local memory (RAM).

A channel context consists in 24 words, one per register. A total of 32 words are reserved for every channel. The additional 8 words are called scratch ram and they are dedicated to each channel. This memory area is commonly used for stack management.

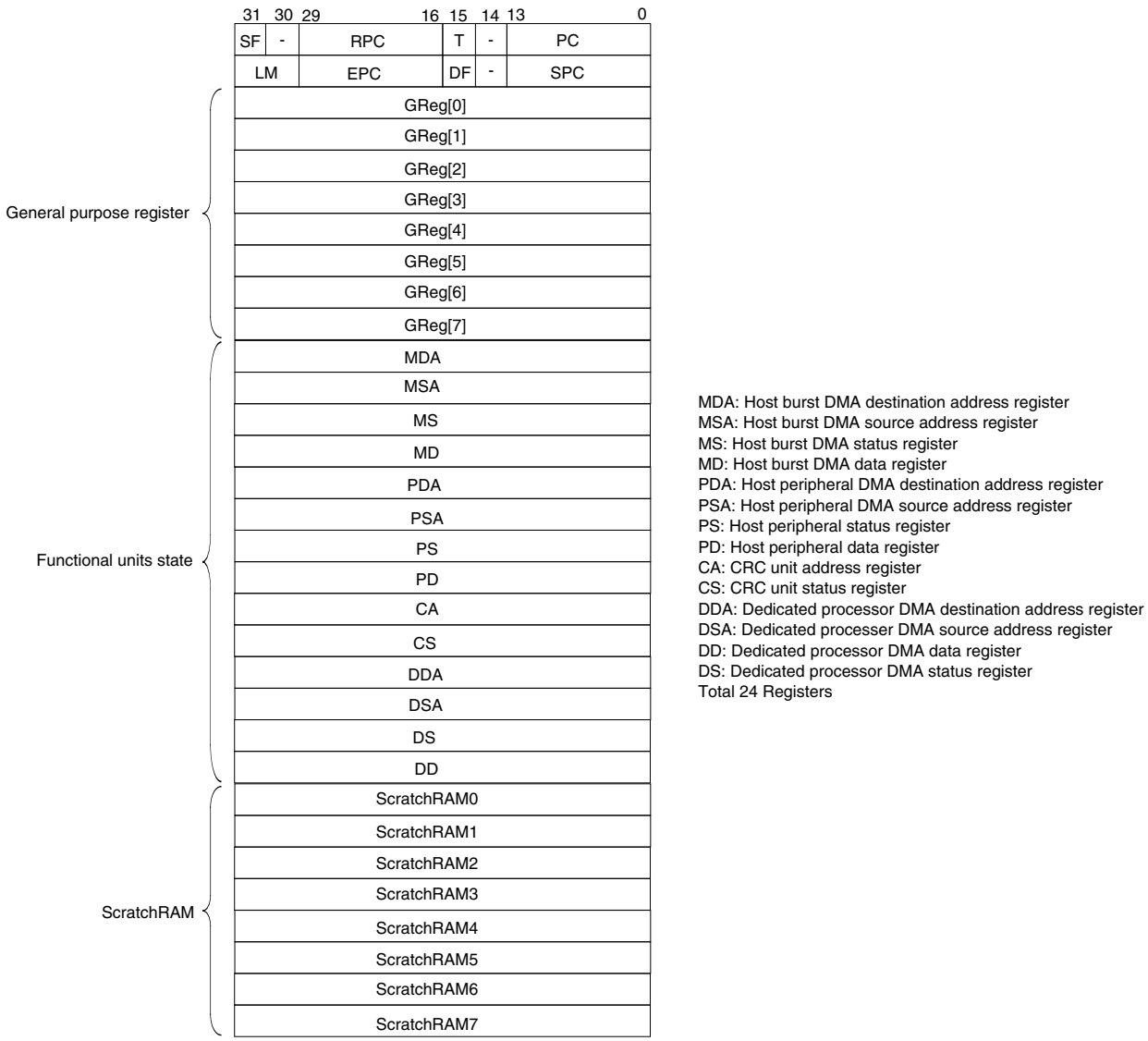


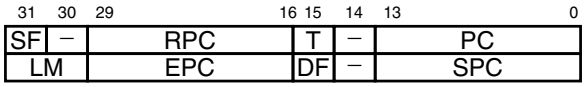
Figure 46-21. Channel Context Memory Structure

The structure is divided in 4 areas:

- Channel status registers
- General purpose registers
- Functional units state registers reflecting the state of the ARM platform DMAs (Burst and Peripheral DMA) and the CRC unit.
- Scratch RAM

The details of the channel context status registers are described in the following figure.

The PC field of the first long register must point to the SDMA RAM address where the script that will be executed on the channel is located and this value equals the one stored in the extended buffer address of the buffer descriptor with C0_SETPM command.



SF: Source fault while loading data
RPC: Return program counter
T: Test bit: status of arithmetic and test instructions
PC: Program counter
LM: Loop mode
EPC: Loop end program counter
DF: Destination fault while storing data
SPC: Loop Start program counter

Figure 46-22. SDMA State Registers (ShPC, ShLoop)

46.19.2 Typical Data Transfer Supported by SDMA DMA Units

This section presents a library of SDMA scripts that perform data transfers through the peripheral DMA and the burst DMA units.

The ARM platform memory and peripherals are devices that either the peripheral DMA or the burst DMA can access. The scripts are given for a peripheral DMA whose address registers are programmed in incremented mode when internal memory is involved. See the following table for the summary.

Table 46-54. Typical Data Transfers Summary

Data Transfer	Peripheral DMA	Burst DMA	Comments
ARM platform External Memory ↔ ARM platform External Memory		3	Copy mode Script example, see Burst DMA Unit Copy Mode and External Memory to External Memory.
ARM platform Peripheral ↔ ARM platform Peripheral	3		Copy mode if same data path width Script example, see Peripheral to Peripheral Transfer.
ARM platform External Memory ↔ ARM platform Peripheral	3	3	Data transit through SDMA Script example, see Transfer Between Peripheral and External Memory.
ARM platform External Memory ↔ ARM platform Internal Memory		3	Copy mode Script example, see Transfer Between External Memory and Internal Memory.

Table continues on the next page...

Table 46-54. Typical Data Transfers Summary (continued)

Data Transfer	Peripheral DMA	Burst DMA	Comments
ARM platform Internal Memory ↔ ARM platform Internal Memory		3	Copy mode Script example, see Internal Memory to Internal Memory.
ARM platform Internal memory ↔ ARM platform Peripheral	3		Data transit through SDMA Script example, see Transfer Between Peripheral and Internal Memory.

NOTE

These scripts are provided as examples of how to use DMA blocks to perform required data transfers: They are not "official" programs.

46.19.2.1 External Memory to External Memory

This section describes the SDMA script that performs data moves in external memory.

For this particular data transfer, only the burst DMA is used. It is programmed in copy mode, so no data transmits through an SDMA general register.

The SDMA core only monitors data transfer status. It is assumed source and destination address values are already present in two SDMA general registers (r1 and r2). For this example, it is also assumed that a 32-bit word-to-move for source-to-destination address is present in r0 and equals 64.

Data Moves in External Memory

```
1          stf r1,MSA                          // Source address setup

2          stf r2,MDA                          // Destination address setup

3          ldi r0,0x64                         // 64 words must be transferred from MSA to
MDA

4          ldi r1,0x8

MAIN_XFER:

5          cmphs r0,r1                         // Is r0 >= 0x8

6          bf LAST_XFER                      // If not, jump to last transfer label

7          stf r1,MD|CPY                        // Copy 8 words from MSA to MDA address.

8          subi r0,0x8                         // Decrement counter

9          jmp MAIN_XFER                     // return to main transfer loop

LAST_XFER:

10         stf r0,MD|CPY                        // perform last transfer
```

All instructions are performed in one cycle (jumps excepted). Instruction 7 triggers a copy transfer: A read burst access of 8-word starts, data is staged in MD and then a write burst of 8 words is executed. Instruction 8, 9, 5, and 6 are executed while the burst access is in progress. If this access is not complete when instruction 7 is executed a second time, SDMA stalls on this instruction as long as the previous copy transfer is not over. In this case, the instruction is no longer a one-cycle instruction.

During the main loop (MAIN_XFER), r1 always equals 8, so burst lengths are 8 words. On the last ldf |CPY instruction (10), r1 equals the reminder of r0 divided by 8; therefore, the length of bursts triggered in copy mode equal r1 value, which is between 1 and 7.

46.19.2.2 Peripheral to Peripheral Transfer

For this data transfer, only the peripheral DMA is used. It is programmed in copy mode, so no data will transmit through the SDMA general register used in the ldf instruction, but the contents of the general register are lost.

The SDMA core only monitors the transfer.

46.19.2.2.1 Source and Destination Target Have the Same Data Path Width

When the source and destination target have the same data path width, the following is true:

- Source target is a *half-word* (16-bit) peripheral located at address 0x1002.
- Destination is a *half-word* (16-bit) peripheral located at address 0x2006.

It is assumed the address values are already present in two SDMA general registers (r1, r2). The script for a transfer of 10 half-word is as follows:

Same Data Path Width for Source and Destination

```
//SETUP SECTION
1         stf r1, PSA|SZ16|F                    //r1=0x1002 Source address register setup
2         stf r2, PDA|SZ16|F                    //r2=0x2006 Destination address register
setup
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0xa                          //loop counter is 10
//MAIN LOOP TRANFER
copy_loop:

5         loop 2,0
6           ldf r7,PD|CPY                     //Reads 1 half-word from src and writes to
dest.
7           yield
8         bdf ERROR_DURING_XFER
ERROR_ADDR_SETUP:
                    //correction of PSA/PDA setup and jumps to main loop transfer
ERROR_DURING_XFER:
//flag error is set,
//PS can be read to know if error occurs during read or write access.
```

If a data transfer must occur between two word peripherals, only the setup section should be updated. The transfer itself is always performed by the hardware loop instruction.

All instructions are executed in one cycle (change of flow excepted). On instruction 6, a single read access is triggered, read data is staged in PD, and a write-to-destination is executed. When the transfers are in progress, the SDMA can execute he next instructions in parallel. If instruction 6, which performs the copy transfer, is executed while the previous access is not over, SDMA is stalled and instruction ldf is a multi-cycle instruction.

46.19.2.2.2 Source and Destination Target Have a Different Data Path Width

When the source and destination target have a different data path width, copy mode cannot be used, and any attempt to initiate a copy transfer immediately raises an error, which is stored in the SF flag.

The following example shows the SDMA code that could transfer 10 words from a *word* (32-bit) peripheral to a *half-word* peripheral whose addresses are preliminary and stored in r1 and r2.

Different Data Path Width for Source and Destination

```
//SETUP SECTION
1         stf r1, PSA|SZ32|F|PF                  //r1=0x1000 and prefetch data
2         stf r2, PDA|SZ16|F                    //r2=0x2006
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0xa                           //loop counter is 10
//MAIN LOOP TRANFER
main_loop_xfer_16_16:
5         loop 6,0
6           ldf r7,PD                         //copy 32-bit of PD in r7
7           stf r7,PD                          //store 16 LSB of r7 in PD and a flush is
executed
8           rorb r7
9           rorb r7                          //16 MSB --> 16 LSB
10          stf r7,PD                          //store 16 LSB of r6 in PD and a flush.
11          yield
```

On instruction 1, when the source address register is programmed and a data prefetch is required, a read access is executed. In parallel, the SDMA executes instructions 2 to 5. On instruction 6, the SDMA tries to read data that was fetched by instruction 1. If data is ready, the ldf will be a one cycle instruction; otherwise, the SDMA is stalled as long as the read access is not finished. Then, the 16 LSB of the read data is stored in PD and automatically flushed to the destination peripheral. In parallel, the SDMA executes the rotation instructions (8, 9), and stores the 16 MSB of the read data into PD. If a previous write access is finished, instruction 10 will be a one-cycle instruction.

The main loop transfer may appear inefficient, but due to wait states imposed to the peripheral DMA each time an external access is performed, a software pipeline is in place. During the time needed to flush PD, the SDMA executes the move and rotation operations. SDMA executes instructions in parallel with DMA accesses.

46.19.2.3 Transfer Between Peripheral and External Memory

46.19.2.3.1 Peripheral to External Memory Transfer

A transfer from a peripheral to the external memory controller involves the peripheral DMA and the burst DMA.

The code for transferring 100 word from word peripheral to the external memory would be as follows:

Peripheral to External Memory Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1         stf r1, PSA|SZ16|F|PF                  //r1=0x1000 and prefetch 32-bit data
2         stf r2, MDA                          //r2=0x2000, setup burst DMA destination
address
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0x64                           //loop counter is 100
5
          //MAIN LOOP TRANFER
6         loop 3,0
7           ldf r1,PD|PF                        // read 32 bits of PD and initiate a new read
access.
8           stf r1,MD|32                         // store 32 bits of r1 in the MD fifo.
9           yield
10        ldf r1,PD                            // last word data is read
11        stf r1,MD|32|FL                        // to flush all remaining bytes of MD
```

On instruction 1, the source address register of the peripheral DMA is programmed and data is fetched. This data is stored in PD and the SDMA reads PD during instruction 7, which is a one-cycle instruction that is read-access finished. On the same instruction (7), a data prefetch is required and a read access to the source peripheral is executed. In parallel, the SDMA stored the previous read data into the data register of MD. When MD (which is an eight-word FIFO) is full, a burst write access is executed to empty the FIFO. As long as the next SDMA instructions do not access the burst DMA, they will be one-cycle instructions. The following figures show how the peripheral DMA and burst DMA work in parallel.

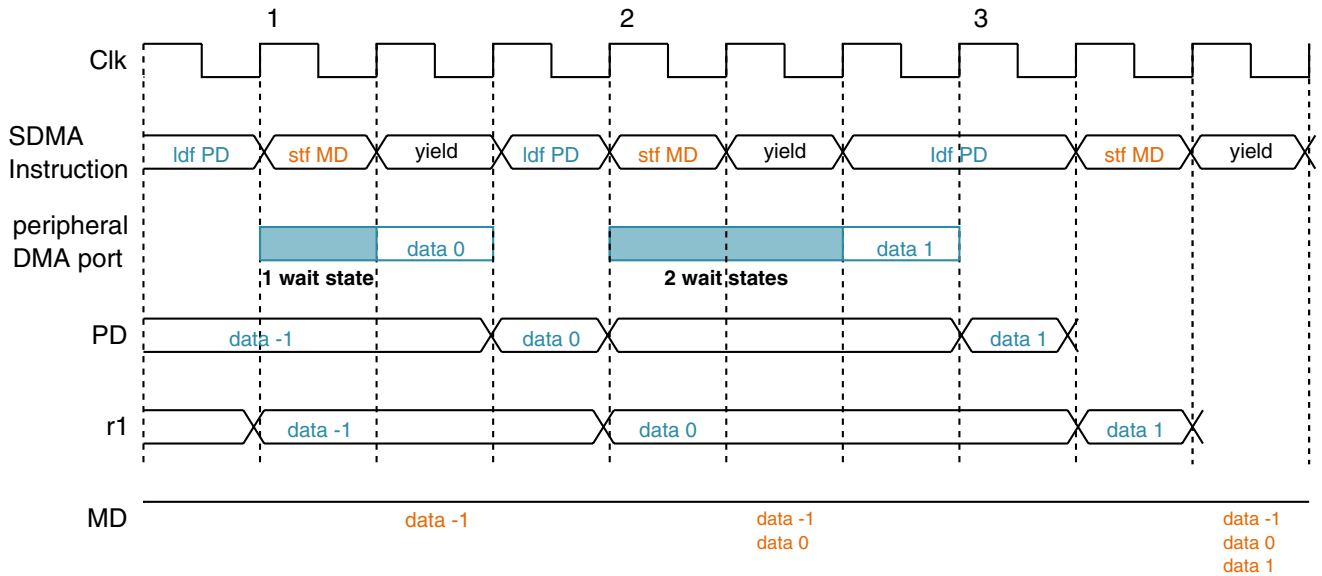


Figure 46-23. Peripheral to External Memory Example (1)

As seen in the figure above, the read access triggered by the ldf PD instruction is symbolized by the blue bar when in progress. After wait states, the read data (data 0, data 1) is stored in PD on the clk rising edge. On edge 2, data 0 is available in PD so it can be transferred to the SDMA general register r1, and then stored in MD FIFO. On edge 3, data 1 is not in PD; therefore, SDMA is stalled on the ldf instruction, which lasts two cycles. The figure below shows an example of when MD FIFO is full with data.

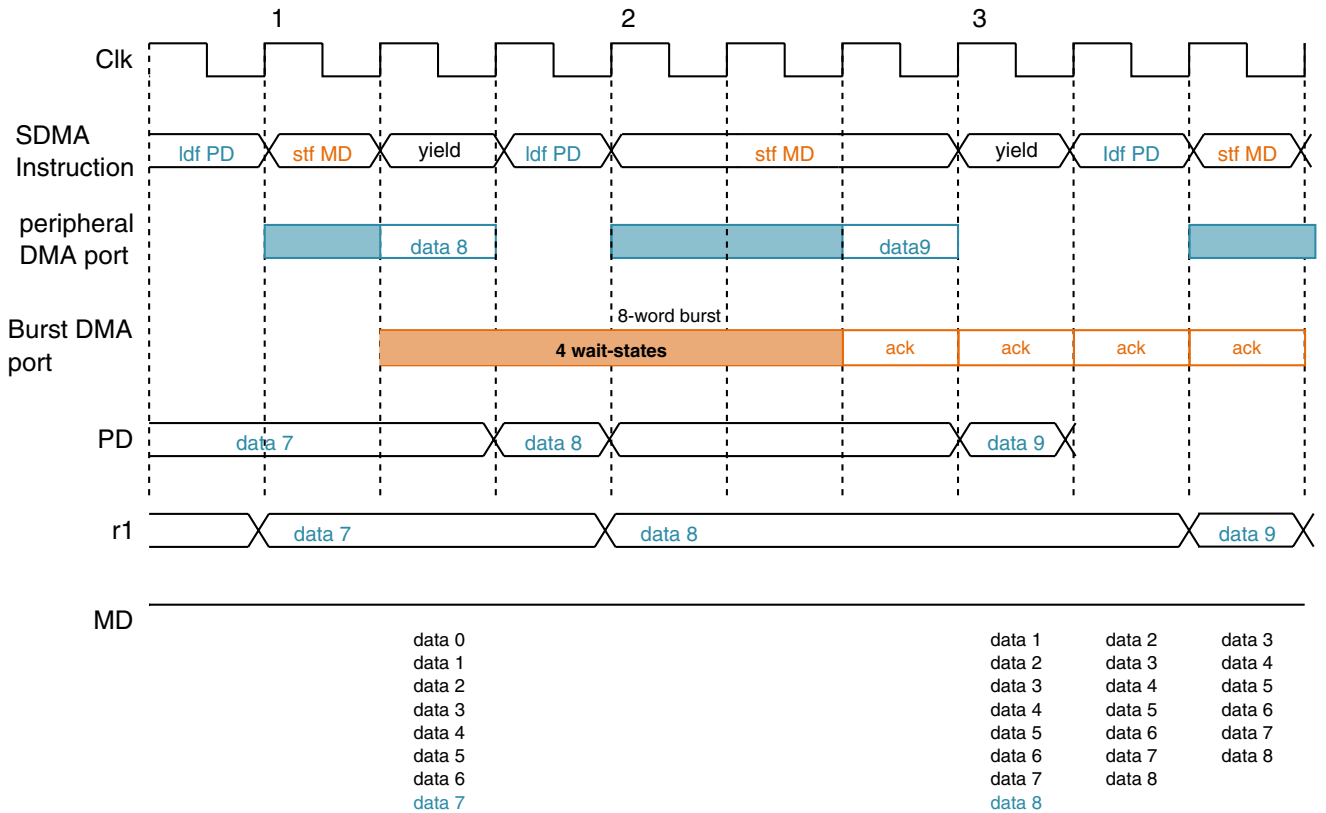


Figure 46-24. Peripheral to External Memory Example (2)

In the previous figure, the write bar means the burst DMA is performing a write burst access. The latency to have the first write acknowledge is four cycles. SDMA is stalled on instruction stf because no acknowledge was received, MD FIFO is full, and there is no empty slot to store data 9. When an acknowledge is sampled by the burst DMA, FIFO is shifted and data 8 is written. As long as there is at least one empty slot in MD FIFO, the stf MD instruction lasts one cycle.

46.19.2.3.2 External Memory to Peripheral Transfer

A transfer from the external memory to a peripheral involves the peripheral DMA and the burst DMA.

The code for transferring 100 word from external memory to a word peripheral would be as follows:

External Memory to Peripheral Transfer

```
//SETUP SECTION source and destination addresses are already in r1 and r2
1         stf r1, MSA|PF              //r1=0x1000 and starts a 8-word read burst
2         stf r2, PDA|SZ32|P           //r2=0x2010, setup peripheral DMA destination address
3         bdf ERROR_ADDR_SETUP
4         ldi r0,0x64                //loop counter is 100
//MAIN LOOP TRANFER
6         loop 3,0
```

```
7           ldf r1,MD|32|PF          // read 32 bits of MD and initiate a new read access
                                  // if MD is empty after this reading.
8           stf r1,PD              // store 32 bits of r1 in the PD.
9           yield
          10 ldf r1,MD|32           // last word data is read
11        stf r1,PD                // last write access
```

On instruction 1, a read burst of 8 words begins. Read data is staged into MD. On instruction 7 (and if data is available in MD), 32 bits are copied into r1. Then instruction 8 writes them into PD and an automatic flush is executed. The SDMA core, peripheral DMA, and burst DMA can work in parallel as long as no SDMA instruction tries to start a new write access on the peripheral DMA while the previous access is still in progress, or as long as there is data in MD when the SDMA tries to read it.

46.19.2.4 Transfer Between External Memory and Internal Memory

Since the internal memory (ARM platform RAM) is accessed via the peripheral DMA and the external memory is accessed via the burst DMA, the SDMA scripts that are described in Transfer Between Peripheral and External Memory can be reused. The exception is that the peripheral DMA address registers (PSA or PDA, depending on the script) should be programmed in incremented mode rather than frozen mode.

46.19.2.4.1 Internal Memory to Internal Memory

The internal memory can only be accessed via the peripheral DMA, so the script described in Peripheral to Peripheral Transfer can be reused with a different programming of the peripheral DMA address registers.

46.19.2.4.2 Transfer Between Peripheral and Internal Memory

For this transfer, the peripheral DMA is also used in copy mode.

The SDMA script is very similar to the one described in Peripheral to Peripheral Transfer, except for the peripheral DMA address registers programming.

46.20 ARM Platform Memory Map and Control Register Definitions

The ARM platform controls the SDMA by means of several interface registers. Those registers are described in the current section.

All registers are clocked with the SDMA clock (which means the ARM platform must ensure that the SDMA clock is running when it wants to access any register).

SDMAARM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_0000	ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)	32	R/W	0000_0000h	46.20.1/ 2734
63FB_0004	Channel Interrupts (SDMAARM_INTR)	32	w1c	0000_0000h	46.20.2/ 2735
63FB_0008	Channel Stop/Channel Status (SDMAARM_STOP_STAT)	32	w1c	0000_0000h	46.20.3/ 2735
63FB_000C	Channel Start (SDMAARM_HSTART)	32	R/W	0000_0000h	46.20.4/ 2735
63FB_0010	Channel Event Override (SDMAARM_EVTOVR)	32	R/W	0000_0000h	46.20.5/ 2736
63FB_0014	Channel BP Override (SDMAARM_DSPOVR)	32	R/W	FFFF_ FFFFh	46.20.6/ 2736
63FB_0018	Channel ARM platform Override (SDMAARM_HOSTOVR)	32	R/W	0000_0000h	46.20.7/ 2737
63FB_001C	Channel Event Pending (SDMAARM_EVTPEND)	32	w1c	0000_0000h	46.20.8/ 2737
63FB_0024	Reset Register (SDMAARM_RESET)	32	R	0000_0000h	46.20.9/ 2738
63FB_0028	DMA Request Error Register (SDMAARM_EVTERR)	32	R	0000_0000h	46.20.10/ 2738
63FB_002C	Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)	32	R/W	0000_0000h	46.20.11/ 2739
63FB_0030	Schedule Status (SDMAARM_PSW)	32	R	0000_0000h	46.20.12/ 2739
63FB_0034	DMA Request Error Register (SDMAARM_EVTERRDBG)	32	R	0000_0000h	46.20.13/ 2740
63FB_0038	Configuration Register (SDMAARM_CONFIG)	32	R/W	0000_0003h	46.20.14/ 2741

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_003C	SDMA LOCK (SDMAARM_SDMA_LOCK)	32	R/W	0000_0000h	46.20.15/ 2742
63FB_0040	OnCE Enable (SDMAARM_ONCE_ENB)	32	R/W	0000_0000h	46.20.16/ 2743
63FB_0044	OnCE Data Register (SDMAARM_ONCE_DATA)	32	R/W	0000_0000h	46.20.17/ 2743
63FB_0048	OnCE Instruction Register (SDMAARM_ONCE_INSTR)	32	R/W	0000_0000h	46.20.18/ 2744
63FB_004C	OnCE Status Register (SDMAARM_ONCE_STAT)	32	R	0000_E000h	46.20.19/ 2744
63FB_0050	OnCE Command Register (SDMAARM_ONCE_CMD)	32	R/W	0000_0000h	46.20.20/ 2746
63FB_0058	Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR)	32	R/W	0000_0001h	46.20.21/ 2746
63FB_005C	Channel 0 Boot Address (SDMAARM_CHN0ADDR)	32	R/W	0000_0050h	46.20.22/ 2747
63FB_0060	DMA Requests (SDMAARM_EVT_MIRROR)	32	R	0000_0000h	46.20.23/ 2748
63FB_0064	DMA Requests 2 (SDMAARM_EVT_MIRROR2)	32	R	0000_0000h	46.20.24/ 2748
63FB_0070	Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)	32	R/W	0000_0000h	46.20.25/ 2749
63FB_0074	Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)	32	R/W	0000_0000h	46.20.26/ 2750
63FB_0100	Channel Priority Registers (SDMAARM_SDMA_CHNPRI0)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0104	Channel Priority Registers (SDMAARM_SDMA_CHNPRI1)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0108	Channel Priority Registers (SDMAARM_SDMA_CHNPRI2)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_010C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI3)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0110	Channel Priority Registers (SDMAARM_SDMA_CHNPRI4)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0114	Channel Priority Registers (SDMAARM_SDMA_CHNPRI5)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0118	Channel Priority Registers (SDMAARM_SDMA_CHNPRI6)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_011C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI7)	32	R/W	0000_0000h	46.20.27/ 2751

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_0120	Channel Priority Registers (SDMAARM_SDMA_CHNPRI8)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0124	Channel Priority Registers (SDMAARM_SDMA_CHNPRI9)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0128	Channel Priority Registers (SDMAARM_SDMA_CHNPRI10)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_012C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI11)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0130	Channel Priority Registers (SDMAARM_SDMA_CHNPRI12)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0134	Channel Priority Registers (SDMAARM_SDMA_CHNPRI13)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0138	Channel Priority Registers (SDMAARM_SDMA_CHNPRI14)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_013C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI15)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0140	Channel Priority Registers (SDMAARM_SDMA_CHNPRI16)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0144	Channel Priority Registers (SDMAARM_SDMA_CHNPRI17)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0148	Channel Priority Registers (SDMAARM_SDMA_CHNPRI18)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_014C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI19)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0150	Channel Priority Registers (SDMAARM_SDMA_CHNPRI20)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0154	Channel Priority Registers (SDMAARM_SDMA_CHNPRI21)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0158	Channel Priority Registers (SDMAARM_SDMA_CHNPRI22)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_015C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI23)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0160	Channel Priority Registers (SDMAARM_SDMA_CHNPRI24)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0164	Channel Priority Registers (SDMAARM_SDMA_CHNPRI25)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0168	Channel Priority Registers (SDMAARM_SDMA_CHNPRI26)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_016C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI27)	32	R/W	0000_0000h	46.20.27/ 2751

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_0170	Channel Priority Registers (SDMAARM_SDMA_CHNPRI28)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0174	Channel Priority Registers (SDMAARM_SDMA_CHNPRI29)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0178	Channel Priority Registers (SDMAARM_SDMA_CHNPRI30)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_017C	Channel Priority Registers (SDMAARM_SDMA_CHNPRI31)	32	R/W	0000_0000h	46.20.27/ 2751
63FB_0200	Channel Enable RAM (SDMAARM_SDMA.CHNENBL0)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0204	Channel Enable RAM (SDMAARM_SDMA.CHNENBL1)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0208	Channel Enable RAM (SDMAARM_SDMA.CHNENBL2)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_020C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL3)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0210	Channel Enable RAM (SDMAARM_SDMA.CHNENBL4)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0214	Channel Enable RAM (SDMAARM_SDMA.CHNENBL5)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0218	Channel Enable RAM (SDMAARM_SDMA.CHNENBL6)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_021C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL7)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0220	Channel Enable RAM (SDMAARM_SDMA.CHNENBL8)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0224	Channel Enable RAM (SDMAARM_SDMA.CHNENBL9)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0228	Channel Enable RAM (SDMAARM_SDMA.CHNENBL10)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_022C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL11)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0230	Channel Enable RAM (SDMAARM_SDMA.CHNENBL12)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0234	Channel Enable RAM (SDMAARM_SDMA.CHNENBL13)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0238	Channel Enable RAM (SDMAARM_SDMA.CHNENBL14)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_023C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL15)	32	R/W	0000_0000h	46.20.28/ 2752

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_0240	Channel Enable RAM (SDMAARM_SDMA.CHNENBL16)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0244	Channel Enable RAM (SDMAARM_SDMA.CHNENBL17)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0248	Channel Enable RAM (SDMAARM_SDMA.CHNENBL18)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_024C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL19)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0250	Channel Enable RAM (SDMAARM_SDMA.CHNENBL20)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0254	Channel Enable RAM (SDMAARM_SDMA.CHNENBL21)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0258	Channel Enable RAM (SDMAARM_SDMA.CHNENBL22)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_025C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL23)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0260	Channel Enable RAM (SDMAARM_SDMA.CHNENBL24)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0264	Channel Enable RAM (SDMAARM_SDMA.CHNENBL25)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0268	Channel Enable RAM (SDMAARM_SDMA.CHNENBL26)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_026C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL27)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0270	Channel Enable RAM (SDMAARM_SDMA.CHNENBL28)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0274	Channel Enable RAM (SDMAARM_SDMA.CHNENBL29)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0278	Channel Enable RAM (SDMAARM_SDMA.CHNENBL30)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_027C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL31)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0280	Channel Enable RAM (SDMAARM_SDMA.CHNENBL32)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0284	Channel Enable RAM (SDMAARM_SDMA.CHNENBL33)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0288	Channel Enable RAM (SDMAARM_SDMA.CHNENBL34)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_028C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL35)	32	R/W	0000_0000h	46.20.28/ 2752

Table continues on the next page...

SDMAARM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_0290	Channel Enable RAM (SDMAARM_SDMA.CHNENBL36)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0294	Channel Enable RAM (SDMAARM_SDMA.CHNENBL37)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_0298	Channel Enable RAM (SDMAARM_SDMA.CHNENBL38)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_029C	Channel Enable RAM (SDMAARM_SDMA.CHNENBL39)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_02A0	Channel Enable RAM (SDMAARM_SDMA.CHNENBL40)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_02A4	Channel Enable RAM (SDMAARM_SDMA.CHNENBL41)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_02A8	Channel Enable RAM (SDMAARM_SDMA.CHNENBL42)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_02AC	Channel Enable RAM (SDMAARM_SDMA.CHNENBL43)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_02B0	Channel Enable RAM (SDMAARM_SDMA.CHNENBL44)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_02B4	Channel Enable RAM (SDMAARM_SDMA.CHNENBL45)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_02B8	Channel Enable RAM (SDMAARM_SDMA.CHNENBL46)	32	R/W	0000_0000h	46.20.28/ 2752
63FB_02BC	Channel Enable RAM (SDMAARM_SDMA.CHNENBL47)	32	R/W	0000_0000h	46.20.28/ 2752

46.20.1 ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)

Address: SDMAARM_MC0PTR is 63FB_0000h base + 0h offset = 63FB_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC0PTR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_MC0PTR field descriptions

Field	Description
31–0 MC0PTR	**Channel 0 Pointer** contains the 32-bit address, in ARM platform memory, of channel 0 control block (the boot channel). Appendix A fully describes the SDMA Application Programming Interface (API). The ARM platform has a read/write access and the SDMA has a read-only access.

46.20.2 Channel Interrupts (SDMAARM_INTR)

Address: SDMAARM_INTR is 63FB_0000h base + 4h offset = 63FB_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HI[31:0]																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_INTR field descriptions

Field	Description
31–0 HI[31:0]	The ARM platform Interrupts register contains the 32 HI[i] bits. If any bit is set, it will cause an interrupt to the ARM platform. This register is a "write-ones" register to the ARM platform. When the ARM platform sets a bit in this register the corresponding HI[i] bit is cleared. The interrupt service routine should clear individual channel bits when their interrupts are serviced, failure to do so will cause continuous interrupts. The SDMA is responsible for setting the HI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

46.20.3 Channel Stop/Channel Status (SDMAARM_STOP_STAT)

Address: SDMAARM_STOP_STAT is 63FB_0000h base + 8h offset = 63FB_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_STOP_STAT field descriptions

Field	Description
31–0 HE	This 32-bit register gives access to the ARM platform Enable bits. There is one bit for every channel. This register is a "write-ones" register to the ARM platform. When the ARM platform writes 1 in bit i of this register, it clears the HE[i] and HSTART[i] bits. Reading this register yields the current state of the HE[i] bits.

46.20.4 Channel Start (SDMAARM_HSTART)

Address: SDMAARM_HSTART is 63FB_0000h base + Ch offset = 63FB_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HSTART/HE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_HSTART field descriptions

Field	Description
31–0 HSTART/HE	The HSTART/HE registers are 32 bits wide with one bit for every channel. When a bit is written to 1, it enables the corresponding channel. Two physical registers are accessed with that address (HSTART and HE), which enables the ARM platform to trigger a channel a second time before the first trigger is processed. • This register is a "write-ones" register to the ARM platform. Neither HSTART[i] bit can be set while the corresponding HE[i] bit is cleared. • When the ARM platform tries to set the HSTART[i] bit by writing a one (if the corresponding HE[i] bit is clear), the bit in the HSTART[i] register will remain cleared and the HE[i] bit will be set. • If the corresponding HE[i] bit was already set, the HSTART[i] bit will be set. The next time the SDMA channel *i* attempts to clear the HE[i] bit by means of a `done` instruction, the bit in the HSTART[i] register will be cleared and the HE[i] bit will take the old value of the HSTART[i] bit. • Reading this register yields the current state of the HSTART[i] bits. This mechanism enables the ARM platform to pipeline two HSTART commands per channel.

46.20.5 Channel Event Override (SDMAARM_EVTOVR)

Address: SDMAARM_EVTOVR is 63FB_0000h base + 10h offset = 63FB_0010h

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

SDMAARM_EVTOVR field descriptions

Field	Description
31–0 EO	The Channel Event Override register contains the 32 EO[i] bits. A bit set in this register causes the SDMA to ignore DMA requests when scheduling the corresponding channel.

46.20.6 Channel BP Override (SDMAARM_DSPOVR)

Address: SDMAARM_DSPOVR is 63FB_0000h base + 14h offset = 63FB_0014h

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

SDMAARM_DSPOVR field descriptions

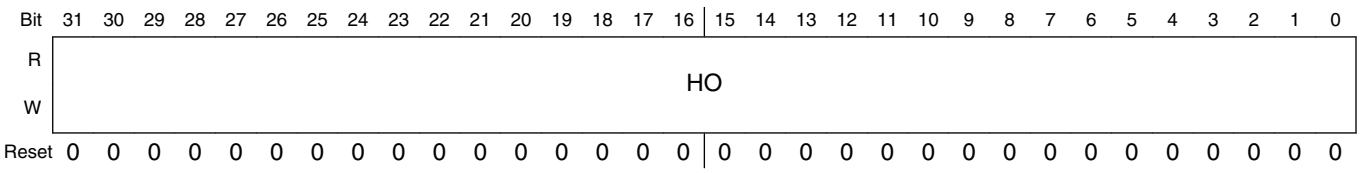
Field	Description
31–0 DO	This register is reserved. All DO bits should be set to the reset value of 1. A setting of 0 will prevent SDMA channels from starting according to the condition described in Runnable Channels Evaluation.

SDMAARM_DSPOVR field descriptions (continued)

Field	Description
	0 - Reserved 1 - Reset value.

46.20.7 Channel ARM platform Override (SDMAARM_HOSTOVR)

Address: SDMAARM_HOSTOVR is 63FB_0000h base + 18h offset = 63FB_0018h

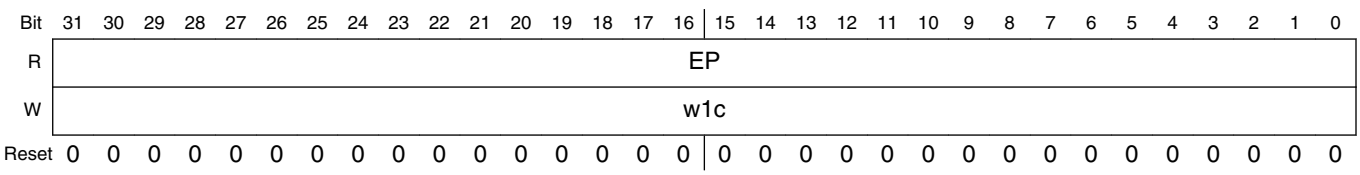


SDMAARM_HOSTOVR field descriptions

Field	Description
31–0 HO	The Channel ARM platform Override register contains the 32 HO[i] bits. A bit set in this register causes the SDMA to ignore the ARM platform enable bit (HE) when scheduling the corresponding channel.

46.20.8 Channel Event Pending (SDMAARM_EVTPEND)

Address: SDMAARM_EVTPEND is 63FB_0000h base + 1Ch offset = 63FB_001Ch

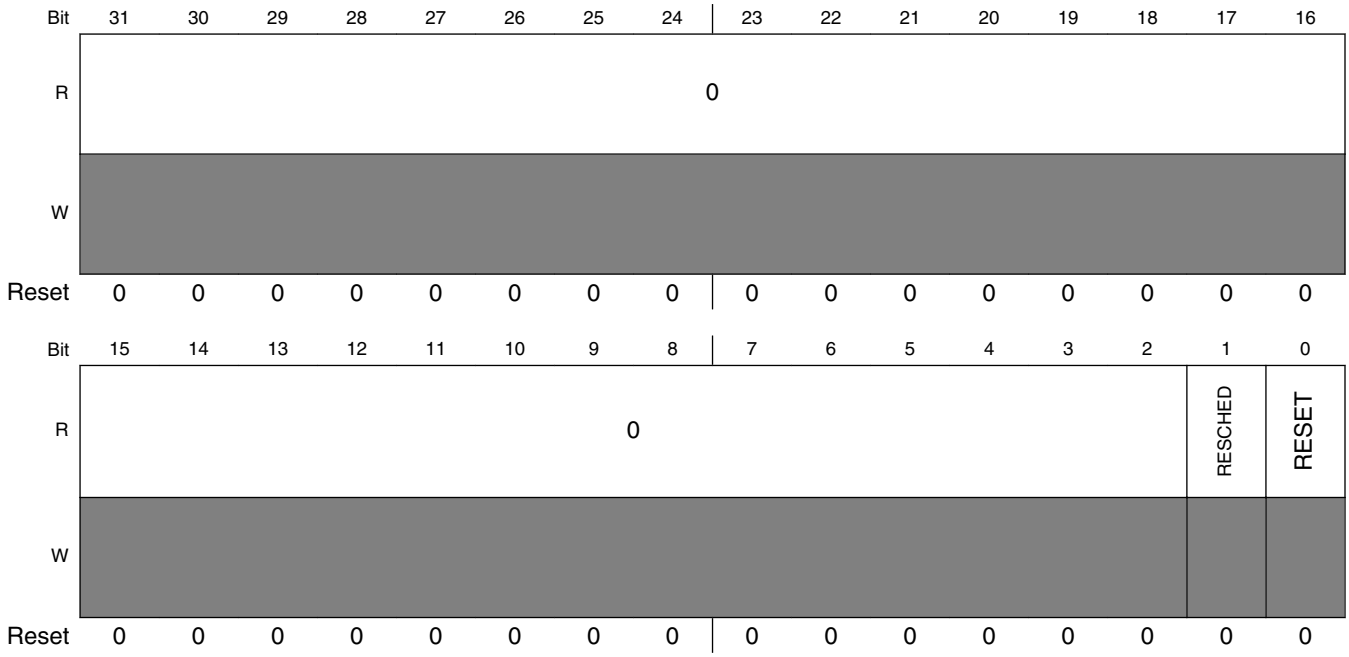


SDMAARM_EVTPEND field descriptions

Field	Description
31–0 EP	The Channel Event Pending register contains the 32 EP[i] bits. Reading this register enables the ARM platform to determine what channels are pending after the reception of a DMA request. • Setting a bit in this register causes the SDMA to reevaluate scheduling as if a DMA request mapped on this channel had occurred. This is useful for starting up channels, so that initialization is done before awaiting the first request. The scheduler can also set bits in the EVTPEND register according to the received DMA requests. • The EP[i] bit may be cleared by the `done` instruction when running the channel *i* script. This a "write-ones" mechanism: Writing a '0' does not clear the corresponding bit.

46.20.9 Reset Register (SDMAARM_RESET)

Address: SDMAARM_RESET is 63FB_0000h base + 24h offset = 63FB_0024h

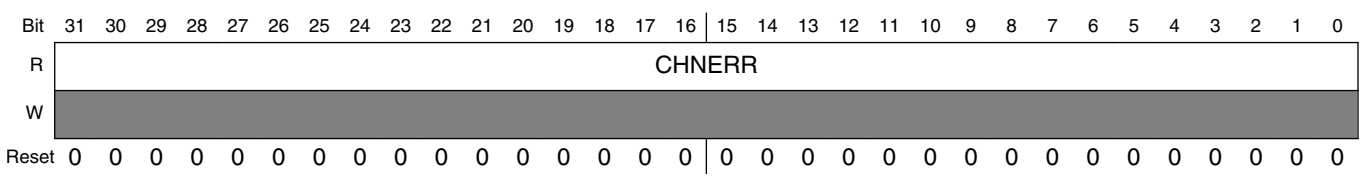


SDMAARM_RESET field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 RESCHED	When set, this bit forces the SDMA to reschedule as if a script had executed a `done` instruction. This enables the ARM platform to recover from a runaway script on a channel by clearing its HE[i] bit via the STOP register, and then forcing a reschedule via the RESCHED bit. The RESCHED bit is cleared when the context switch starts.
0 RESET	When set, this bit causes the SDMA to be held in a software reset. The internal reset signal is held low 16 cycles; the RESET bit is automatically cleared when the internal reset signal rises.

46.20.10 DMA Request Error Register (SDMAARM_EVTERR)

Address: SDMAARM_EVTERR is 63FB_0000h base + 28h offset = 63FB_0028h



SDMAARM_EVTERR field descriptions

Field	Description
31–0 CHNERR	This register is used by the SDMA to warn the ARM platform when an incoming DMA request was detected and it triggers a channel that is already pending or being serviced. This probably means there is an overflow of data for that channel. • An interrupt is sent to the ARM platform if the corresponding channel bit is set in the INTRMASK register. • This is a "write-ones" register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the ARM platform or during SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set; the EVTERR[i] bit is unaffected if the ARM platform tries to set the EP[i] bit, whereas, that EP[i] bit is already set.

46.20.11 Channel ARM platform Interrupt Mask (SDMAARM_INTRMASK)

Address: SDMAARM_INTRMASK is 63FB_0000h base + 2Ch offset = 63FB_002Ch

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

SDMAARM_INTRMASK field descriptions

Field	Description
31–0 HIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit HIMASK[i] is set, the HI[i] bit is set and an interrupt is sent to the ARM platform when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

46.20.12 Schedule Status (SDMAARM_PSW)

Address: SDMAARM_PSW is 63FB_0000h base + 30h offset = 63FB_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																NCP[2:0]			NCR[4:0]					CCP[2:0]				CCR[4:0]			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_PSW field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

SDMAARM_PSW field descriptions (continued)

Field	Description
15–13 NCP[2:0]	The Next Channel Priority gives the next pending channel priority. When the priority is 0, it means there is no pending channel and the NCR value has no meaning. 0 No running channel 1 Active channel priority
12–8 NCR[4:0]	The Next Channel Register indicates the number of the next scheduled pending channel with the highest priority.
7–4 CCP[2:0]	The Current Channel Priority indicates the priority of the current active channel. When the priority is 0, no channel is running: The SDMA is idle and the CCR value has no meaning. In the case that the SDMA has finished running the channel and has entered sleep state, CCP will indicate the priority of previous running channel. 0 No running channel 1 Active channel priority
3–0 CCR[4:0]	The Current Channel Register indicates the number of the channel that is being executed by the SDMA. SDMA. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel.

46.20.13 DMA Request Error Register (SDMAARM_EVTERRDBG)

Address: SDMAARM_EVTERRDBG is 63FB_0000h base + 34h offset = 63FB_0034h

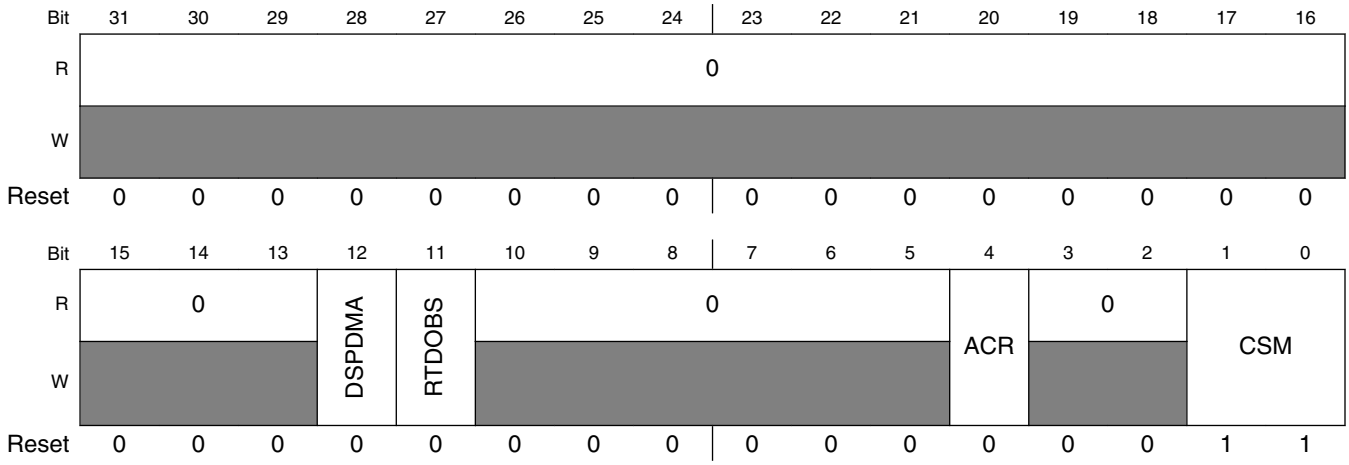
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHNERR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVTERRDBG field descriptions

Field	Description
31–0 CHNERR	This register is the same as EVTERR, except reading it does not clear its contents. This address is meant to be used in debug mode. The ARM platform OnCE may check this register value without modifying it.

46.20.14 Configuration Register (SDMAARM_CONFIG)

Address: SDMAARM_CONFIG is 63FB_0000h base + 38h offset = 63FB_0038h



SDMAARM_CONFIG field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value zero. Reserved
12 DSPDMA	This bit's function is reserved and should be configured as zero. 0 - Reset Value 1 - Reserved
11 RTDOBS	Indicates if Real-Time Debug pins are used: They do not toggle by default in order to reduce power consumption. 0 RTD pins disabled 1 RTD pins enabled
10–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4 ACR	ARM platform DMA / SDMA Core Clock Ratio. Selects the clock ratio between ARM platform DMA interfaces (burst DMA and peripheral DMA) and the internal SDMA core clock. The frequency selection is determined separately by the chip clock controller. This bit has to match the configuration of the chip clock controller that generates the clocks used in the SDMA. 0 ARM platform DMA interface frequency equals twice core frequency 1 ARM platform DMA interface frequency equals core frequency
3–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1–0 CSM	Selects the Context Switch Mode. The ARM platform has a read/write access. The SDMA cannot modify that register. The value at reset is 3, which selects the dynamic context switch by default. That register can be modified at anytime but the new context switch configuration will only be taken into account at the start of the next restore phase.

Table continues on the next page...

SDMAARM_CONFIG field descriptions (continued)

Field	Description
	NOTE: The first call to SDMA's channel 0 Bootload script after reset should use static context switch mode to ensure the context RAM for channel 0 is initialized in the channel SAVE Phase. After Channel 0 is run once, then any of the dynamic context modes can be used. 0 static 1 dynamic low power 2 dynamic with no loop 3 dynamic

46.20.15 SDMA LOCK (SDMAARM_SDMA_LOCK)

Address: SDMAARM_SDMA_LOCK is 63FB_0000h base + 3Ch offset = 63FB_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0														SRESET_ LOCK_CLR	LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_SDMA_LOCK field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1 SRESET_ LOCK_CLR	The SRESET_LOCK_CLR bit determine if the LOCK bit is cleared on a software reset triggered by writing to the RESET register. This bit cannot be changed if LOCK=1. SREST_LOCK_CLR is cleared by conditions that clear the LOCK bit. 0 Software Reset does not clear the LOCK bit. 1 Software Reset clears the LOCK bit.
0 LOCK	The LOCK bit is used to restrict access to update SDMA script memory through ROM channel zero scripts and through the OnCE interface under ARM platform control. The LOCK bit is set: • The SDMA_LOCK, ONCE_ENB,CH0ADDR, and ILLINSTADDR registers cannot be written. These registers can be read, but writes are ignored. • SDMA software executing out of ROM or RAM may check the LOCK bit in the LOCK register Lock Status Register (SDMACORE_SDMA_LOCK) to determine if certain operations are allowed, such as up-loading new scripts.

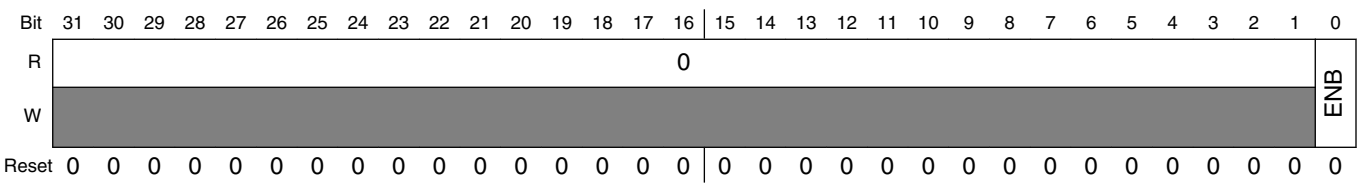
Table continues on the next page...

SDMAARM_SDMA_LOCK field descriptions (continued)

Field	Description
	Once the LOCK bit is set to 1, only a reset can clear it. The LOCK bit is cleared by a hardware reset. LOCK is cleared by a software reset only if SRESET_LOCK_CLR is set. 0 LOCK disengaged. 1 LOCK enabled.

46.20.16 OnCE Enable (SDMAARM_ONCE_ENB)

Address: SDMAARM_ONCE_ENB is 63FB_0000h base + 40h offset = 63FB_0040h

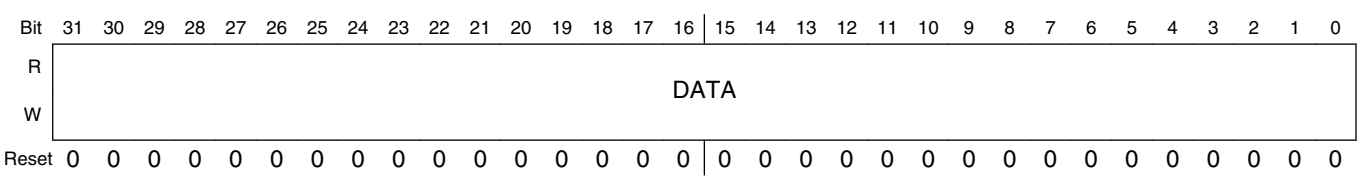


SDMAARM_ONCE_ENB field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 ENB	The OnCE Enable register selects the OnCE control source: When cleared (0), the OnCE registers are accessed through the JTAG interface; when set (1), the OnCE registers may be accessed by the ARM platform through the addresses described, as follows. • After reset, the OnCE registers are accessed through the JTAG interface. • Writing a 1 to ENB enables the ARM platform to access the ONCE_* as any other SDMA control register. • When cleared (0), all the ONCE_xxx registers cannot be written. The value of ENB cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

46.20.17 OnCE Data Register (SDMAARM_ONCE_DATA)

Address: SDMAARM_ONCE_DATA is 63FB_0000h base + 44h offset = 63FB_0044h

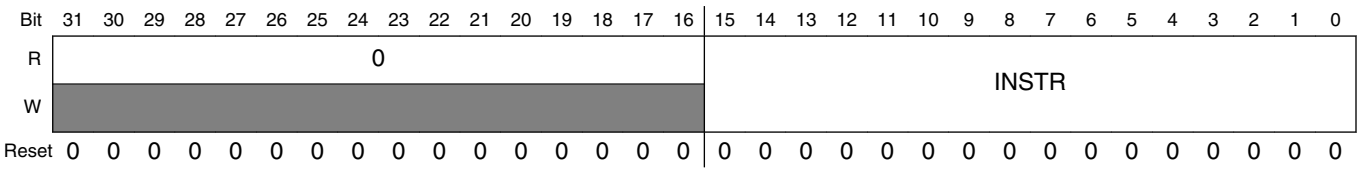


SDMAARM_ONCE_DATA field descriptions

Field	Description
31–0 DATA	Data register of the OnCE JTAG controller. Refer to OnCE and Real-Time Debug for information on this register.

46.20.18 OnCE Instruction Register (SDMAARM_ONCE_INSTR)

Address: SDMAARM_ONCE_INSTR is 63FB_0000h base + 48h offset = 63FB_0048h

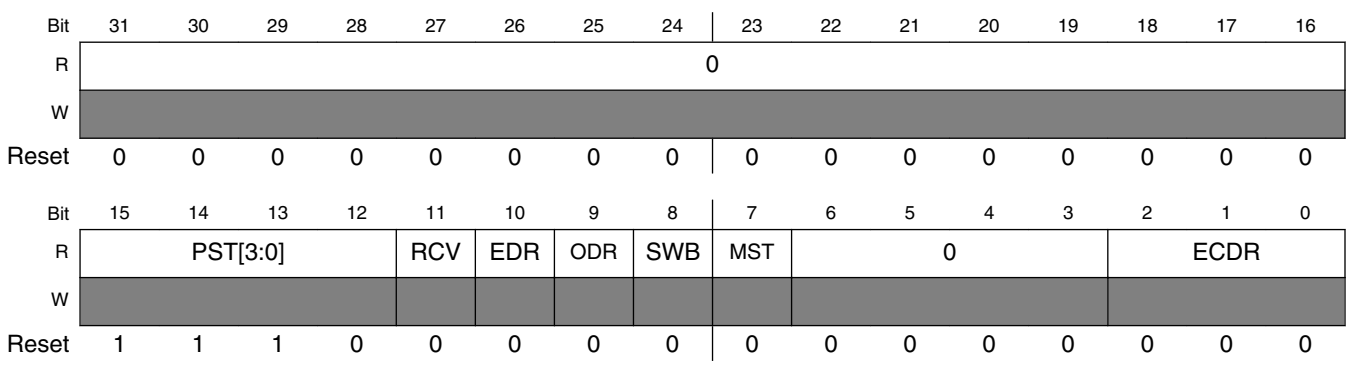


SDMAARM_ONCE_INSTR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 INSTR	Instruction register of the OnCE JTAG controller. Refer to OnCE and Real-Time Debug for information on this register.

46.20.19 OnCE Status Register (SDMAARM_ONCE_STAT)

Address: SDMAARM_ONCE_STAT is 63FB_0000h base + 4Ch offset = 63FB_004Ch



SDMAARM_ONCE_STAT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. Its states are as follows: • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode.

Table continues on the next page...

SDMAARM_ONCE_STAT field descriptions (continued)

Field	Description
	• The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In "Sleep" modes, no script is running (this is the RISC engine idle state). The "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel: They are used when entering debug mode after reset. The reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the ARM platform peripheral interface. 0 The JTAG interface controls the OnCE. 1 The ARM platform peripheral interface controls the OnCE.
6–3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2–0 ECDR	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one of the EDR bits is set (the meaning of the encoding is given below). The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addra_cond, addrb_cond, and data_cond conditions. The value of those fields is given by the EDR bits. 0 1 matched addra_cond

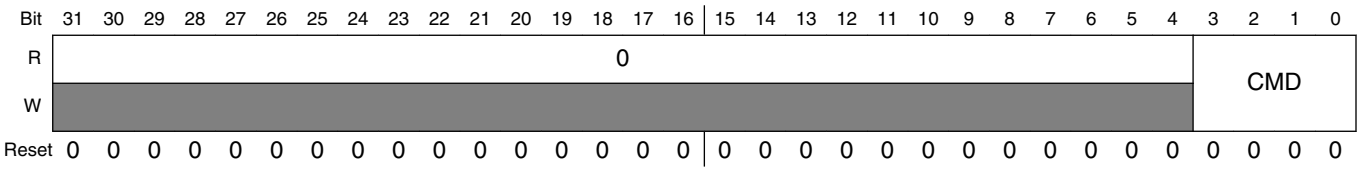
Table continues on the next page...

SDMAARM_ONCE_STAT field descriptions (continued)

Field	Description
	1 1 matched addrb_cond 2 1 matched data_cond

46.20.20 OnCE Command Register (SDMAARM_ONCE_CMD)

Address: SDMAARM_ONCE_CMD is 63FB_0000h base + 50h offset = 63FB_0050h

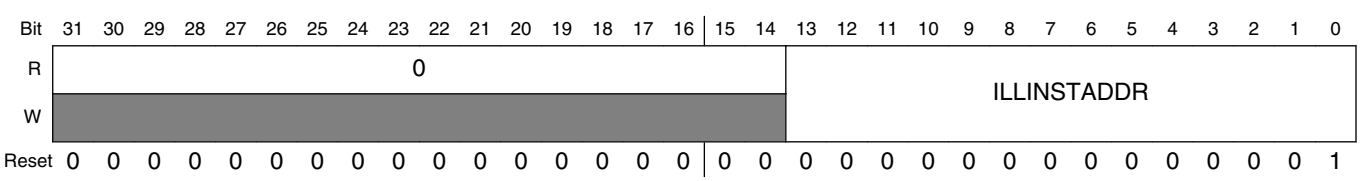


SDMAARM_ONCE_CMD field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value zero. Reserved
3–0 CMD	Writing to this register will cause the OnCE to execute the command that is written. When needed, the ONCE_DATA and ONCE_INSTR registers should be loaded with the correct value before writing the command to that register. For a list of the OnCE commands and their usage, see OnCE and Real-Time Debug. **NOTE:** 7-15 *reserved* 0 rstatus 1 dmov 2 exec_once 3 run_core 4 exec_core 5 debug_rqst 6 rbuffer

46.20.21 Illegal Instruction Trap Address (SDMAARM_ILLINSTADDR)

Address: SDMAARM_ILLINSTADDR is 63FB_0000h base + 58h offset = 63FB_0058h

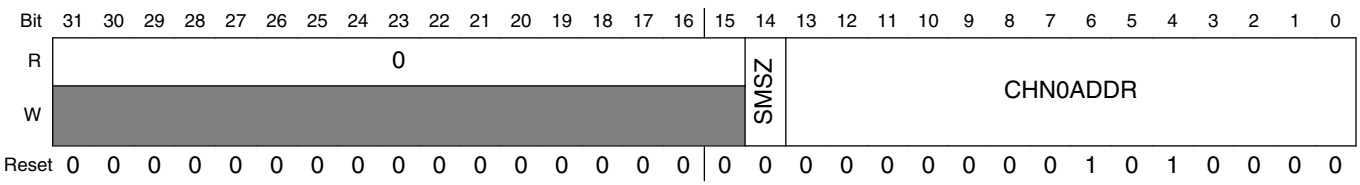


SDMAARM_ILLINSTADDR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value zero. Reserved
13–0 ILLINSTADDR	The Illegal Instruction Trap Address is the address where the SDMA jumps when an illegal instruction is executed. It is 0x0001 after reset. The value of ILLINSTADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

46.20.22 Channel 0 Boot Address (SDMAARM_CHN0ADDR)

Address: SDMAARM_CHN0ADDR is 63FB_0000h base + 5Ch offset = 63FB_005Ch



SDMAARM_CHN0ADDR field descriptions

Field	Description
31–15 Reserved	This read-only field is reserved and always has the value zero. Reserved
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. The value of SMSZ cannot be changed if the LOCK bit in the SDMA_LOCK register is set. 0 24 words per context 1 32 words per context
13–0 CHN0ADDR	This 14-bit register is used by the boot code of the SDMA. After reset, it points to the standard boot routine in ROM (channel 0 routine). By changing this address, you can perform a boot sequence with your own routine. The very first instructions of the boot code fetch the contents of this register (it is also mapped in the SDMA memory space) and jump to the given address. The reset value is 0x0050 (decimal 80). The value of CHN0ADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

46.20.23 DMA Requests (SDMAARM_EVT_MIRROR)

Address: SDMAARM_EVT_MIRROR is 63FB_0000h base + 60h offset = 63FB_0060h

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

SDMAARM_EVT_MIRROR field descriptions

Field	Description
31–0 EVENTS	This register reflects the DMA requests received by the SDMA for events 31-0. The ARM platform and the SDMA have a read-only access. There is one bit associated with each of 32 DMA request events. This information may be useful during debug of the blocks that generate the DMA requests. The EVT_MIRROR register is cleared following read access. 0 DMA request event not pending 1 DMA request event pending

46.20.24 DMA Requests 2 (SDMAARM_EVT_MIRROR2)

Address: SDMAARM_EVT_MIRROR2 is 63FB_0000h base + 64h offset = 63FB_0064h

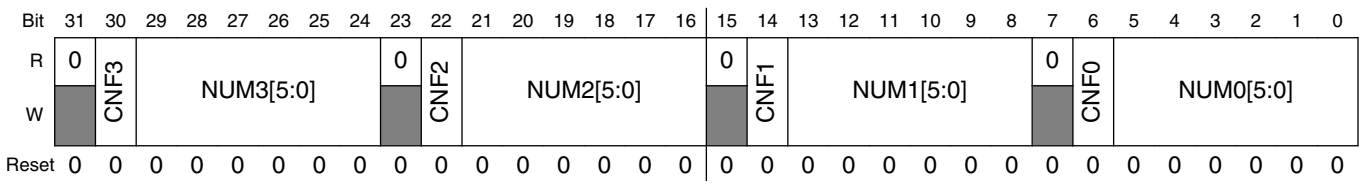
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EVENTS[47:32]															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMAARM_EVT_MIRROR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 EVENTS[47:32]	This register reflects the DMA requests received by the SDMA for events 47-32. The ARM platform and the SDMA have a read-only access. There is one bit associated with each of DMA request events. This information may be useful during debug of the blocks that generate the DMA requests. The EVT_MIRROR2 register is cleared following read access. 0 - DMA request event not pending 1- DMA request event pending

46.20.25 Cross-Trigger Events Configuration Register 1 (SDMAARM_XTRIG_CONF1)

Address: SDMAARM_XTRIG_CONF1 is 63FB_0000h base + 70h offset = 63FB_0070h



SDMAARM_XTRIG_CONF1 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value zero. Reserved
30 CNF3	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by the reception of a DMA request or by the starting of a channel script execution. 0 channel 1 DMA request
29–24 NUM3[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 CNF2	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM2[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15 Reserved	This read-only field is reserved and always has the value zero. Reserved
14 CNF1	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
13–8 NUM1[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7 Reserved	This read-only field is reserved and always has the value zero. Reserved

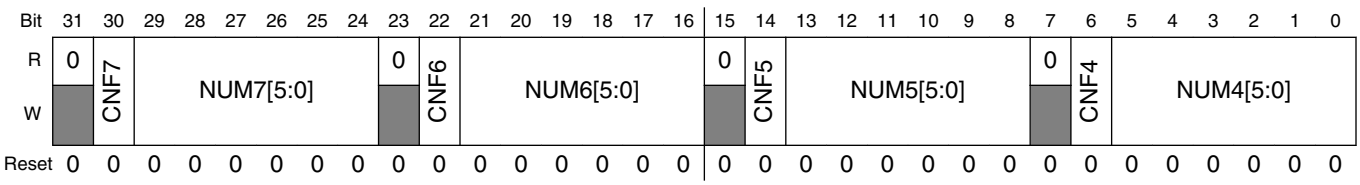
Table continues on the next page...

SDMAARM_XTRIG_CONF1 field descriptions (continued)

Field	Description
6 CNF0	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
5–0 NUM0[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

46.20.26 Cross-Trigger Events Configuration Register 2 (SDMAARM_XTRIG_CONF2)

Address: SDMAARM_XTRIG_CONF2 is 63FB_0000h base + 74h offset = 63FB_0074h



SDMAARM_XTRIG_CONF2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value zero. Reserved
30 CNF7	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
29–24 NUM7[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22 CNF6	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM6[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

SDMAARM_XTRIG_CONF2 field descriptions (continued)

Field	Description
14 CNF5	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution 0 channel 1 DMA request
13–8 NUM5[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7 Reserved	This read-only field is reserved and always has the value zero. Reserved
6 CNF4	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
5–0 NUM4[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

46.20.27 Channel Priority Registers (SDMAARM_SDMA_CHNPRI*n*)

Addresses: 63FB_0000h base + 100h offset + (4d × *n*), where *n* = 0d to 31d

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

SDMAARM_SDMA_CHNPRI*n* field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2–0 CHNPRIn	This contains the priority of channel number *n*. Useful values are between 1 and 7; 0 is reserved by the SDMA hardware to determine when there is no pending channel. Reset value is 0, which prevents the channels from starting.

46.20.28 Channel Enable RAM (SDMAARM_SDMA.CHNENBL*n*)

Addresses: 63FB_0000h base + 200h offset + (4d × *n*), where *n* = 0d to 47d

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

SDMAARM_SDMA.CHNENBL*n* field descriptions

Field	Description
31–0 ENBLn	This 32-bit value selects the channels that are triggered by the DMA request number *n*. If ENBLn[i] is set to 1, bit EP[i] will be set when the DMA request *n* is received. These 48 32-bit registers are physically located in a RAM, with no known reset value. It is thus essential for the ARM platform to program them before any DMA request is triggered to the SDMA, otherwise an unpredictable combination of channels may be started.

46.21 BP Memory Map and Control Register Definitions

The following section describes SDMA control registers available to the BP.

NOTE

These registers are physically implemented in all platforms, but are not accessible when the SDMA BP control port is not connected. Reset values are calculated to allow the system to work when those registers cannot be accessed.

All registers are clocked with the SDMA clock (which means the SDMA clock must be running when the BP wants to access any register).

SDMABP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_0000	Channel 0 Pointer (SDMABP_DC0PTR)	32	R/W	0000_0000h	46.21.1/ 2753
63FB_0004	Channel Interrupts (SDMABP_INTR)	32	w1c	0000_0000h	46.21.2/ 2753
63FB_0008	Channel Stop/Channel Status (SDMABP_STOP_STAT)	32	R/W	0000_0000h	46.21.3/ 2754

Table continues on the next page...

SDMABP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_000C	Channel Start (SDMABP_DSTART)	32	R	0000_0000h	46.21.4/ 2754
63FB_0028	DMA Request Error Register (SDMABP_EVTERR)	32	R	0000_0000h	46.21.5/ 2755
63FB_002C	Channel DSP Interrupt Mask (SDMABP_INTRMASK)	32	R/W	0000_0000h	46.21.6/ 2755
63FB_0034	DMA Request Error Register (SDMABP_EVTERRDBG)	32	R	0000_0000h	46.21.7/ 2756

46.21.1 Channel 0 Pointer (SDMABP_DC0PTR)

Address: SDMABP_DC0PTR is 63FB_0000h base + 0h offset = 63FB_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DC0PTR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_DC0PTR field descriptions

Field	Description
31–0 DC0PTR	**Channel 0 Pointer** contains the 32-bit address, in BP memory, of the array of channel control blocks starting with the one for channel 0 (the control channel). This register should be initialized by the BP before it enables a channel (for example, channel 0). See the API document i.MX50 SDMA Scripts User Manual for the use of this register. The BP has a read/write access and the SDMA has a read-only access.

46.21.2 Channel Interrupts (SDMABP_INTR)

Address: SDMABP_INTR is 63FB_0000h base + 4h offset = 63FB_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DI																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_INTR field descriptions

Field	Description
31–0 DI	The BP Interrupts register contains the 32 DI[i] bits. If any bit is set, it will cause an interrupt to the BP. • This register is a "write-ones" register to the BP. When the BP sets a bit in this register, the corresponding DI[i] bit is cleared.

SDMABP_INTR field descriptions (continued)

Field	Description
	• The interrupt service routine should clear individual channel bits when their interrupts are serviced; failure to do so will cause continuous interrupts. • The SDMA is responsible for setting the DI[i] bit corresponding to the current channel when the corresponding `done` instruction is executed.

46.21.3 Channel Stop/Channel Status (SDMABP_STOP_STAT)

Address: SDMABP_STOP_STAT is 63FB_0000h base + 8h offset = 63FB_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DE																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMABP_STOP_STAT field descriptions

Field	Description
31–0 DE	This 32-bit register gives access to the BP (DSP) Enable bits, DE. There is one bit for every channel. • This register is a "write-ones" register to the BP. • When the BP writes 1 in bit *i* of this register, it clears the DE[i] and DSTART[i] bits. • Reading this register yields the current state of the DE[i] bits.

46.21.4 Channel Start (SDMABP_DSTART)

Address: SDMABP_DSTART is 63FB_0000h base + Ch offset = 63FB_000Ch

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

SDMABP_DSTART field descriptions

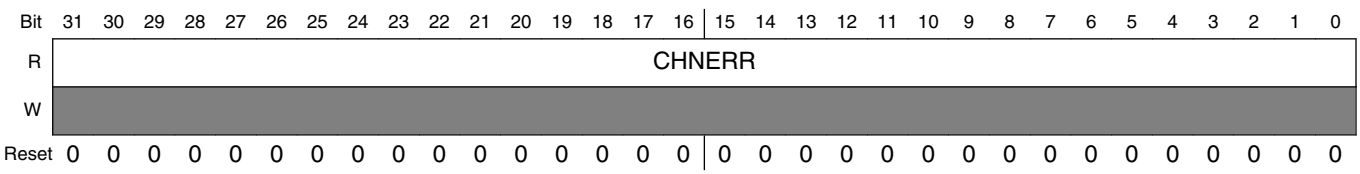
Field	Description
31–0 DSTART/DE	The DSTART/DE registers are 32 bits wide with one bit for every channel. • When a bit is written to 1, it enables the corresponding channel. • Two physical registers are accessed with that address (DSTART and DE), which enables the BP to trigger a channel a second time before the first trigger was processed. • This register is a "write-ones" register to the BP. Neither DSTART[i] bit can be set while the corresponding DE[i] bit is cleared. • When the BP tries to set the DSTART[i] bit by writing a one (if the corresponding DE[i] bit is clear), the bit in the DSTART[i] register will remain cleared and the DE[i] bit will be set. If the corresponding DE[i] bit was already set, the DSTART[i] bit will be set.

SDMABP_DSTART field descriptions (continued)

Field	Description
	• The next time the SDMA channel *i* attempts to clear the DE[i] bit by means of a `done` instruction, the bit in the DSTART[i] register will be cleared and the DE[i] bit will take the old value of the DSTART[i] bit. • Reading this register yields the current state of the DSTART[i] bits. This mechanism enables the BP to pipeline two DSTART commands per channel.

46.21.5 DMA Request Error Register (SDMABP_EVTERR)

Address: SDMABP_EVTERR is 63FB_0000h base + 28h offset = 63FB_0028h

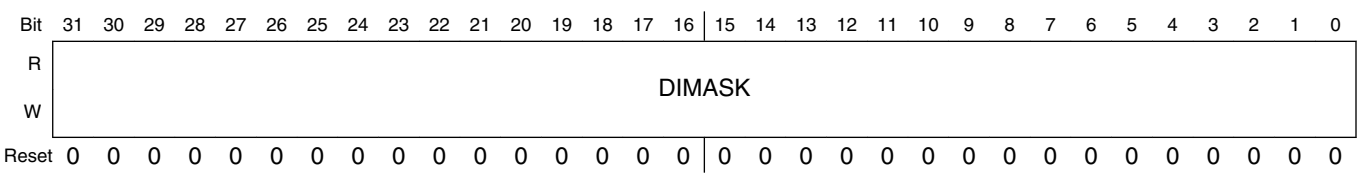


SDMABP_EVTERR field descriptions

Field	Description
31–0 CHNERR	This register is used by the SDMA to warn the BP when an incoming DMA request was detected; it then triggers a channel that is already pending or being serviced, which may mean there is an overflow of data for that channel. An interrupt is sent to the BP if the corresponding channel bit is set in the INTRMASK register. • This is a "write-ones" register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the BP or during an SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set. The EVTERR[i] bit is unaffected if the BP tries to set the EP[i] bit when that EP[i] bit is already set.

46.21.6 Channel DSP Interrupt Mask (SDMABP_INTRMASK)

Address: SDMABP_INTRMASK is 63FB_0000h base + 2Ch offset = 63FB_002Ch

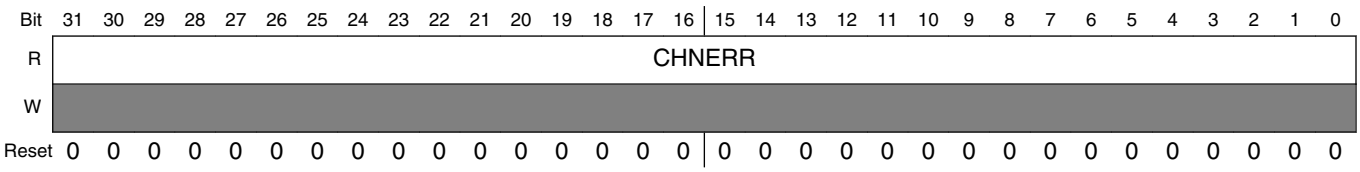


SDMABP_INTRMASK field descriptions

Field	Description
31–0 DIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit DIMASK[i] is set, the DI[i] bit is set and an interrupt is sent to the BP when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

46.21.7 DMA Request Error Register (SDMABP_EVTERRDBG)

Address: SDMABP_EVTERRDBG is 63FB_0000h base + 34h offset = 63FB_0034h



SDMABP_EVTERRDBG field descriptions

Field	Description
31–0 CHNERR	This register is the same as EVTERR except reading it does not clear its contents. This address is meant to be used in debug mode. The BP OnCE may check this register value without modifying it.

46.22 SDMA Internal (Core) Memory Map and Internal Register Definitions

The actual SDMA memory mapped registers are summarized in the following sections; for peripherals' memory maps, refer to the respective chapters.

The following definitions serve as a key for the SDMA internal register summary.

SDMACORE memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_0000	ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)	32	R	0000_0000h	46.22.1/ 2757
63FB_0002	Current Channel Pointer (SDMACORE_CCPTR)	32	R	0000_0000h	46.22.2/ 2758
63FB_0003	Current Channel Register (SDMACORE_CCR)	32	R	0000_0000h	46.22.3/ 2758
63FB_0004	Highest Pending Channel Register (SDMACORE_NCR)	32	R	0000_0000h	46.22.4/ 2759
63FB_0005	External DMA Requests Mirror (SDMACORE_EVENTS)	32	R	0000_0000h	46.22.5/ 2760
63FB_0006	Current Channel Priority (SDMACORE_CCPRI)	32	R	0000_0000h	46.22.6/ 2761
63FB_0007	Next Channel Priority (SDMACORE_NCPRI)	32	R	0000_0000h	46.22.7/ 2761

Table continues on the next page...

SDMACORE memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FB_0009	OnCE Event Cell Counter (SDMACORE_ECOUNT)	32	R/W	0000_0000h	46.22.8/ 2762
63FB_000A	OnCE Event Cell Control Register (SDMACORE_ECTL)	32	R/W	0000_0000h	46.22.9/ 2762
63FB_000B	OnCE Event Address Register A (SDMACORE_EAA)	32	R/W	0000_0000h	46.22.10/ 2764
63FB_000C	OnCE Event Cell Address Register B (SDMACORE_EAB)	32	R/W	0000_0000h	46.22.11/ 2764
63FB_000D	OnCE Event Cell Address Mask (SDMACORE_EAM)	32	R/W	0000_0000h	46.22.12/ 2765
63FB_000E	OnCE Event Cell Data Register (SDMACORE_ED)	32	R/W	0000_0000h	46.22.13/ 2765
63FB_000F	OnCE Event Cell Data Mask (SDMACORE_EDM)	32	R/W	0000_0000h	46.22.14/ 2765
63FB_0018	OnCE Real-Time Buffer (SDMACORE_RTB)	32	R/W	0000_0000h	46.22.15/ 2766
63FB_0019	OnCE Trace Buffer (SDMACORE_TB)	32	R	0000_0000h	46.22.16/ 2766
63FB_001A	OnCE Status (SDMACORE_OSTAT)	32	R	0000_0000h	46.22.17/ 2767
63FB_001C	Channel 0 Boot Address (SDMACORE_MCHN0ADDR)	32	R	0000_0000h	46.22.18/ 2769
63FB_001D	ENDIAN Status Register (SDMACORE_ENDIANNESS)	32	R	0000_0001h	46.22.19/ 2770
63FB_001E	Lock Status Register (SDMACORE_SDMA_LOCK)	32	R	0000_0000h	46.22.20/ 2771
63FB_001F	External DMA Requests Mirror #2 (SDMACORE_EVENTS2)	32	R	0000_0000h	46.22.21/ 2771

46.22.1 ARM platform Channel 0 Pointer (SDMACORE_MC0PTR)

Address: SDMACORE_MC0PTR is 63FB_0000h base + 0h offset = 63FB_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC0PTR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_MC0PTR field descriptions

Field	Description
31–0 MC0PTR	Contains the address-in the ARM platform memory space-of the initial SDMA context and scripts that are loaded by the SDMA boot script running on channel 0.

46.22.2 Current Channel Pointer (SDMACORE_CCPTR)

Address: SDMACORE_CCPTR is 63FB_0000h base + 2h offset = 63FB_0002h

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

SDMACORE_CCPTR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 CCPTR	Contains the start address of the context data for the current channel: Its value is *CONTEXT_BASE* + 24* *CCR* or *CONTEXT_BASE* + 32* *CCR* where CONTEXT_BASE = 0x0800. The value 24 or 32 is selected according to the programmed channel scratch RAM size in the register shown in Channel 0 Boot Address (SDMAARM_CHN0ADDR) .

46.22.3 Current Channel Register (SDMACORE_CCR)

Address: SDMACORE_CCR is 63FB_0000h base + 3h offset = 63FB_0003h

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

SDMACORE_CCR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4–0 CCR	Contains the number of the current running channel whose context is installed. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel. The PST bits in the OSTAT register indicate when the SDMA is in sleep state.

46.22.4 Highest Pending Channel Register (SDMACORE_NCR)

Address: SDMACORE_NCR is 63FB_0000h base + 4h offset = 63FB_0004h

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

SDMACORE_NCR field descriptions

Field	Description
31–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4–0 NCR	Contains the number of the pending channel that the scheduler has selected to run next.

46.22.5 External DMA Requests Mirror (SDMACORE_EVENTS)

This register is very useful in the case of DMA requests that are active when a peripheral FIFO level is above the programmed watermark. The activation of the DMA request (rising edge) is detected by the SDMA logic and it can enable one or several channels. One of the channels accesses the peripheral and reads or writes a number of data that matches the watermark level (for example, if the watermark is four words, the channel reads or writes four words). If the channel is effectively executed long after the DMA request was received, reading or writing the watermark number of data may not be sufficient to reset the DMA request (for example, if the FIFO watermark is four and at the channel execution it already contains nine pieces of data). This means no new rising edge may be detected by the SDMA, although there still remains transfers to perform. Therefore, if the channel were terminated at that time, it would not be restarted, causing potential overrun or underrun of the peripheral.The proposed mechanism is for the channel to check this register after it has performed the "watermark" number of accesses to the peripheral. If the bit for the DMA request that triggers this channel is set, it means there is still another watermark number of data to transfer. This goes on until the bit is cleared. The same script can be used for multiple channels that require this behavior. The script can determine its channel number from the CCR register and infer the corresponding DMA request bit to check. It needs a reference table that is coherent with the request-channel matrix that the ARM platform programmed.

Address: SDMACORE_EVENTS is 63FB_0000h base + 5h offset = 63FB_0005h

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

SDMACORE_EVENTS field descriptions

Field	Description
31–0 EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 0-31. The EVENTS2 register displays events 32-47.

46.22.6 Current Channel Priority (SDMACORE_CCPRI)

Address: SDMACORE_CCPRI is 63FB_0000h base + 6h offset = 63FB_0006h

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

SDMACORE_CCPRI field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2–0 CCPRI	Contains the 3-bit priority of the channel whose context is installed. It is 0 when no channel is running. **NOTE:** 1-7 current channel priority 0 no running channel

46.22.7 Next Channel Priority (SDMACORE_NCPRI)

Address: SDMACORE_NCPRI is 63FB_0000h base + 7h offset = 63FB_0007h

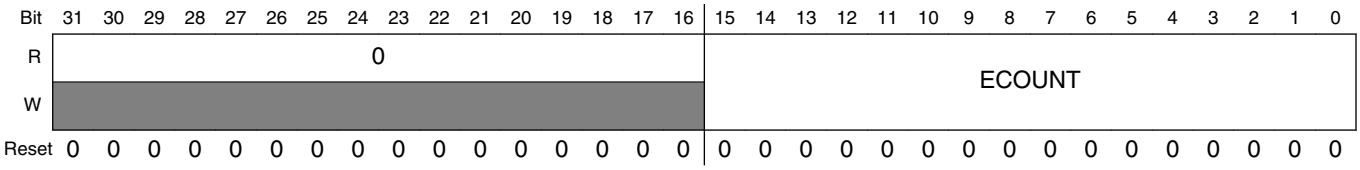
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													NCPRI		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_NCPRI field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2–0 NCPRI	Contains the 3-bit priority of the channel the scheduler has selected to run next. It is 0 when no other channel is pending.

46.22.8 OnCE Event Cell Counter (SDMACORE_ECOUNT)

Address: SDMACORE_ECOUNT is 63FB_0000h base + 9h offset = 63FB_0009h

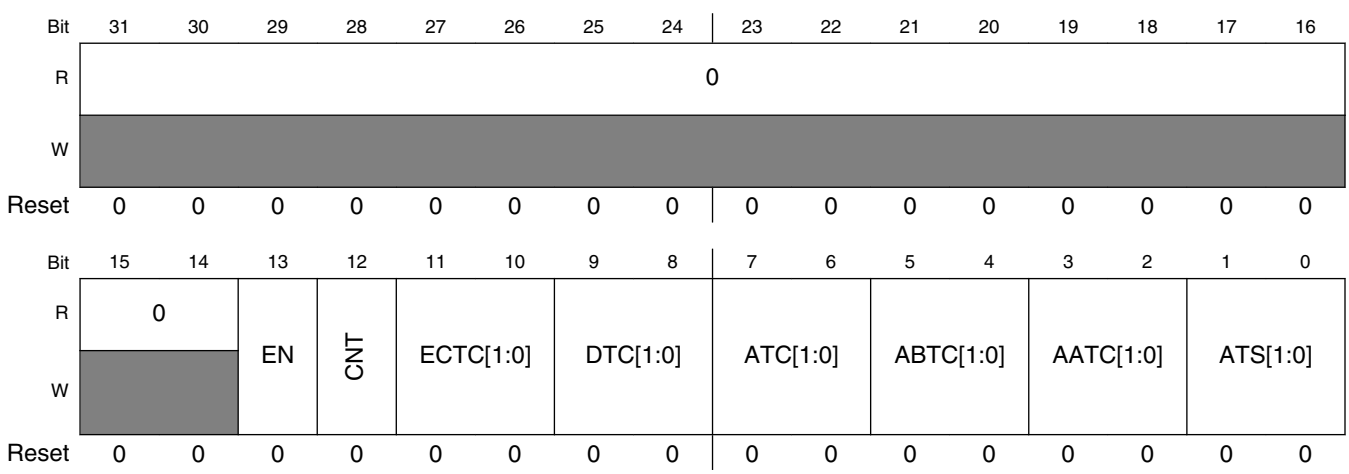


SDMACORE_ECOUNT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 ECOUNT	The event cell counter contains the number of times minus one that an event detection must occur before generating a debug request. • This register should be written before any attempt to use the event detection counter during an event detection process. • The counter is cleared on a JTAG reset.

46.22.9 OnCE Event Cell Control Register (SDMACORE_ECTL)

Address: SDMACORE_ECTL is 63FB_0000h base + Ah offset = 63FB_000Ah



SDMACORE_ECTL field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value zero. Reserved
13 EN	Event Cell Enable. If the EN bit is set, the event cell is allowed to generate debug requests (the cell is awakened). If it is cleared, the event detection unit is disabled and no hardware breakpoint is generated, but matching conditions are still reflected on the emulation pin.

Table continues on the next page...

SDMACORE_ECTL field descriptions (continued)

Field	Description
	0 Cell is disabled. 1 Cell is enabled.
12 CNT	Event Counter Enable. The event counter enable bit determines if the cell counter is used during the event detection. In order to use the event counter during an event detection process, the event cell counter register should be loaded with a value equal to the number of times minus one that an event occurs before a debug request is sent. After every event detection, the counter is decreased. When the counter reaches the value 0, the event detection cell sends a debug request to the core. The event counter register should be written and the EN bit should be set before each new event detection process uses the event counter. 0 Counter is disabled. 1 Counter is enabled.
11–10 ECTC[1:0]	The event cell trigger condition bits select the combination of address and data matching conditions that generate the final address/data condition. During program execution, if this event cell trigger condition goes to 1, a debug request is sent to the SDMA. The EN bit must be set to enable the debug request generation. 00 address ONLY 01 data ONLY 10 address AND data 11 address OR data
9–8 DTC[1:0]	The data trigger condition bits define when data is considered matching after comparison with the data register of the event detection unit. The operations are performed on unsigned values. 00 equal 01 not equal 10 greater than 11 less than
7–6 ATC[1:0]	The address trigger condition bits select how the two address conditions (addressA and addressB) are combined to define the global address matching condition. The supported combinations are described, as follows. 00 addressA ONLY 01 addrA AND addrB 10 addrA OR addrB 11 reserved
5–4 ABTC[1:0]	The Address B Trigger Condition (ABTC) controls the operations performed by address comparator B. All operations are performed on unsigned values. This comparator B outputs the addressB condition. 00 equal 01 not equal 10 greater than 11 less than
3–2 AATC[1:0]	The Address A Trigger Condition (AATC) controls the operations performed by address comparator A. All operations are performed on unsigned values. This comparator A outputs the addressA condition. 00 equal 01 not equal

Table continues on the next page...

SDMACORE_ECTL field descriptions (continued)

Field	Description
	10 greater than 11 less than
1–0 ATS[1:0]	The access type select bits define the memory access type required on the SDMA memory bus. 00 read ONLY 01 write ONLY 10 read or write 11 -

46.22.10 OnCE Event Address Register A (SDMACORE_EAA)

Address: SDMACORE_EAA is 63FB_0000h base + Bh offset = 63FB_000Bh

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

SDMACORE_EAA field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 EAA	Event Cell Address Register A computes an address A condition. It is cleared on a JTAG reset.

46.22.11 OnCE Event Cell Address Register B (SDMACORE_EAB)

Address: SDMACORE_EAB is 63FB_0000h base + Ch offset = 63FB_000Ch

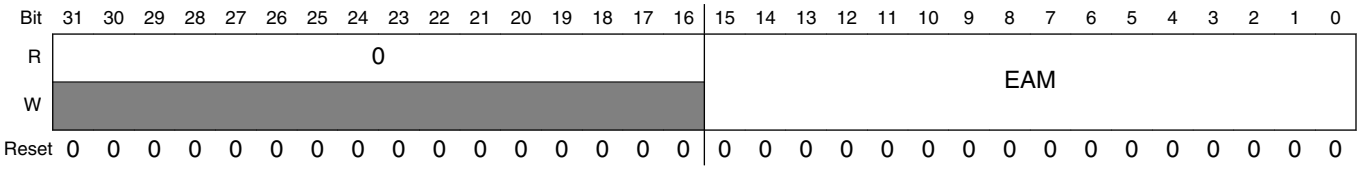
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																EAB															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_EAB field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 EAB	Event Cell Address Register B computes an address B condition. It is cleared on a JTAG reset.

46.22.12 OnCE Event Cell Address Mask (SDMACORE_EAM)

Address: SDMACORE_EAM is 63FB_0000h base + Dh offset = 63FB_000Dh

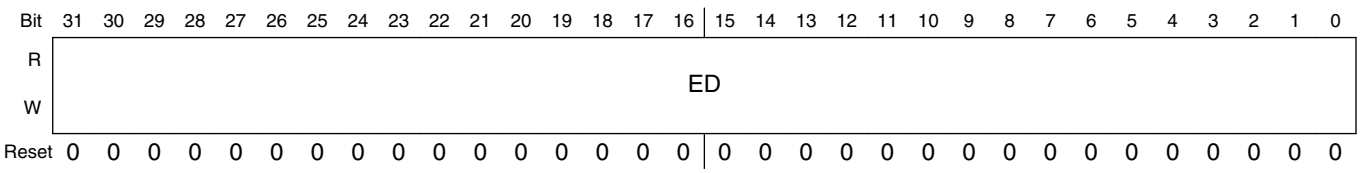


SDMACORE_EAM field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 EAM	The Event Cell Address Mask contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before performing the address comparison. **NOTE:** There is a common address mask value for both address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The register is cleared on a JTAG reset.

46.22.13 OnCE Event Cell Data Register (SDMACORE_ED)

Address: SDMACORE_ED is 63FB_0000h base + Eh offset = 63FB_000Eh

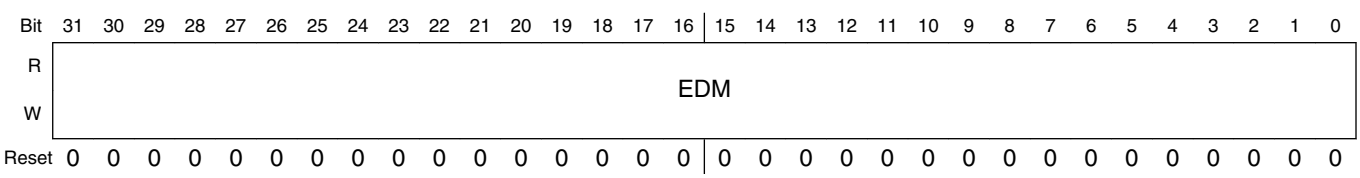


SDMACORE_ED field descriptions

Field	Description
31–0 ED	The event cell data register contains a user defined data value. This data value is an input for the data comparator which generates the data condition. It is cleared on a JTAG reset.

46.22.14 OnCE Event Cell Data Mask (SDMACORE_EDM)

Address: SDMACORE_EDM is 63FB_0000h base + Fh offset = 63FB_000Fh



SDMACORE_EDM field descriptions

Field	Description
31–0 EDM	The event cell data mask register contains the user-defined data mask value. • This mask is applied to the data value latched from the memory bus before performing the data comparison. • Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. • The data mask is cleared on a JTAG reset.

46.22.15 OnCE Real-Time Buffer (SDMACORE_RTB)

Address: SDMACORE_RTB is 63FB_0000h base + 18h offset = 63FB_0018h

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

SDMACORE_RTB field descriptions

Field	Description
31–0 RTB	The Real Time Buffer register stores and retrieves run time information without putting the SDMA in debug mode. Writing to that register triggers a pulse on a specific real-time debug pin whose connection depends on the chip implementation. The RTB value can be accessed by the OnCE under ARM platform or JTAG control using the rbuffer command.

46.22.16 OnCE Trace Buffer (SDMACORE_TB)

Address: SDMACORE_TB is 63FB_0000h base + 19h offset = 63FB_0019h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			TBF	TADDR[-2:2]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TADDR[1:0]		CHFADDR													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_TB field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value zero. Reserved
28 TBF	The Trace Buffer Flag is set when the buffer contains the addresses of a valid change of flow. The contents of the buffer should be ignored otherwise. 0 Invalid information 1 Valid information
27–14 TADDR	The target address is the address taken after the execution of the change of flow instruction.
13–0 CHFADDR	The change of flow address is the address where the change of flow is taken when executing a change of flow instruction.

46.22.17 OnCE Status (SDMACORE_OSTAT)

Address: SDMACORE_OSTAT is 63FB_0000h base + 1Ah offset = 63FB_001Ah

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0				ECDR[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDMACORE_OSTAT field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. • The "Program" state is the usual instruction execution cycle. • The "Data" state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The "Change of Flow" state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel-switching instructions). • The "Change of Flow in Loop" state is used when an error causes a hardware loop exit. • The "Debug" state means the SDMA is in debug mode. • The "Functional Unit" state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf).

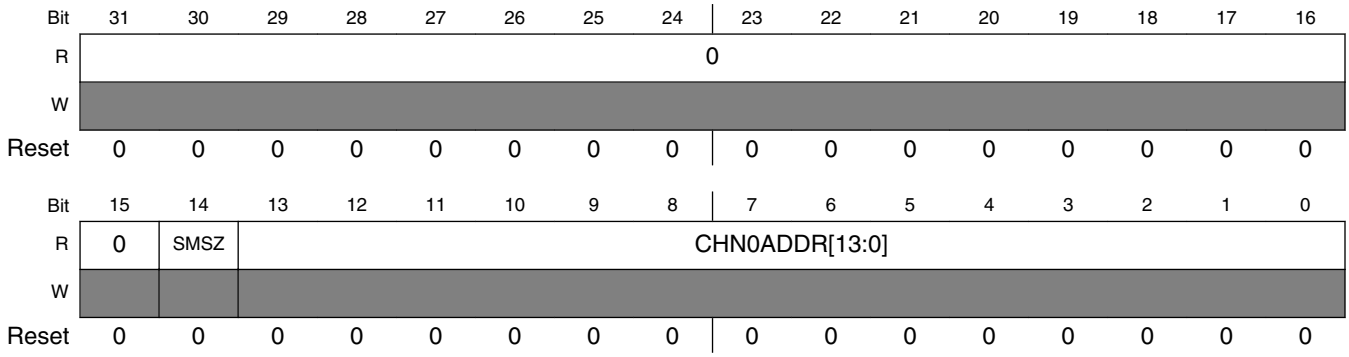
Table continues on the next page...

SDMACORE_OSTAT field descriptions (continued)

Field	Description
	• In "Sleep" modes, no script is running (this is the RISC engine idle state). The "after Reset" is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The "in Sleep" states are the same as above except they do not have any corresponding channel. They are used when entering debug mode after reset; the reason is that it is necessary to return to the "Sleep after Reset" state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow Loop Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the ARM platform peripheral interface. 0 JTAG interface controls the OnCE. 1 ARM platform peripheral interface controls the OnCE.
6–3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2–0 ECDR[2:0]	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addressA, addressB, and data conditions; the value of those fields is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one EDR bit is set; the meaning of the encoding is as follows: 0 1 matched addressA condition 1 1 matched addressB condition 2 1 matched data condition

46.22.18 Channel 0 Boot Address (SDMACORE_MCHN0ADDR)

Address: SDMACORE_MCHN0ADDR is 63FB_0000h base + 1Ch offset = 63FB_001Ch

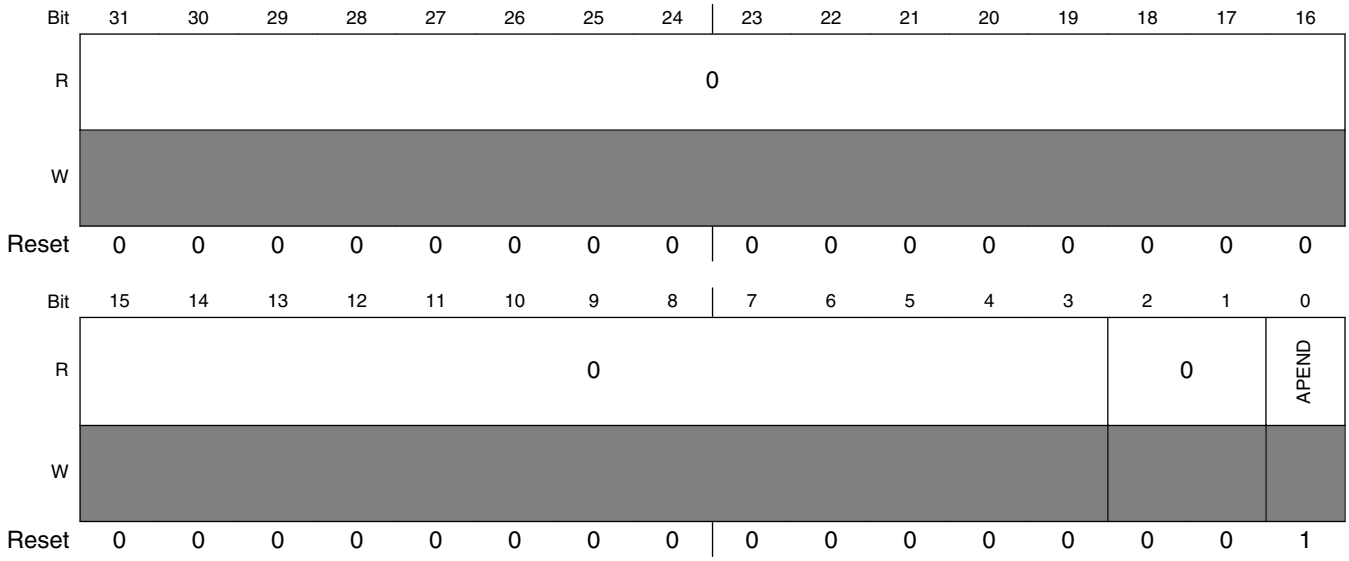


SDMACORE_MCHN0ADDR field descriptions

Field	Description
31–15 Reserved	This read-only field is reserved and always has the value zero. Reserved
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. 0 24 words per context 1 32 words per context
13–0 CHN0ADDR[13:0]	Contains the address of the channel 0 routine programmed by the ARM platform; it is loaded into a general register at the very start of the boot and the SDMA jumps to the address it contains. By default, it points to the standard boot routine in ROM.

46.22.19 ENDIAN Status Register (SDMACORE_ENDIANNESS)

Address: SDMACORE_ENDIANNESS is 63FB_0000h base + 1Dh offset = 63FB_001Dh

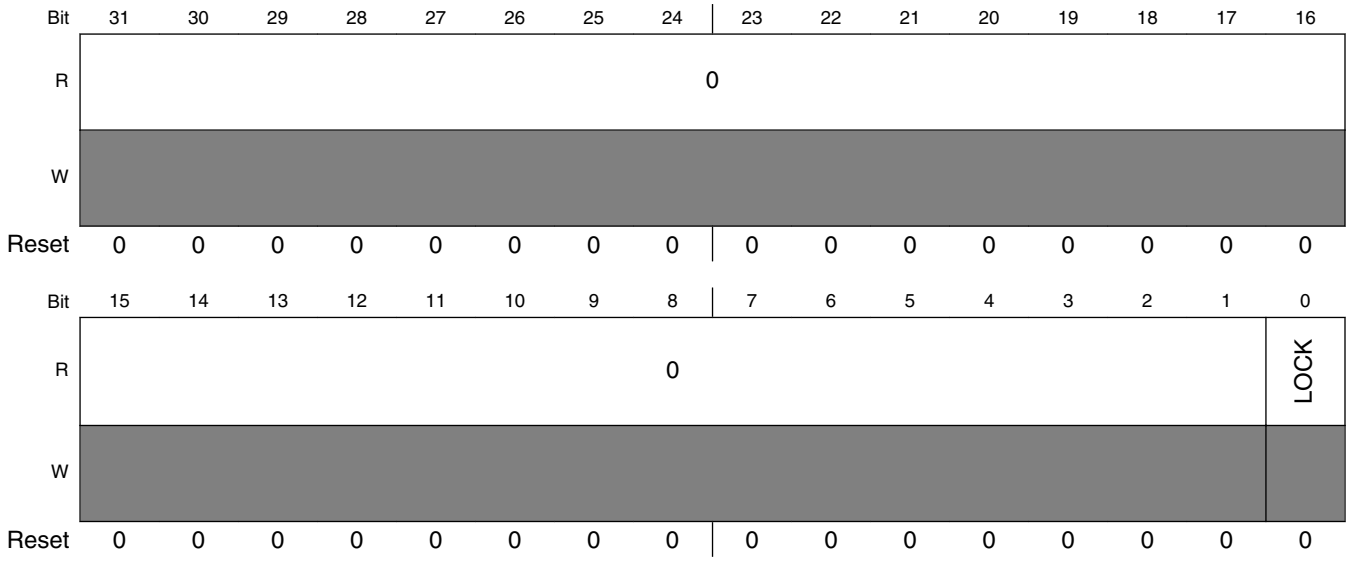


SDMACORE_ENDIANNESS field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value zero. Reserved
2–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 APEND	APEND indicates the endian mode of the Peripheral and Burst DMA interfaces. This bit is tied to logic '1' indicating little-endian mode. 0 - ARM platform is in big-endian mode 1 - ARM platform is in little-endian mode

46.22.20 Lock Status Register (SDMACORE_SDMA_LOCK)

Address: SDMACORE_SDMA_LOCK is 63FB_0000h base + 1Eh offset = 63FB_001Eh

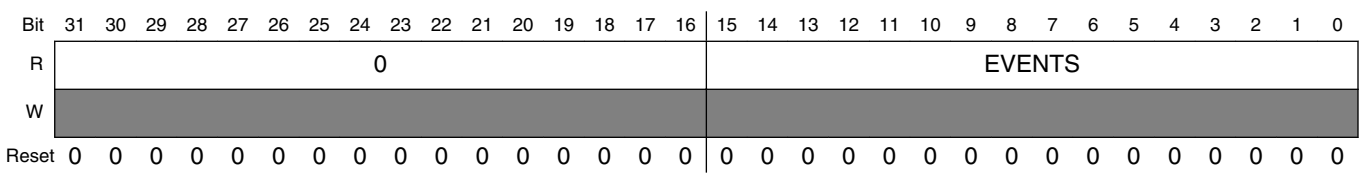


SDMACORE_SDMA_LOCK field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 LOCK	The LOCK bit reports the value of the LOCK bit in the SDMA_LOCK status register. SDMA software may use this value to determine if certain operations such as loading of new scripts is allowed. 0 - LOCK bit clear 1 - LOCK bit set

46.22.21 External DMA Requests Mirror #2 (SDMACORE_EVENTS2)

Address: SDMACORE_EVENTS2 is 63FB_0000h base + 1Fh offset = 63FB_001Fh



SDMACORE_EVENTS2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

SDMACORE_EVENTS2 field descriptions (continued)

Field	Description
15–0 EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 32-47. The separate EVENTS register displays events 0-31.

46.23 SDMA Peripheral Registers

Refer to the respective peripherals' chapters more information.

Chapter 47
System JTAG Controller (SJC)

47.1 Introduction

The System JTAG Controller (SJC) provides debug and test control with the maximum security. The test access port (TAP) is designed to support features compatible with the IEEE Standard 1149.1 v2001 (JTAG).

The figure below shows an overview of the JTAG architecture.

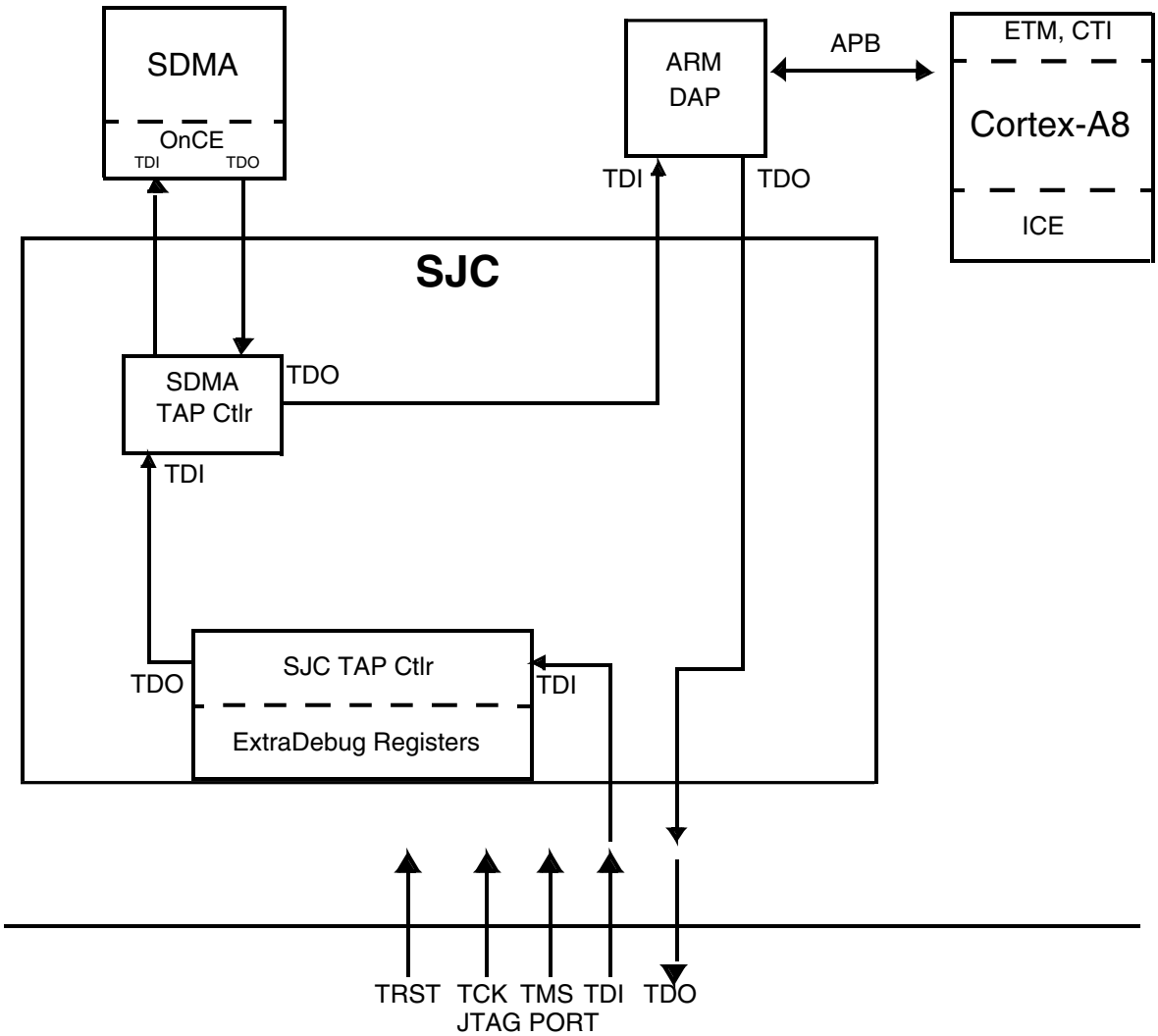


Figure 47-1. System JTAG Controller (SJC) Block Diagram

47.1.1 Overview

The System JTAG Controller (SJC) provides the following capabilities:

- Supports JTAG IEEE1149.1 mandatory instructions, see EXTEST Instruction, SAMPLE/PRELOAD Instruction, and BYPASS Instruction.
- Supports JTAG IEEE1149.1 optional instructions, see ID_CODE Instruction (IDCODE), and HIGHZ Instruction.
- Supports JTAG IEEE P1149.1 (standard JTAG) interface to off-chip test and development equipment including an SJC-only mode for true IEEE 1149.1 compliance, used primarily for board-level implementation of boundary scan.

- Debug-related control and status, such as putting selected cores into reset and/or debug mode and the ability to monitor individual core status signals via JTAG.
- Provides means for accessing each OnCE/ICE TAP controller independently to control a target system (see Modes of Operation).
- Provides means for accessing the ExtraDebug logic (see ENABLE_ExtraDebug Instruction).
- The maximum clock speed of the SJC is one-eight of the lowest frequency of the accessed OnCE/ICE. For example in normal operation (no core in low-power mode), this frequency is one-eight of the SDMA frequency if this core is present in the TDI-TDO chain (serially connected with other cores or standalone). The user must also consider the 25 MHz frequency limitation on the CE bus.
- Core compliant modes to support standalone core debuggers (see Modes of Operation).
- Multi-cores daisy chained mode (default one) to support multi-core debuggers (see Modes of Operation).

Detailed information about the SJC is provided in the Security Reference Manual. Contact your Freescale representative for information about obtaining this document.

47.2 Modes of Operation

The SJC modes are controlled through both the TAP select register (SJC_TSR) and the "sjc_mod" input port.

The "sjc_mod" port (typically connected to pad of the same name: SJC_MOD) selects between two possible topologies of TAP connections, as seen at SoC level:

- Negating it (this should be the default state) selects all the TAPs (SJC, SDMA and DAP and ARM/ETM) to be connected in the TDI-TDO chain, which is referred to as "daisy chain" mode, throughout this chapter.
- Asserting it only selects the SJC TAP to be connected in the TDI-TDO chain.

IEEE1149.1 standard features are enabled by configuring the SJC input pin: SJC_MOD. Refer to the following table for SJC_MOD settings details:

Table 47-1. SJC Modes

SJC_MOD	Name	Description
0	Daisy chain ALL	For common SW debug (High speed and production)
1	SJC only	IEEE 1149.1 JTAG compliant mode

The following figure shows the SJC mode selection flow. The numbers shown in parenthesis below each block name indicates the TAP's IR length.

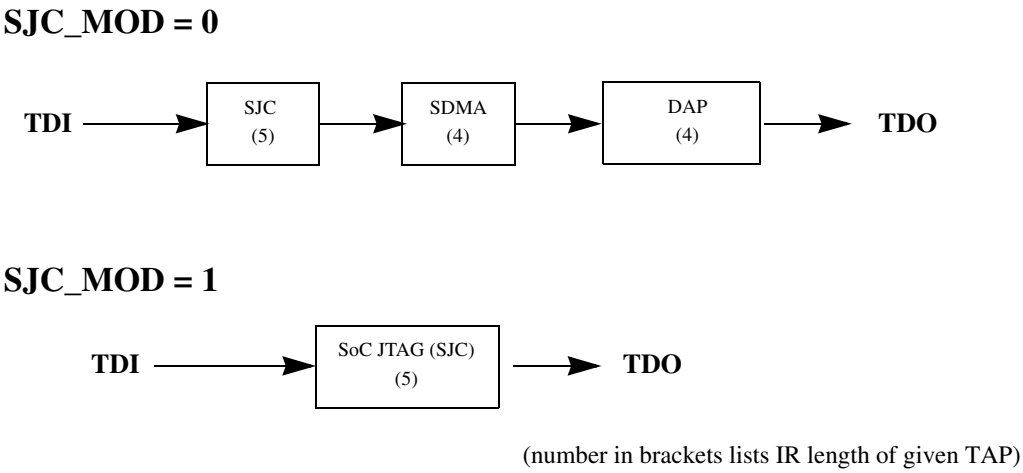


Figure 47-2. SJC Mode Selection Using sjc_mod Pin Sampling

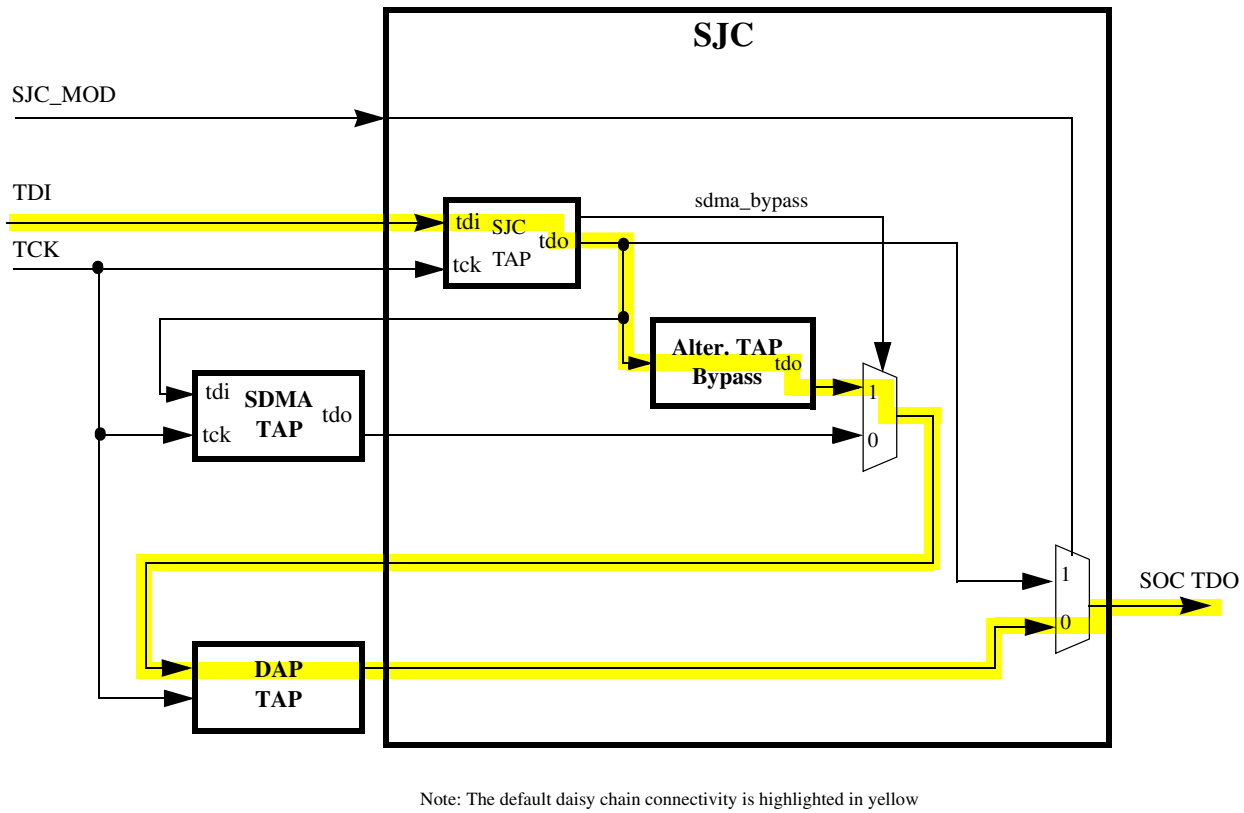
The Connect SDMA bit inside TAP select register controls the SDMA TAP bypass.

- When negated (should be the default state), the SDMA TAP is bypassed with a single D-FF (Flip-flop) during Shift-Dr path
- When asserted SDMA TAP is connected inside the chain
- When taking the SDMA into bypass or out of bypass (by writing to tapsel reg), additional cycle with TMS '0' should be given

The TAP selection block (TSB) provides a simple method of integrating various pieces of IP that have embedded TAPs.

- Provides a way to connect up multiple TAPs within a single SoC
- Identify the SJC TAP as the master TAP which controls the boundary chain (for IEEE 1149.1 standard compliance)
- Follow the state of SJC TAP, and when the Test-Logic-Reset (TLR) state is reached, reset all TAPs

The figure below shows the TAP Selection Block and SOC TAP Chain Scheme.



Note: The default daisy chain connectivity is highlighted in yellow

Figure 47-3. TAP Selection Block and SoC TAP Chain Scheme

NOTE

It is the responsibility of the user to ensure that in any configuration of the TAP controllers chosen, all of the TAPs in the chain comply with the demands of TCK clock frequency as well as the required ratio between TCK clock frequency and that of the core's to which the TAP refers.

47.3 TAP Selection Block (TSB)

As described in Modes of Operation, the SJC can access cores in different modes selected through a TSB.

47.3.1 Select Mode Using Software

Conceptually, the SJC_TSR is a data register which is accessed through Access TSR IR instruction of SJC TAP.

The following figure shows the process of using reserved IR to access the SJC_TSR.

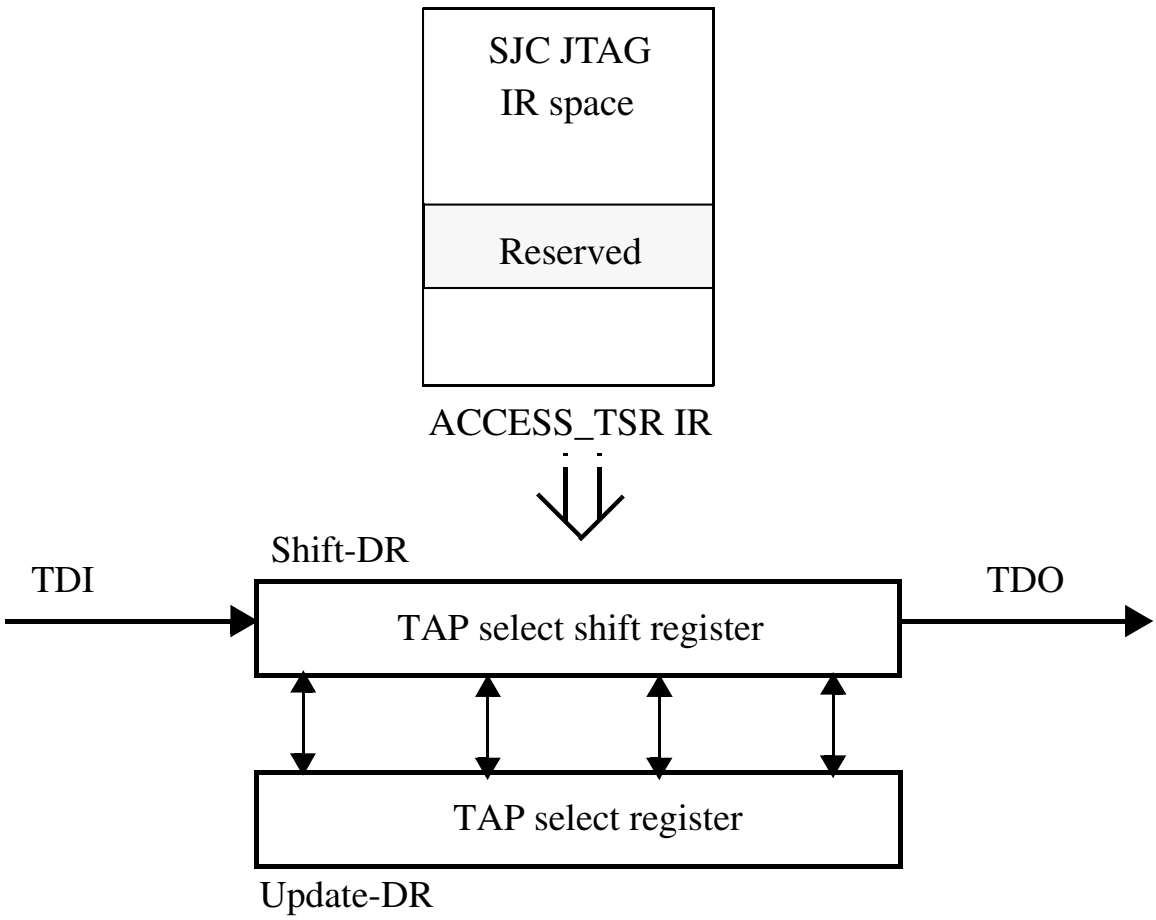


Figure 47-4. Using Reserved IR to Access the TAP Select Register (SJC_TSR)

The SJC_TSR can only be changed during the update-DR state of the TSB JTAG state machine. This is necessary to prevent a TAP that is being selected from losing synchronization with the TSB state machine when the TSB state machine returns to run-test-idle. Therefore, an associated shift register for the SJC_TSR is loaded into the SJC_TSR during the update-DR state (see the figure above). The shift register must also capture the state of the SJC_TSR when in the Capture-DR state for visibility of the contents of the SJC_TSR. See TAP Select Instruction, for more information.

47.4 External Signal Description

47.4.1 External Signal Overview

The SJC provides test and debug control with a minimum number of contacts.

Figure 47-5 shows SJC connections to external contacts and other chip blocks.

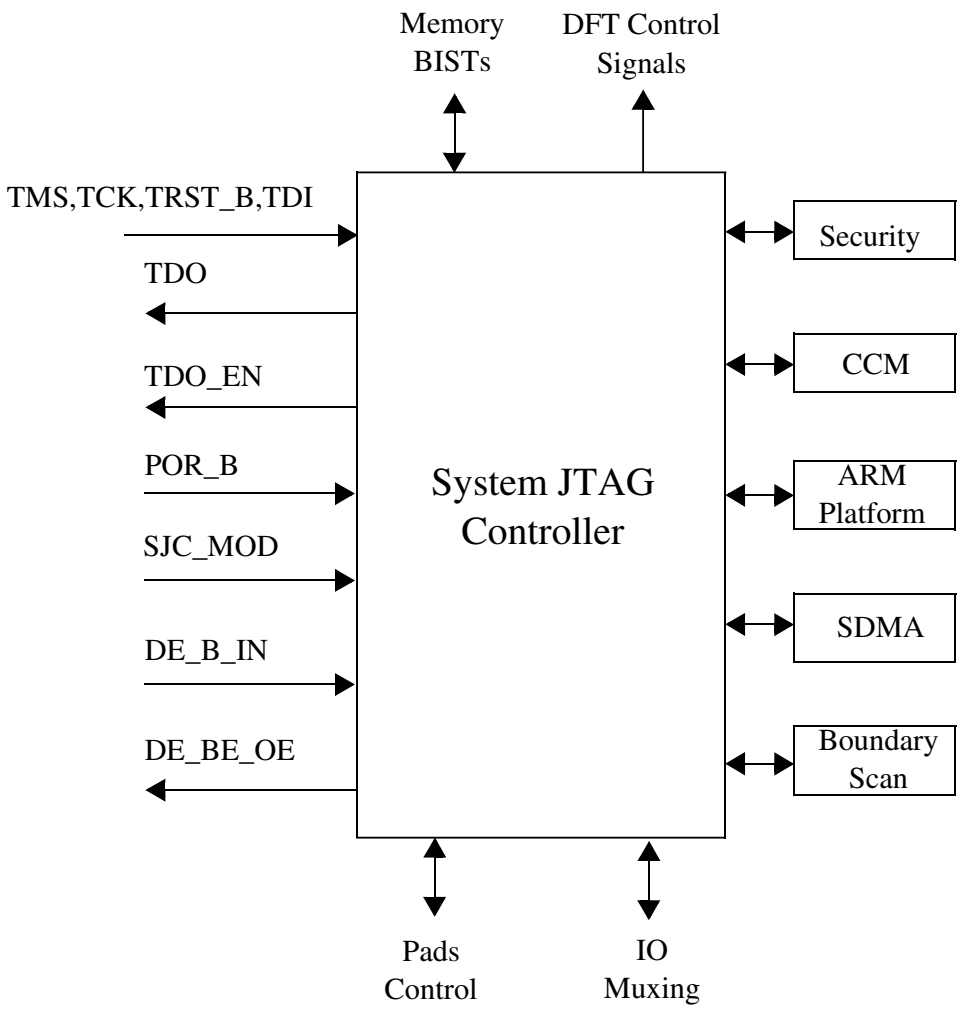


Figure 47-5. SJC Connections

SJC external connections are described along with the signal properties in the table below.

Table 47-2. SJC Signal Properties

Signal	Function	Reset State	Pull up
POR_B	POR reset input. This input can arrive from either pad of IC or from IC's internal logic, for example, Clock Controller.	0	-

Table continues on the next page...

Table 47-2. SJC Signal Properties (continued)

Signal	Function	Reset State	Pull up
TCK	Test Clock (TCK). This is used to synchronize the test logic and includes an internal pull-up resistor	0	-
TDI	Test Data Input (TDI). Serial test instruction and data are received through the test data input (TDI) pin. TDI is sampled on the rising edge of TCK and includes an internal pullup resistor	1	Active
TDO	Test Data Output (TDO). The serial output for test instructions and data. TDO is tri-statable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK	-	-
TDO_EN	Test Data Output Enable (TDO_EN). This signal enables tri-state buffer which output is connected to TDO pad and input connected to IC internal SJC generated TDO signal. Whenever TDO_EN is negated TDO is tri-stated.	-	-
TMS	Test Mode Select (TMS). This is used to sequence the test controller's state machine. TMS is sampled on the rising edge of TCK and includes an internal pullup resistor	1	Active
TRST_B	Test Reset (TRST). This is used to asynchronously initialize the test controller. The TRST pin has an internal pullup resistor	1	Active
SJC_MOD	SJC mode selection. This pin is sampled at TRST reset to determine two possible modes for the TAP connection configuration.	11	Active
DE_IN_B	SoC debug request/acknowledge pin. The DE_IN_B pin is used to propagate an external debug request event to the core(s). This functionality must be enabled first, by set of DE_to_ARM / DE_to_SDMA bits in SJC's DCR register. It is SoC implementation dependent, whether this pin can also be used to reflect the debug acknowledge event back from the cores (in the case where an Open-Drain scheme is used externally).	1	Active

47.4.2 TAP Controller

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of TCK signal. For a description of the TAP controller states, refer to the appropriate IEEE 1149.1 document.

The state machine is shown in the following figure.

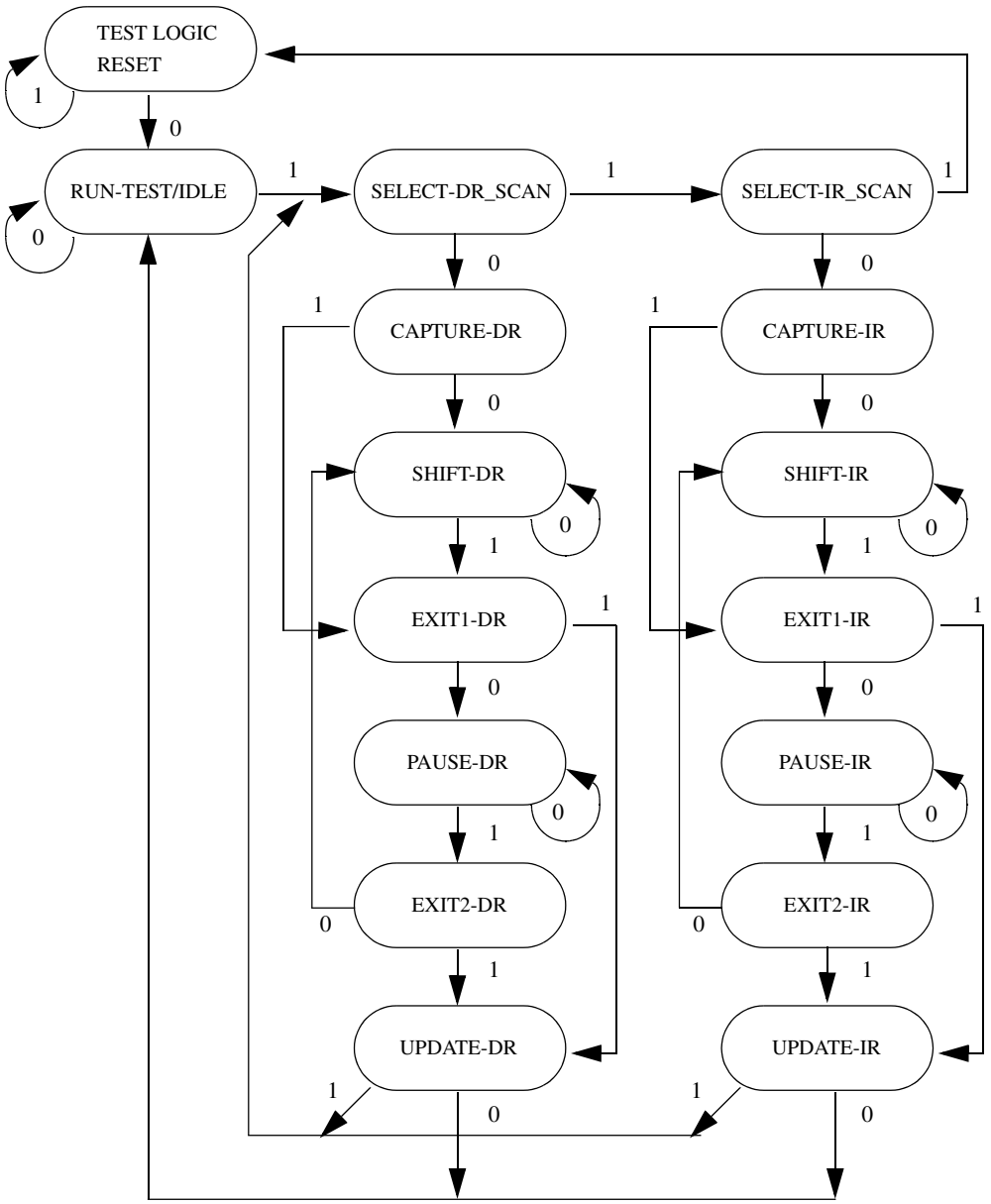


Figure 47-6. TAP Controller State Machine

The change of the JTAG state machine occurs on the rising edge of TCK. TMS and TDI change on the falling edge of TCK. TDO also changes on the falling edge of TCK following entry into the Shift_DR or Shift_IR states (TDO_EN is the enable of the tristate buffer driving the TDO output).

The figure below shows the timings of the SJC signals.

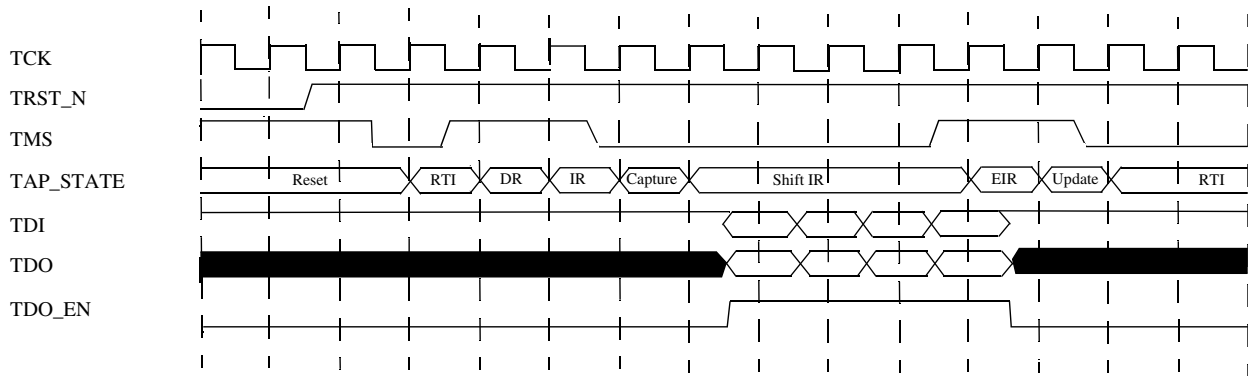


Figure 47-7. SJC Signals Timing Diagram

47.4.3 Accessing ExtraDebug Registers

Accessed through the Select-DR-Scan path, the ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bit data field (max length, see extradebug register description), a 5 bit address field and read/write bit.

The write actually takes place when the JTAG TAP controller enters the Update-DR state. On a read, the data field is ignored (the user should shift only 5 times to enter Read=1 and the address), the read takes place on the next path through DR at the Capture-DR state, the data is shifted-out during the Shift-DR state.

On the second path for a read access, simultaneous write access is not supported: command converter software shifts in zeros so the TAP decodes a write to the CSR (read-only register) which does not have any effect on the circuit.

The number of shift depends on the width of the accessed register as explained in the following diagrams.

First a write access (one path through Select-DR-Scan):

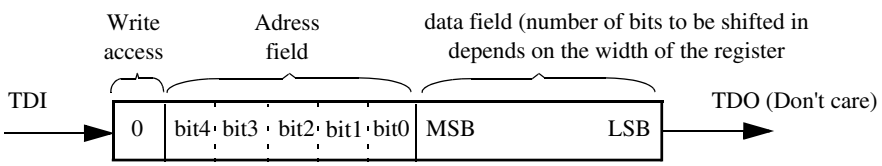


Figure 47-8. TDI/TDO on write access

Then a read access (requires two paths through Jtag DR Scan path):

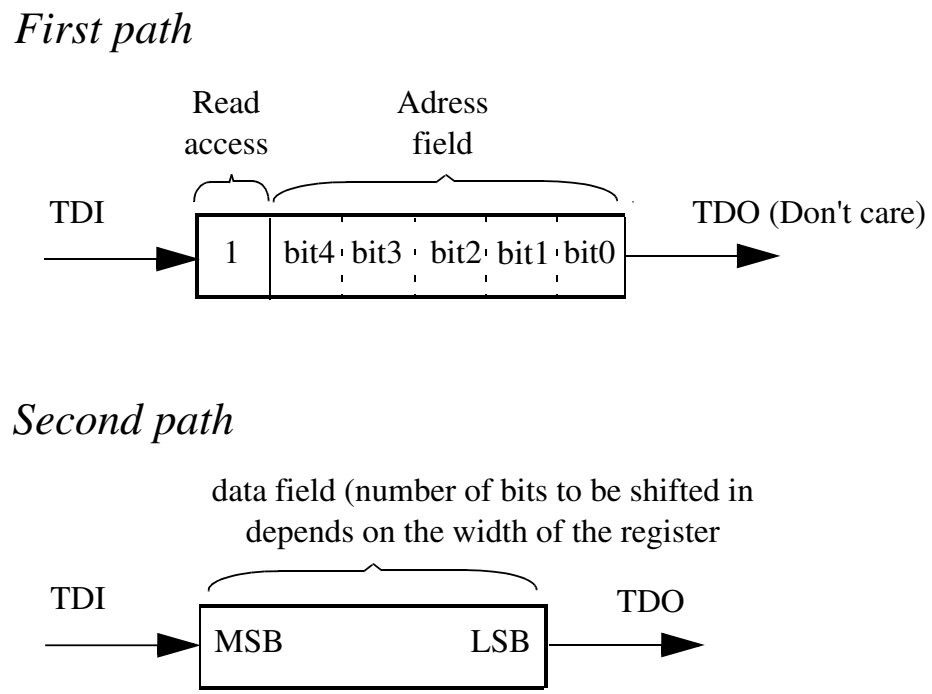


Figure 47-9. TDI/TDO on Read Access

For example, write value 0b1010_1100 to Debug Control Register (address = 0b00110).

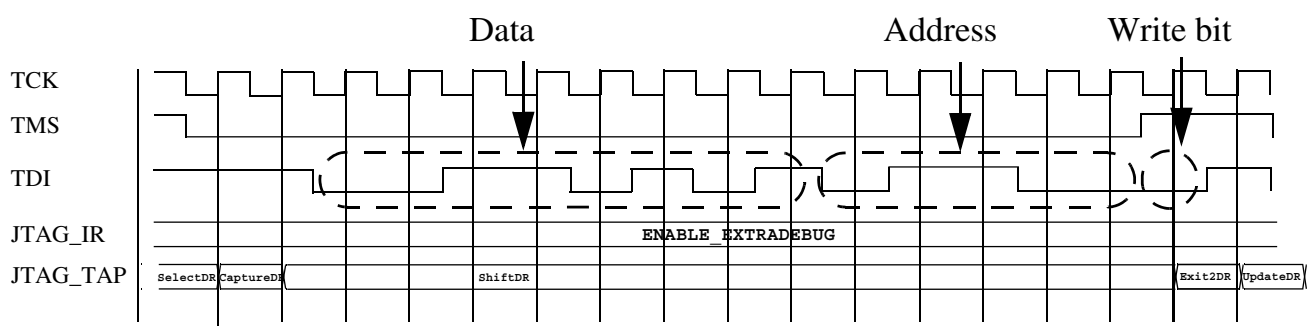


Figure 47-10. Example: Write Access to DCR

The SJC registers have different levels of security (refer to JTAG Security Modes):

- Secured- accessible only in mode 2 (supposed correct response entered), mode 3 and mode 4.
- Unsecured- accessible in all modes

The level of security of each register is indicated in its name or description, in "Programmable Registers" section.

A single DE_B pin is dedicated for debug request input/output in bidirectional open drain functionality (including an internal pull-up device).

The main function of the embedded cross trigger is to pass debug events from one core to another. For example this can be programmed to pass debug request from one block as debug request to ARM so that ARM enters debug mode when the debug request is generated. ECT system_debug is connected to SJC and can be programmed for observability on $\overline{DE}$ pad.

Bits 6:4 in DCR register serve as mask bits, controlling the propagation of external debug request to each recipients (ARM Platform, SDMA).

The bits 2:0 define the propagation enable of IR debug request to recipient cores.

Figure 47-11 shows the $\overline{DE}$ Pin Select Logic.

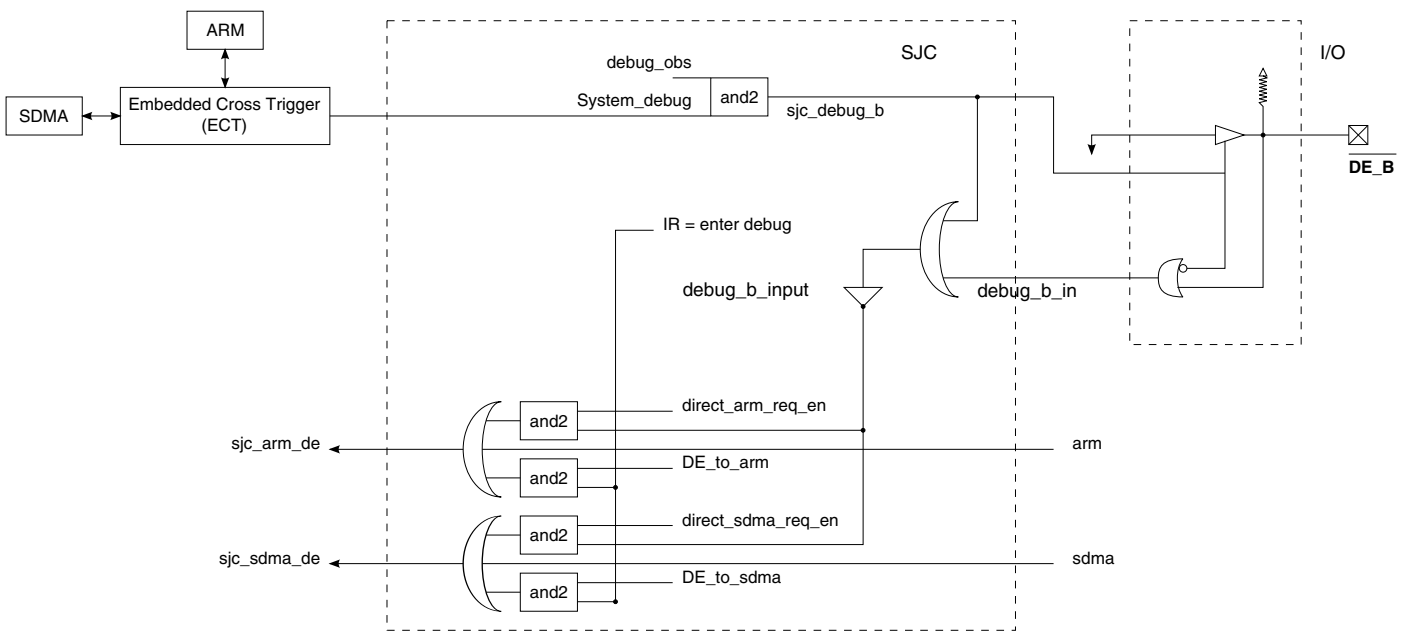


Figure 47-11. $\overline{DE}$ Pin Select Logic

For security reasons, bits for output and input propagation control are at their negated values after reset. A user cannot put the cores in debug mode through $\overline{DE}$ without any Jtag access.

The configuration after reset prevents propagation of debug requests / acknowledges to or from the cores.

47.5 Boundary Scan Register (BSR)

The Boundary Scan Register (BSR) in the JTAG implementation contains bits for all device signal and clock pins and associated control signals. All SoC bidirectional pins have a single register bit in the boundary scan register for pin data, and are controlled by an associated control bit in the boundary scan register.

47.6 SoC JTAG Instruction Register (SJIR)

The SoC JTAG Instruction register is 5 bits wide.

Table 47-3. SoC JTAG Instruction Register (SJIR)

Code					SJC IR
B4	B3	B2	B1	B0	
0	0	0	0	0	IDCODE
0	0	0	0	1	SAMPLE/PRELOAD
0	0	0	1	0	EXTEST
0	0	0	1	1	HI-Z
0	0	1	0	0	ENABLE_ExtraDebug
0	0	1	0	1	ENTER_DEBUG (secured)
0	0	1	1	0	Reserved
0	0	1	1	1	TAP select
0	1	0	0	0	Reserved
0	1	0	0	1	Reserved
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Security Output challenge
0	1	1	0	1	Security Enter response
-					Reserved
1	1	1	1	1	BYPASS

The instruction register is reset to 0b00000 in the test-logic-reset controller state which is equivalent to the IDCODE instruction.

During the capture-IR controller state, the parallel inputs to the instruction register are loaded with the code 01 in the least significant bits as required by the standard; the most significant bits are loaded with the values 00, leading to a capture value of 0b00001.

47.6.1 ID_CODE Instruction (IDCODE)

Selects the ID register, and the system logic controls the I/O pins. This instruction is provided as a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP.

The table below shows the ID register configuration.

Table 47-4. ID Configuration Register (IDCODE)

IDCODE				ID Configuration Register												
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Version Information[3:0]				Design Center Number						Core number					
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	z	z	z	z	z	z	y	y	y	y	y	x
Note:																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Chip Derivative Number				Manufacturer Identity											1
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	x	x	x	0	0	0	0	0	0	0	0	1	1	1	0	1
Note:																

Table 47-5. ID Configuration Register Description (IDCODE)

Field	Description
31-28 Version Information	Version information number This number is incremented for each metal fix of the IC. Bits [31:28] - 0000 (for initial SoC, 0001 for first metal fix, and so on. Refer to System Integration chapter for details on specific SOC.
27-12 Customer Part Number	Customer Part Number The Customer Part Number consists of two parts: Bits [27:22] - Freescale Design Center Number Bits [21:12] - A sequence number divided into two parts: Bits [21:17] - Core Number (SoC project) Bits [16:12] - Chip Derivative Number See SoC specific *Multimedia Applications Processor Reference Manual* document for more information.

Table continues on the next page...

Table 47-5. ID Configuration Register Description (IDCODE) (continued)

11-1 Manufacturer Identity	Manufacturer Identity Freescale's Manufacturer Identity code. Bits [11:1] - 00000001110
0	Tied to logic 1.

NOTE

The IDCODE value is programmed by SOC integration of the SJC block. See "Debug" chapter for exact register value for a specific SOC.

Initial IDCODE value, for Rev 1.0 silicon, are provided in table below:

idvia	I	Version Information Bits 31-28[1]	Customer Part NumberBits [27-12]			Manufacturer Identity Bits 11-1 (Freescale)	0	Final number (Hex)
			Bits [27-22] - Freescale internal numbering.	Bits [21-17] - Core Number (SoC project)	Bits [16-12] - Chip Derivative Number			
i.MX50	I	0000	001011	01000	00001	00000001110	1	02D0101D
Tortola 1.0	I	0000	000110	01000	00001	00000001110	1	0190101D
Tortola1.1	I	0001	000110	01000	00001	00000001110	1	1190101D

1. Version number may vary based on new releases. Please refer to "Debug" chapter for exact version number listing.

One application of the ID register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge which conform to the IEEE 1149.1 standard, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design to determine the type of each component in each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

Once the IDCODE instruction is decoded, it selects the ID register which is a 32 Bit data register. Because the bypass register loads a logic 0 at the start of a scan cycle, whereas the ID register loads a logic 1 into its least significant bit, examination of the first bit of data shifted out of a component during a test data scan sequence immediate following exit from Test-Logic-Reset controller state shows whether such a register is included in the design. When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic as required by the IEEE 1149.1 standard.

47.6.2 SAMPLE/PRELOAD Instruction

Selects the boundary scan register and the system logic controls the I/O pins.

The SAMPLE/PRELOAD instruction provides two separate functions:

- First, it provides a means to obtain a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR controller state. The data can be observed by shifting it transparently through the boundary scan register.
- The second function of SAMPLE/PRELOAD is to initialize the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data appears on the outputs when entering the EXTEST instruction.

NOTE

Because there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results.

For more details on the function and use of SAMPLE/PRELOAD, refer to the appropriate IEEE 1149.1 document.

47.6.3 EXTEST Instruction

Selects the boundary scan register, and the 1149.1 test logic has control of the I/O pins.

By using the TAP controller, the register is capable of:

- Scanning user-defined values into the output buffers,
- Capturing values presented to input pins
- Controlling the direction of bidirectional pins,
- Controlling the output drive of tri-statable output pins.

For more details on the function and use of EXTEST, refer to the appropriate IEEE 1149.1 document.

The EXTEST instruction also asserts internal reset for the cores (through CCM, refer to Figure 47-13) to force a predictable internal state while performing external boundary scan operations.

47.6.4 HIGHZ Instruction

All output drivers, including the two-state drivers, are turned off (that is, high impedance). The instruction selects the bypass register. In this mode, all internal pullup resistors on all the pins (except for the TMS TDI TCK $\overline{\text{TRST}}$ pins) are disabled. This disabling functionality is not built into SJC, but should be implemented by some logic in the SOC/IO Pads.

For more details on the function and use of HIGHZ, refer to the IEEE 1149.1 document.

The HIGHZ instruction also asserts internal reset for the cores (through CCM, refer to Figure 47-13) to force a predictable internal state while performing external boundary scan operations.

47.6.5 BYPASS Instruction

Selects the single Bit bypass register and the system logic controls the I/O pins. This creates a shift-register path from TDI to the bypass register and, finally, to TDO, circumventing the boundary scan register. This instruction is used to enhance test efficiency when a component other than the SoC Core based device becomes the device under test.

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register is always a logic zero.

For more details on the function and use of BYPASS, refer to the appropriate IEEE 1149.1 document.

47.6.6 ENABLE_ExtraDebug Instruction

The TDI and TDO pins are connected directly to the ExtraDebug registers, the SJC TAP controller remaining connected to TDI and TMS.

The ExtraDebug shift register consists of 38 bits (maximum) comprising a 32-bits data field (maximum length, see Accessing ExtraDebug Registers,), a 5 bits address field and read/write bit. On a register read, the data field does not need to be filled in. The particular ExtraDebug register connected between TDI and TDO at a given time is selected by the ExtraDebug controller depending on the ExtraDebug Address being currently decoded. All communication with the ExtraDebug controller is done through the Select-DR-Scan path of the JTAG TAP Controller.

47.6.7 ENTER_DEBUG instruction

The ENTER_DEBUG instruction is used to generate a debug request event to SDMA and the ARMCore Platform simultaneously (practically, inherited minimal skew is expected, due to difference in event signal propagation in the different modules).

The TDI and TDO are connected to the Instruction Register (IR). After the acknowledgment of the Debug Mode is received (can be checked by reading the Core Status Register part of the ExtraDebug logic), the user can perform system debug functions on the cores.

NOTE

The ENTER_DEBUG event issue to the cores, can be masked, by bits in DCR register.

It is user's responsibility to shift-in another IR value (like IDCODE) before trying to bring the cores out of debug mode, as the debug request signals to the cores remains asserted as long as ENTER_DEBUG IR is in place.

The user need to check that cores are in debug mode (watching debug acknowledge signal) before leaving ENTER_DEBUG instruction, otherwise debug request might not take affect.

47.6.8 TAP Select Instruction

By means of TAP select instruction a user can access TAP select register and by controlling its only bit SDMA Bypass, control whether SDMA TAP is bypassed or not.

Table 47-6. TAP Select Register (TSR)

	TAP Select Register
	BIT 0
	Connect SDMA
TYPE	rw
RESET	0
Note:	

Table 47-7. TAP Select Register Description

Field	Description
0 SDMA Bypass	Connect SDMA Control whether SDMA TAP is bypassed or not: • 0 - SDMA TAP is bypassed by the alternate TAP inside SJC (emulating 4-bit IR and 1-bit bypass path). • 1 - SDMA TAP is connected to the TDI-TDO chain. **NOTE:** Additional cycle with TMS '0' should be inserted, after writing to this register, to allow the SDMA tap be sync before SDMA get into / out of bypass.

47.7 Security

JTAG manipulation is one of the known hackers' ways of executing unauthorized program code, getting control over the OS and run code in privileged modes.

The SJC provides a debug access to several H/W blocks including the ARM processor and the system bus. This allows for program control and manipulation as well as visibility into system peripherals and memory. The bus transactions to be traced. Together these tools provide the hacker all the access needed to completely comprise the system. Means must be provided to block any malicious JTAG access.

The SJC provides a way of regulating the JTAG access.

The following are the different JTAG security modes:

- Mode #1: No Debug-Maximum Security. All security sensitive JTAG features are permanently blocked.
- Mode #2: Secure JTAG-High security. JTAG use is regulated by secret key based authentication mechanism.
- Mode #3: JTAG Enabled-Low security. JTAG always enabled.

The JTAG security modes are configured using eFUSEs which can burned after packaging by applying electrical signals. The fuse burning is irreversible process, once a fuse is burned (e-fuse or laser fuse) it is impossible to change the fuse back to the un-burned state.

47.7.1 JTAG Security Modes

JTAG can be in one of JTAG security modes which is selected by setting the SJC eFUSE configuration. The physical location of the fuses is not in the SJC.

47.7.1.1 Mode 1: No Debug - Maximum Security

No Debug JTAG security mode provides the highest security level.

In this mode, all JTAG features are disabled except for:

- ScanBoundary Scan
- MBIST, all modes except for debug modes which enable controlled memory contents output
- PLL BIST

- BIST monitor mode, allowing routing to external pins BIST pass/fail/invoke information
- PLL bypass- Bypass ARM or/and USB PLL.
- Visibility of the following status bits: power mode - normal, standby, stop, shutdown, and so on

These features do not reduce the security level of the product, and they allows to perform important tests and board connectivity checks.

47.7.1.2 Mode 2: Secure JTAG - High Security

The Secure JTAG mode limits the JTAG access by using challenge/response based authentication mechanism. Any access to JTAG port is being checked. Only authorized debug devices (that is, devices having the right response) can access the JTAG, unauthorized JTAG access attempts are denied.

The intent of this mode is to allow return field testing. When a secured JTAG device is being returned for debugging, this mode allows authorized re-activation of the JTAG.

47.7.1.2.1 Challenge/Response Mechanism in System JTAG Mode

When SJC is in Sysytem JTAG mode the authentication process is as follows:

1. Shift Output Challenge instruction to IR.
2. Passing through Capture-DR state of the SJC and by performing Shift-DR operations Challenge code can be accessed from TDO.
3. Shift Enter Response instruction to IR. By performing Shift-DR, operations enter Response code value through TDI. As Update-DR state is entered, Response code is compared with the correct one.

In Fixed challenge-response pair mode, each part has its individual challenge - response pair which is determined at manufacturing time, and does not change later on. The SJC compares the user's response to the expected response.

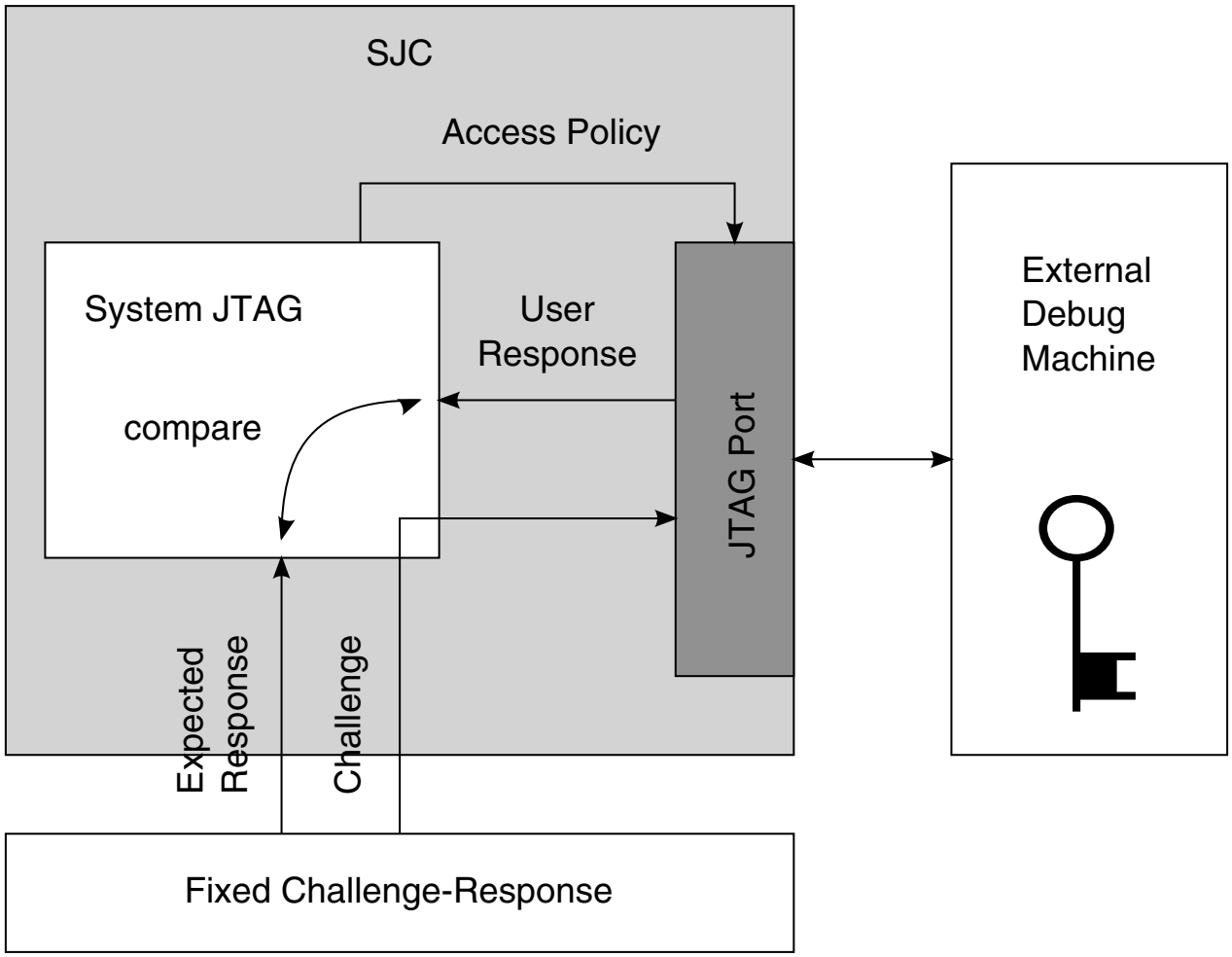


Figure 47-12. Mode #2 - Secure JTAG with Fixed Challenge-response Pair

47.7.1.3 Mode 3: JTAG Enabled - Low Security

In the JTAG Enabled JTAG security mode, all JTAG features are enabled.

47.7.2 Software Enabled JTAG

To increase the flexibility of the SJC, an option to enable the JTAG via software is added and is available only in Secure JTAG mode. By writing '1' to HAB_JDE (HAB JTAG DEBUG ENABLE) bit in the e-fuse controller module (IIM), the JTAG is opened, regardless of its security mode. It is the responsibility of software to assert or negate this bit.

Additionally, a corresponding lock bit is available (in the e-fuse control module) to ensure that only trusted software is able to set the JDE bit. When the LOCK bit is set, no future change of JDE is possible, until the next POR (power-on-reset) cycle.

The platform initialization software should set the LOCK bit for JDE bit before transferring control to the application code.

The S/W JTAG enable allows JTAG enabling without activating the challenge-Response mechanism (which requires JTAG access tool enhancement or special H/W). The JTAG S/W enable does not allow debug in case of boot or memory fault as it requires reset before entering debug.

This feature can be permanently blocked by burning the dedicated e-fuse.

NOTE

The S/W enabled JTAG feature reduces the overall security level of the system as it relies on S/W protections. If this feature is not required, it is strongly recommended to burn the JTAG_HEO e-fuse which disables this feature.

47.7.3 Kill Trace

The kill trace signal disables any output of the block. The can be accessed either via JTAG port and/or by direct software code. Blocking the JTAG port also yields assertion of the kill trace signal. This resulted in blocking of trace port. The intention of this action is to block any attempt to break into the system via software manipulation of the debug modules. The kill trace, when active, prevents trace output even in case where it can be activated via chip pin.

The kill trace feature needs to be activated by burning a dedicated e-fuse. If the fuse is left intact, kill trace is never activated as seen in

The kill trace is asserted when "kill trace enable" fuse is burned and "ipt_secur_block" signal in SJC is asserted, which happens when at least one of the following is true:

- Mode #2 (Secure JTAG) and no code has been entered
- Mode #2 (Secure JTAG) with incorrect response entered
- Mode #1 (No debug)
- TRST_B signal is active
- POR has not ever been asserted

47.7.4 SJC Disable Fuse

In addition to the different JTAG security modes that are implemented internally in the System JTAG Controller (SJC), there is an option to disable the SJC functionality by eFUSE configuration. This creates additional JTAG mode that is, JTAG Disabled with highest level of JTAG protection. In this mode all JTAG features are disabled.

Specifically, the following debug features are disabled in addition to the features that were already disabled in No Debug JTAG mode:

- Memory BIST
- Boundary scan register (SJC_BSR)
- Non-Secure JTAG control registers (PLL configuration, Deterministic Reset, PLL bypass)
- Non-Secure JTAG status registers (Core status)
- Chip Identification Code (IDCODE)

47.8 Functional Description

47.8.1 Static Core Debug

The SJC JTAG TAP controller is fully compatible with the IEEE 1149.1a-2001 Standard Test Access Port and Boundary Scan Architecture specifications.

The SDMA has a TAP controller to manage its own OnCE, see SDMA OnCE specifications for more details.

The OnCE and ICE provide a mean of interacting with the cores and their peripherals non-intrusively so that a user may examine registers, memories to facilitate hardware and software development. Refer to TAP Selection Block (TSB), for more information.

47.8.2 Reset Mechanism

The following figure shows the SJC reset logic

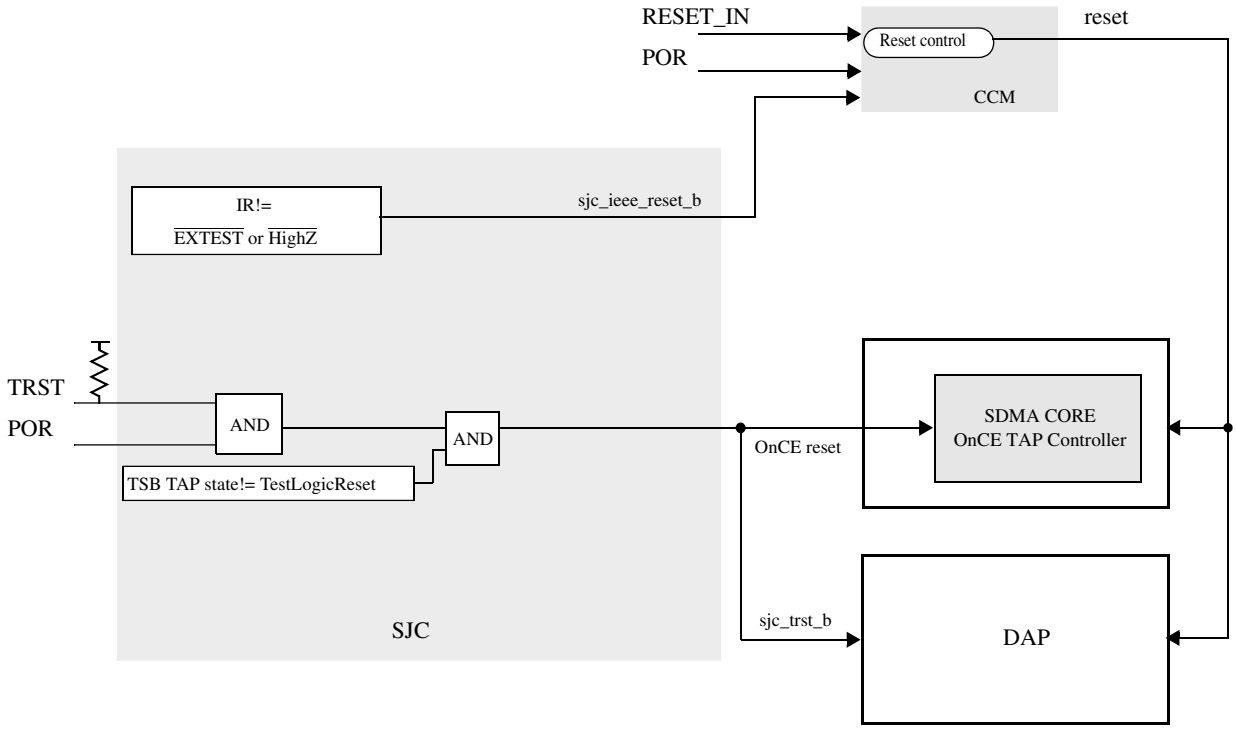


Figure 47-13. SJC Reset Logic

NOTE

- Asserting TRST in any scan mode resets the TCR loosing the testmode configuration and selects default TAP.
- SJC generates an IEEE reset signal to the CCM when in one of the IEEE modes HIGHZ or EXTEST. This signal generates a system reset to the cores until exit from one of these modes.
- The TSB generates Once/ICE reset (either TRST if implemented or other) when its TAP state reaches Test-Logic-Reset (meaning that TAP accessed is also reaching Test-Logic-Reset).

47.9 Initialization/Application Information

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the SJC output drivers are enabled into actively driven networks.

There are two constraints related to the JTAG interface:

- Ensure that the JTAG test logic is kept transparent to the system logic by forcing TAP into the Test-Logic-Reset controller state. During power-up, SJC's internal $\overline{\text{TRST}}$ is asserted as IC's $\overline{\text{POR}}$ is asserted which forces the TAP controller into this state. After that, if TMS either remains unconnected or is connected to VCC, then the TAP controller cannot leave the Test-Logic-Reset state, regardless of the state of TCK.
- $\overline{\text{DE\_B}}$ is an IO pin with pullup and care must be taken of the direction when driving this signal.

47.10 Programmable Registers

In addition to the standard accessible JTAG registers (per IEEE1149.1 standard) listed in SoC JTAG Instruction Register (SJIR), the chip contains the following registers accessed using the ExtraDebug mechanism, controlled via "ENABLE_ExtraDebug" IR instruction.

NOTE

SJC registers are only accessible by JTAG interface. They are not memory mapped to processor address space, so the absolute addresses provided by default in the SJC memory map are not valid.

This section assumes the JTAG controller is accessed in standalone mode or daisy chained (defined by TAP Selection Block) using the appropriate TSB configuration.

See "System Debug" chapter for more details about the general purpose register descriptions that are unique to this chip.

SJC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0000_0000	General Purpose Unsecured Status Register 1 (SJC_GPUSR1)	32	R	0000_0000h	47.10.1/ 2798
0000_0001	General Purpose Unsecured Status Register 2 (SJC_GPUSR2)	32	R	0000_0000h	47.10.2/ 2799
0000_0002	General Purpose Unsecured Status Register 3 (SJC_GPUSR3)	32	R	0000_0000h	47.10.3/ 2800
0000_0003	General Purpose Secured Status Register (SJC_GPSSR)	32	R	0000_0000h	47.10.4/ 2800
0000_0004	Debug Control Register (SJC_DCR)	32	R/W	0000_0000h	47.10.5/ 2801

Table continues on the next page...

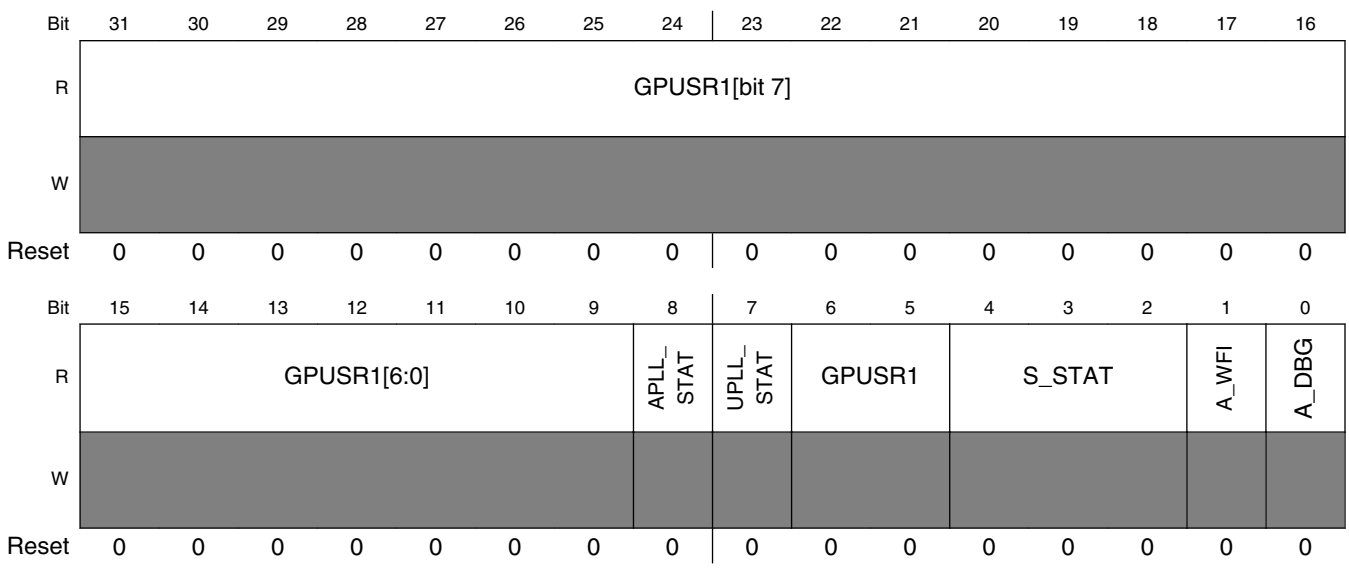
SJC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0000_0005	Security Status Register (SJC_SSR)	32	R	0000_0000h	47.10.6/ 2802
0000_0007	General Purpose Clocks Control Register (SJC_GPCCR)	32	R/W	0000_0000h	47.10.7/ 2804
0000_0008	General Purpose Unsecured Control Register n (SJC_GPUCR)	32	R/W	0000_0000h	47.10.8/ 2805
0000_000B	General Purpose Secured Control Register (SJC_GPSCR)	32	R/W	0000_0000h	47.10.9/ 2805

47.10.1 General Purpose Unsecured Status Register 1 (SJC_GPUSR1)

The General Purpose Unsecured Status Register 1 is a read only registers used to check the status of the different Cores and of the PLL. The rest of its bits are for general purpose use.

Address: SJC_GPUSR1 is 0h base + 0h offset = 0000_0000h



SJC_GPUSR1 field descriptions

Field	Description
31–9 GPUSR1	General Purpose Unsecured Status Register Register is used for testing and debug.
8 APLL_STAT	ARM platform PLL status bit This bits indicates the status of the ARM platform PLL. When this bit is HIGH, the ARM platform PLL is locked.

Table continues on the next page...

SJC_GPUSR1 field descriptions (continued)

Field	Description
7 UPLL_STAT	USB PLL status bit This bits indicates the status of the USB PLL. When this bit is HIGH, the USB PLL is locked.
6–5 GPUSR1	General Purpose Unsecured Status Register Register is used for testing and debug.
4–2 S_STAT	3 LSBits of SDMA core statusH.
1 A_WFI	ARM core wait-for interrupt bit Bit 1 is the ARM core standbywfi (stand by wait-for interrupt). When this bit is HIGH, ARM core is in wait for interrupt mode.
0 A_DBG	ARM core debug status bit Bit 0 is the ARM core DBGACK (debug acknowledge) DBGACK can be overwritten in the ARM core DCR to force a particular DBGACK value. Consequently interpretation of the DBGACK value is highly dependent on the debug sequence. When this bit is HIGH, ARM core is in debug.

47.10.2 General Purpose Unsecured Status Register 2 (SJC_GPUSR2)

Address: SJC_GPUSR2 is 0h base + 1h offset = 0000_0001h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPUSR2																												S_STAT			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SJC_GPUSR2 field descriptions

Field	Description
31–4 GPUSR2	General Purpose Unsecured Status Register Register is used for testing and debug.
3–0 S_STAT	S_STAT

47.10.3 General Purpose Unsecured Status Register 3 (SJC_GPUSR3)

Address: SJC_GPUSR3 is 0h base + 2h offset = 0000_0002h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPUSR3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SJC_GPUSR3 field descriptions

Field	Description
31–0 GPUSR3	General Purpose Unsecured Status Register Register is used for testing and debug.

47.10.4 General Purpose Secured Status Register (SJC_GPSSR)

The General Purpose Secured Status Register is a read-only register used to check the status of the different critical information in the SoC. This register cannot be accessed in secure modes.

Address: SJC_GPSSR is 0h base + 3h offset = 0000_0003h

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

SJC_GPSSR field descriptions

Field	Description
31–0 GPSSR	General Purpose Secured Status Register Register is used for testing and debug.

47.10.5 Debug Control Register (SJC_DCR)

This register is used to control propagation of debug request from DE_B pad to the cores and debug signals from internal logic to the DE_B pad.

Address: SJC_DCR is 0h base + 4h offset = 0000_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DCR[bit 9]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DCR[8:0]									DIRECT_ ARM_REQ_ EN	DIRECT_ SDMA_ REQ_EN	DCR[4]	DEBUG_ OBS	DCR[2]	DE_TO_ SDMA	DE_TO_ ARM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SJC_DCR field descriptions

Field	Description
31–7 DCR	General Purpose Unsecured Status Register Register is used for testing and debug.
6 DIRECT_ARM_ REQ_EN	Pass Debug Enable event from $\overline{\text{DE\_B}}$ pin to ARM platform debug request signal(s). This bit controls the propagation of debug request DE_B to the Arm platform. 0 Disable propagation of system debug to (DE pin) to Arm platform. 1 Enable propagation of system debug to (DE pin) to Arm platform.
5 DIRECT_ SDMA_REQ_EN	Debug enable of the sdma debug request This bit controls the propagation of debug request DE_B to the sdma. 0 Disable propagation of system debug to (DE pin) to sdma. 1 Enable propagation of system debug to (DE pin) to sdma.
4 DCR[4]	Reserved
3 DEBUG_OBS	Debug observability This bit controls the propagation of the "system debug" input to SJC , to the DE_B pad. 0 Disable propagation of system debug to DE pin 1 Enable propagation of system debug to DE pin
2 DCR[2]	Reserved
1 DE_TO_SDMA	SDMA debug request input propagation

Table continues on the next page...

SJC_DCR field descriptions (continued)

Field	Description
	This bit controls the propagation of debug request to SDMA, when the JTAG state machine is put in "ENTER_DEBUG" IR instruction.. 0 Disable propagation of debug request to SDMA 1 Enable propagation of debug request to SDMA
0 DE_TO_ARM	ARM platform debug request input propagation This bit controls the propagation of debug request to ARM platform ("dbgreq"), when the JTAG state machine is put in "ENTER_DEBUG" IR instruction. 0 Disable propagation of debug request to ARM platform 1 Enable propagation of debug request to ARM platform

47.10.6 Security Status Register (SJC_SSR)

Address: SJC_SSR is 0h base + 5h offset = 0000_0005h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SSR[bit 3]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SSR[2:0]			RSSTAT		SJM		FT	BSF	RSF	EBG	EBF	SWE	SWF	KTA	KTF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SJC_SSR field descriptions

Field	Description
31–13 SSR	Security Status Register Register is used for testing and debug.
12–11 RSSTAT	Response status Response status bits 00 Response wasn't entered 01 Response was entered but not verified 10 Response was entered and is incorrect 11 Response is correct
10–9 SJM	SJC mode Secure JTAG mode, as set by external fuses. These bits do not include the setting of the BSF fuse. 00 No debug (#1) 01 Secure JTAG (#2)

Table continues on the next page...

SJC_SSR field descriptions (continued)

Field	Description
	10 JTAG enabled (#3) 11 Reserved
8 FT	Fuse type Fuse type bit - e-fuse or laser fuse 0 E-fuse technology 1 Laser fuse technology
7 BSF	Bypass secure JTAG fuse Status of the bypass secure JTAG fuse. 0 (intact) - no bypass 1
6 RSF	Re-enable secure JTAG fuse Status of the re-enable secure JTAG fuse 0 (intact) - no re-enable 1 (burned) - secure JTAG is re-enabled
5 EBG	External boot granted External boot enabled, requested and granted 1 granted 0 not granted
4 EBF	External Boot fuse Status of the external boot disable fuse 0 (intact) - external boot is allowed 1 (burned) - external boot is disabled
3 SWE	SW enable SW JTAG enable status 1 enabled 0 disabled
2 SWF	Software JTAG enable fuse Status of the no SW disable JTAG fuse 0 (intact) - SW enable possible 1 (intact) - no SW enable possible
1 KTA	Kill Trace is active 1 active 0 not active
0 KTF	Kill Trace Enable fuse value 0 (intact) - kill trace is never active 1 (burned) - kill trace functionality enabled

47.10.7 General Purpose Clocks Control Register (SJC_GPCCR)

This register is used to configure clock related modes in SOC, see System Configuration chapter for more information. Those bits are directly connected to JTAG outputs. Bit 0 of GPCCR controls SDMA clocks invocation. When out of reset, the SDMA is in sleep mode with no SDMA clock running. Unlike events, debug requests does not wake SDMA if it is in sleep mode. The debug request is recognized by the SDMA only when it exits sleep mode upon reception of an event. To be able to enter debug mode even if no event is triggered, the SDMA clock on bit needs to be set prior to sending the debug request (clear at reset).

Address: SJC_GPCCR is 0h base + 7h offset = 0000_0007h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPCCR[bit 13]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPCCR[12:0]													SWRES	ACLKOFFDIS	SCLKR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SJC_GPCCR field descriptions

Field	Description
31–3 GPCCR	General Purpose Clocks Control Register-Register is used for testing and debug.
2 SWRES	SW reset
1 ACLKOFFDIS	Disable/prevent ARM platform clock/power shutdown
0 SCLKR	SDMA Clock ON Register - This bit will force the clock on of the SDMA

47.10.8 General Purpose Unsecured Control Register n (SJC_GPUCR)

Address: SJC_GPUCR is 0h base + 8h offset = 0000_0008h

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

SJC_GPUCR field descriptions

Field	Description
31–0 GPUCR	General Purpose Unsecured Control Register Register is used for testing and debug.

47.10.9 General Purpose Secured Control Register (SJC_GPSCR)

This register is used to configure JTAG for special test or debug modes. This register is secured (accessible in secure jtag mode #3, #4 and #2 with response entered). Those bits are directly connected to SJC outputs.

Address: SJC_GPSCR is 0h base + Bh offset = 0000_000Bh

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

SJC_GPSCR field descriptions

Field	Description
31–0 GPSCR	General Purpose Control Register Register is used for testing and debug.

Chapter 48
Shared Peripheral Bus Arbiter (SPBA)

48.1 Introduction

The Shared Peripheral Bus Arbiter (SPBA) is a three-to-one IP Bus line interface arbiter, with a resource locking mechanism. The masters can access up to thirty one shared peripherals through the SPBA.

The figure below shows the SPBA block diagram

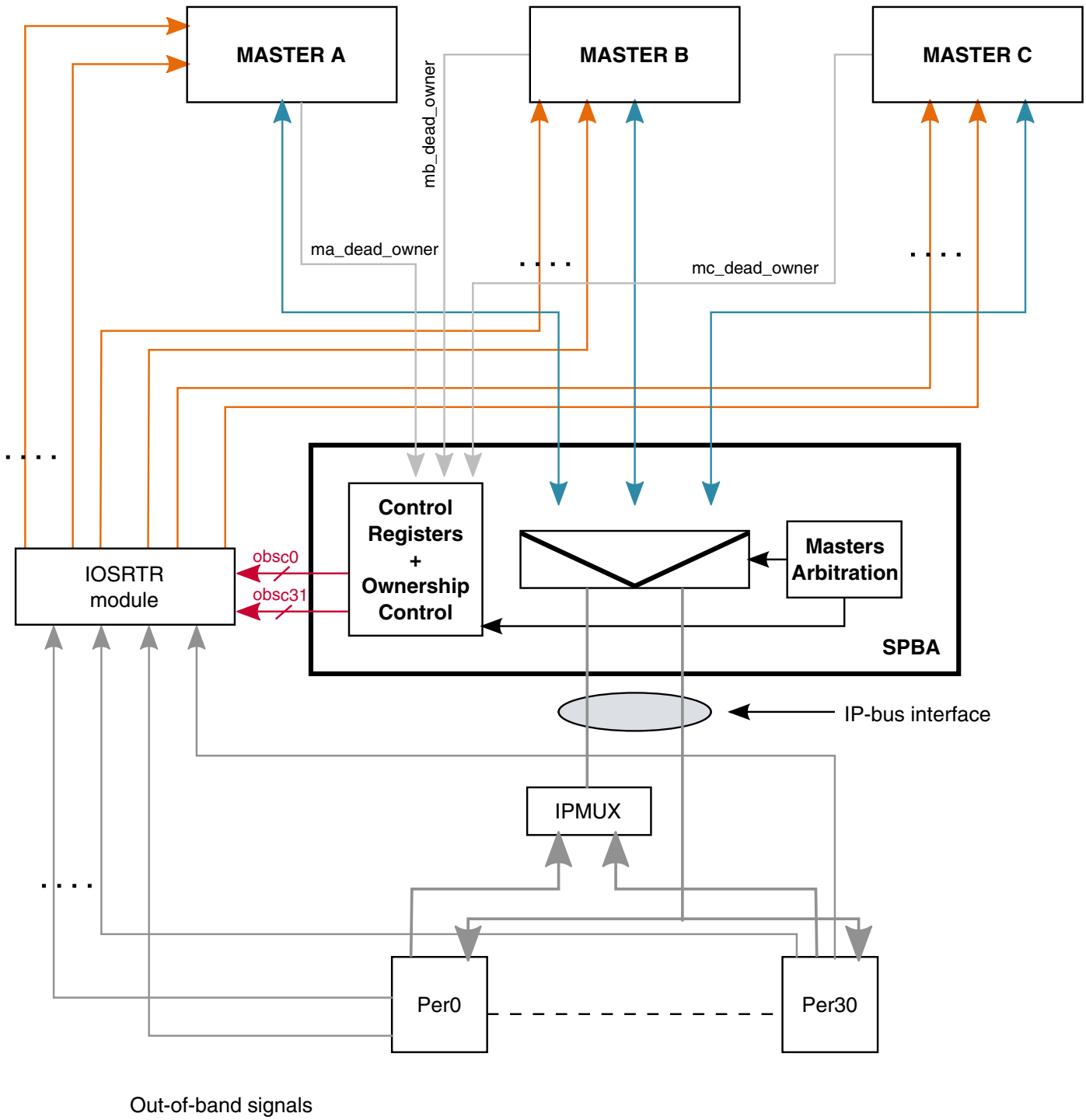


Figure 48-1. SPBA Block Diagram

48.2 General Overview

The Shared Peripheral Bus Arbiter (SPBA) is a three-to-one IP Bus interfaces arbiter. Three masters arbitrate for shared peripherals access through the SPBA.

Indeed three main parts are involved in SPBA.

- The IP Bus Line switches a master to one peripheral.
- The Masters arbiter arbitrates between the three masters to solve concurrent access or restricted access to peripherals.
- The Control Registers and Ownership Control including a set of registers, which are reachable through software, permits the access scheme definition to each peripheral (Resource Ownership and Access Control). It generates signals usable for the external steering logic of interrupts and DMA signals.

48.3 Features

The SPBA includes the following features:

- Three IP Bus masters arbitration, master A, master B and master C
- Support for DMA masters.
- 32 bits data
- Supports up to 31 shared peripherals, each consuming 16 Kilobytes of address space.
- SPBA can be considered as the 32th peripheral, used for resource ownership and access control mechanism to the 31 peripherals,
- Provides 31 sets of Out of Band Steering Control (obsc) signals to the off-block steering logic,
- Operating frequency up to 67 MHz,
- clocks: ipg_clk, ipg_clk_s.

48.4 Modes of Operation

Thus the SPBA behavior is transparent when accessing a peripheral, it can distinguish different modes of operation:

Reset/Abort

The SPBA has a hardware reset which initialize all registers, arbitration and PRR (peripherals rights registers).

Additionally, an abort signal input is provided allowing each master to abort their current access and release their ownership (in case of master reset sequence,...).

Functional

Once a master request is granted, its IP Bus signals are steering to the requested peripheral.

Standby

No clock needed. The SPBA needs clocks only during: access to the PRRs, arbitration and abort phases. It generates two clock enable signals indicating when the clocks must be provided.

Configuration

This is the phase where a master is accessing the SPBA PRRs. Indeed, SPBA memory mapped registers are seen as a shared peripheral.

48.5 Functional Description

The following section intends to describe the main features of the SPBA.

48.5.1 Masters Arbitration

The arbitration mechanism determines which port will control the master port, based on a simple round-robin arbitration scheme.

We can distinguish between different cases:

- Only one master request per different access. So the master is switched to the shared peripheral bus, without arbitration. Figure 48-2 shows the MB request on the global module enable signal, served without wait state.
- If two masters simultaneously access to SPBA, then the last granted master is held-off, using **<master>_ips_xfr_wait** output signal (default value is high). When the master is granted **sips_xfr_wait** from shared IP Bus peripheral is connected to <master>_ips_xfr_wait outputs.
- If three masters simultaneously access to SPBA, then the last two granted masters are held-off, using their <master>_ips_xfr_wait signal. Figure 48-3 shows the case when the last two accesses granted are MA then MB, and how the requests are used even if they are in the same cycle.
- After reset, at the first multiple access, no master has been granted, thus, the priority is static: Master A (MA), Master B (MB) and last Master C (MC) port.
- No master request. No master switch to shared peripherals.

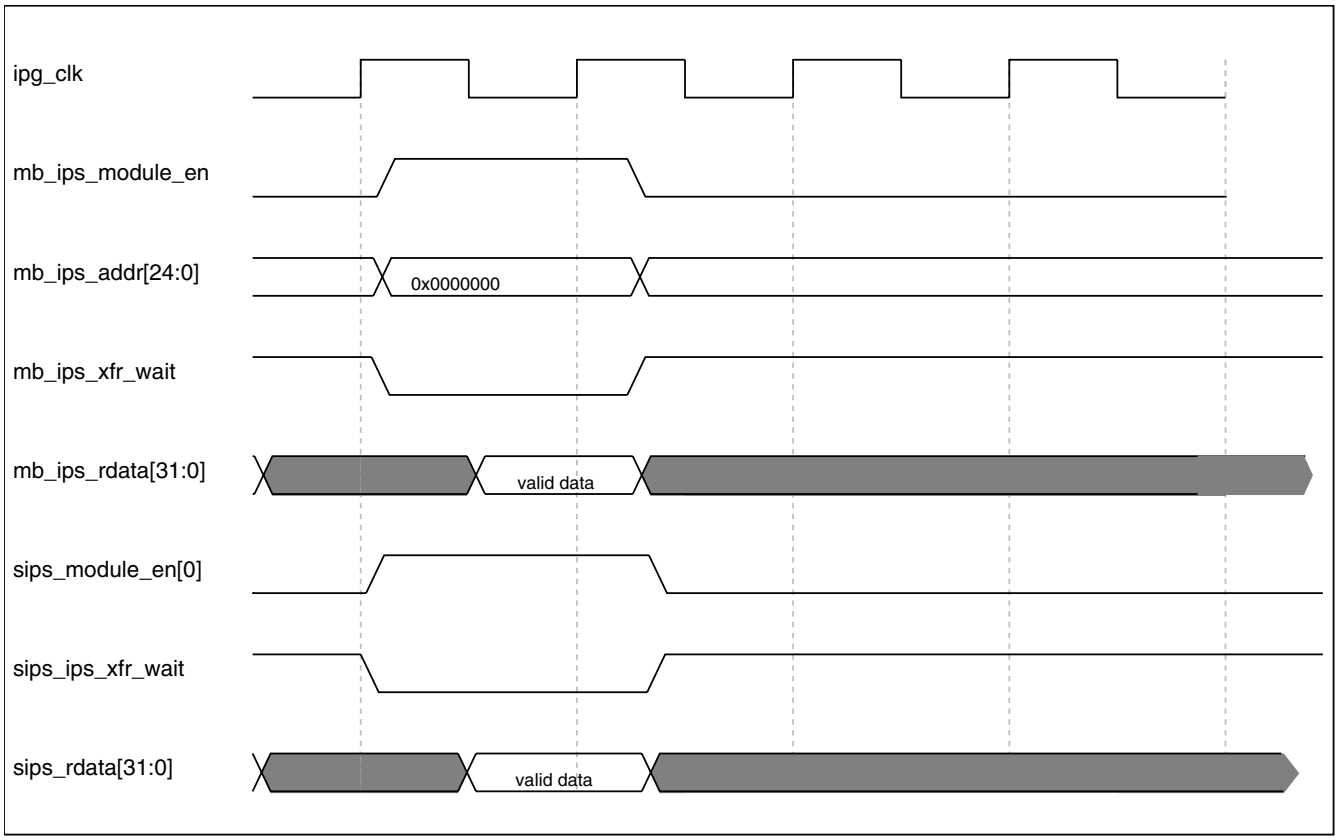


Figure 48-2. Example of one master request: no SPBA Arbitration;

The following figure assumes MA and MB have been the last two masters already granted in the previous transfers (MA then MB).

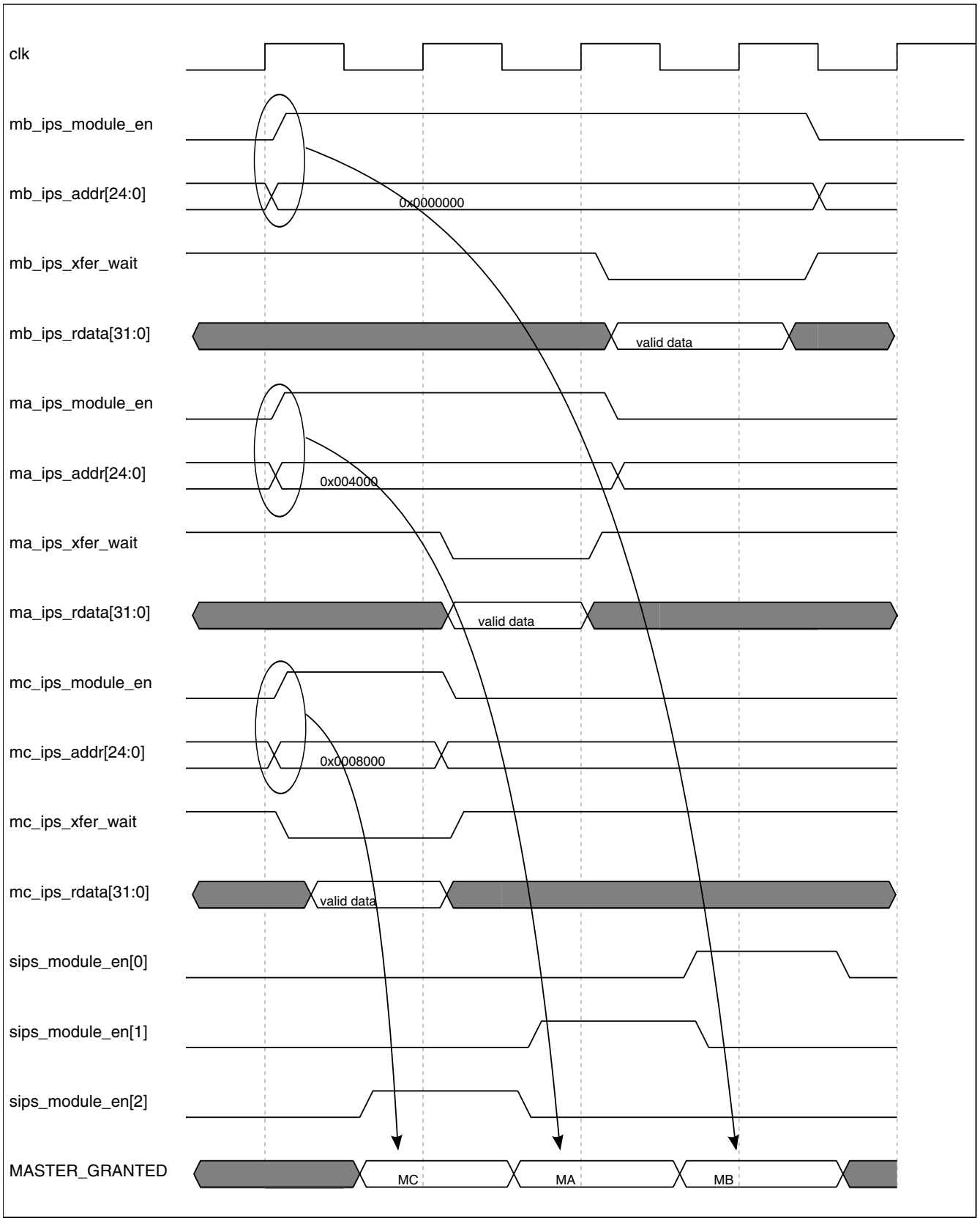


Figure 48-3. Example of three master requests: Masters already granted are "waited";

48.6 Resource Ownership Control

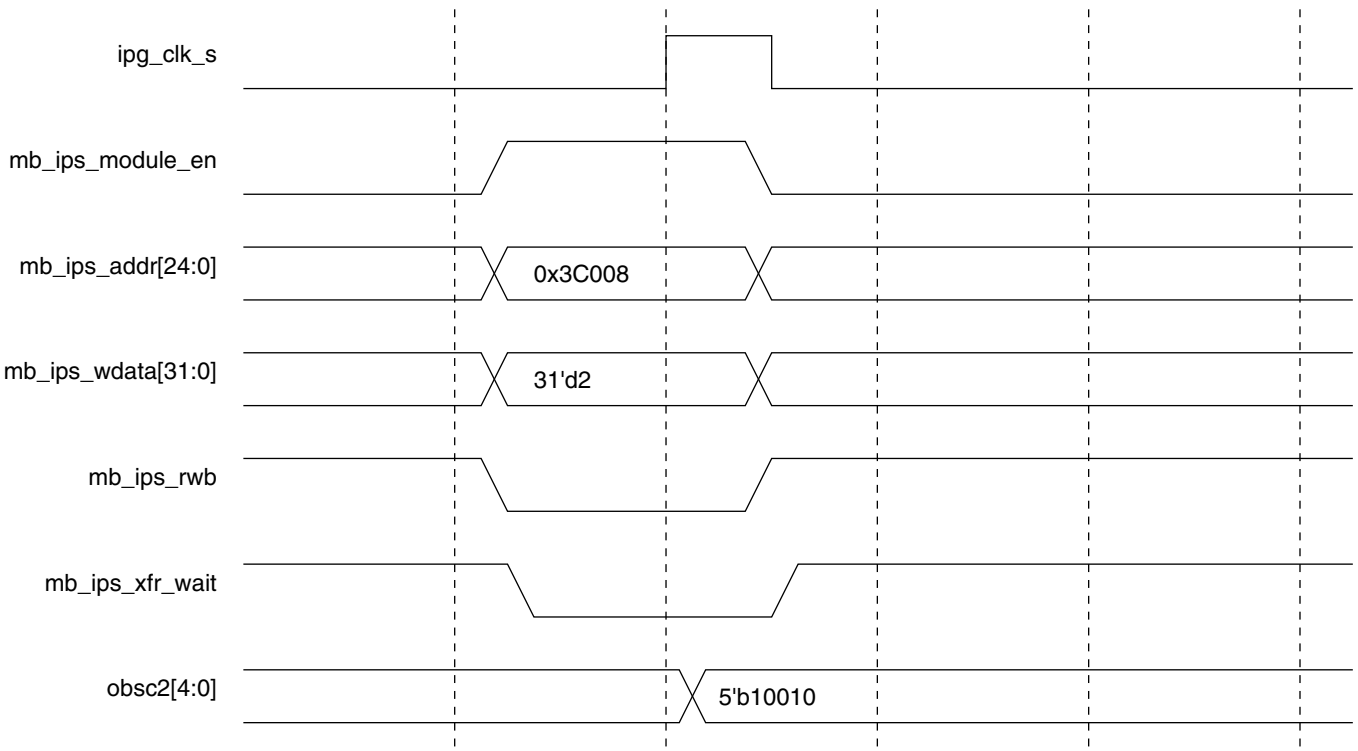
The Resource Ownership Control controls accesses to the shared peripherals and determines steering of out-of-band signals.

48.6.1 Access Control

48.6.1.1 Peripheral Access

The peripheral access (a.k.a resource access) of the requesting master is given by the corresponding RAR bit of the Peripheral Right Register. It determines if the master has access privilege to the resource.

Any attempted access to a resource by a requesting master whose access privilege bit is not set (in the PRR), is terminated with a bus error (**<master>_ips_xfr_err** is asserted by SPBA logic). The master which owns the resource can lock the peripheral for itself and/or grant other masters access to the peripheral by setting the appropriate bit(s) in the RAR field.



Master B is taking ownership of peripheral 2 by writing 3'b010 in the SPBA peripheral 2 right register (rarfield)
This ownership can be checked on obsc2 output as roi2[1:0] = 2'b10 and rar2[2:0] = 3'b010
(obsc[4:0] = {roi2[1], roi2[0], rar2[2], rar2[1], rar2[0]})

Figure 48-4. Example of one master B gaining ownership of peripheral 2

48.6.1.2 Peripheral Right Register Access

The ROI bits of the Peripheral Right Register (PRR) determine which master is allowed to make write access to PRR. The identification of the requesting master is compared to the ROI bits of the Peripheral Right Register to determine if the master has ownership of the corresponding register.

Any attempted write access to a Peripheral Right Register (PRR) already owned by another master will be ignored.

48.6.2 Owner Election

When the peripheral is not owned by any master (ROI="00", such as after coming out of reset), the first master to perform successfully a write to the RAR bits of the PRR is granted ownership of the peripheral and its associated PRR.

After writing to the PRR (RAR bit(s)), the master must read it back to make sure that it was granted ownership. If the RMO field is 2'b11, then the ownership claim is successful. If RMO is 2'b10, then another master claimed ownership before this master was able to complete its write. This resolves the case in which two or more masters attempt to write the PRR at the same time; only the first master will be granted ownership. However all masters must read the PRR to determine if this case occurred, and if so, whether they were the first master which was actually granted ownership.

NOTE

A master which has been granted ownership of the PRR does not automatically have the right access to the peripheral; it must still set its own RAR bits in the PRR to access the peripheral.

48.6.3 Ending Ownership

Ownership may be voluntarily ended by the owning master, or automatically upon assertion of a master-specific dead_owner signal.

The former is appropriate for software-controlled yielding of ownership. The latter is appropriate for automatic yielding of ownership when the owner has gone into reset. Hardware Controlled Ownership Ending

When a master is reset, it clears the ROI bits of the PRRs owned by the corresponding master. When the owner is dead (owner is in reset), all peripherals previously owned by that master must be changed to the un-owned state.

NOTE

It is the programmer's responsibility to make sure the peripherals are placed in an appropriate state before ending ownership.

48.6.3.1 Software Controlled Ownership Ending

The ROI bits will be automatically cleared when the master which owns the PRR access right clears (write) the RAR bits (Table 48-5).

It will then end the ownership of the PRR.

48.6.4 The Un-owned State

During the time when the peripheral is un-owned (i.e the ROI field contains all 0's), all masters have full access to it (RAR bits can then be modified by a master if ROI[1:0] = 2'b0).

In such cases it is necessary for software to ensure any necessary coherency in the resource, there is no hardware protection.

48.7 Memory Map/Register Definition

The following section describes the memory maps and detailed descriptions of all registers which are accessible to the end user within and through SPBA.

From a master side, the absolute address of one peripheral memory map registers is accessible by setting the <master>_ips_addr[24:0]:

- <master>_ips_addr[24:19] to the SPBA master base address (defined by top level parameters SPBA_<MASTER>_BASE_ADDR),
- <master>_ips_addr[18:14] to one of the 32 IP Bus peripherals (including SPBA). Refer to the table below,
- and <master>_ips_addr[13:0] to one of the 16-kilobyte memory mapped registers (16 Kilobytes only if accessing to a shared peripheral, not the same for SPBA PRR access).

The table below describes for each IP Bus peripheral, its absolute memory mapped address range accessible through the SPBA for one master (does not include the <master>_ips_addr[24:19] bits).

Table 48-1. Peripherals Absolute Address

Peripheral	<master>_ips_addr[18:0] start address	<master>_ips_addr[18:0] end address[1]
Peripheral 0	{5'b0_0000, 14'b00_0000_0000_0000}	{5'b0_0000, 14'b11_1111_1111_1111}
Peripheral 1	{5'b0_0001, 14'b00_0000_0000_0000}	{5'b0_0001, 14'b11_1111_1111_1111}

Table continues on the next page...

Table 48-1. Peripherals Absolute Address (continued)

Peripheral	<master>_ips_addr[18:0] start address	<master>_ips_addr[18:0] end address[1]
Peripheral 2	{5'b0_0010, 14'b00_0000_0000_0000}	{5'b0_0010, 14'b11_1111_1111_1111}
Peripheral 3	{5'b0_0011, 14'b00_0000_0000_0000}	{5'b0_0011, 14'b11_1111_1111_1111}
Peripheral 4	{5'b0_0100, 14'b00_0000_0000_0000}	{5'b0_0100, 14'b11_1111_1111_1111}
Peripheral 5	{5'b0_0101, 14'b00_0000_0000_0000}	{5'b0_0101, 14'b11_1111_1111_1111}
Peripheral 6	{5'b0_0110, 14'b00_0000_0000_0000}	{5'b0_0110, 14'b11_1111_1111_1111}
Peripheral 7	{5'b0_0111, 14'b00_0000_0000_0000}	{5'b0_0111, 14'b11_1111_1111_1111}
Peripheral 8	{5'b0_1000, 14'b00_0000_0000_0000}	{5'b0_1000, 14'b11_1111_1111_1111}
Peripheral 9	{5'b0_1001, 14'b00_0000_0000_0000}	{5'b0_1001, 14'b11_1111_1111_1111}
Peripheral 10	{5'b0_1010, 14'b00_0000_0000_0000}	{5'b0_1010, 14'b11_1111_1111_1111}
Peripheral 11	{5'b0_1011, 14'b00_0000_0000_0000}	{5'b0_1011, 14'b11_1111_1111_1111}
Peripheral 12	{5'b0_1100, 14'b00_0000_0000_0000}	{5'b0_1100, 14'b11_1111_1111_1111}
Peripheral 13	{5'b0_1101, 14'b00_0000_0000_0000}	{5'b0_1101, 14'b11_1111_1111_1111}
Peripheral 14	{5'b0_1110, 14'b00_0000_0000_0000}	{5'b0_1110, 14'b11_1111_1111_1111}
Peripheral 15	{5'b0_1111, 14'b00_0000_0000_0000}	{5'b0_1111, 14'b11_1111_1111_1111}
Peripheral 16	{5'b1_0000, 14'b00_0000_0000_0000}	{5'b1_0000, 14'b11_1111_1111_1111}
Peripheral 17	{5'b1_0001, 14'b00_0000_0000_0000}	{5'b1_0001, 14'b11_1111_1111_1111}
Peripheral 18	{5'b1_0010, 14'b00_0000_0000_0000}	{5'b1_0010, 14'b11_1111_1111_1111}
Peripheral 19	{5'b1_0011, 14'b00_0000_0000_0000}	{5'b1_0011, 14'b11_1111_1111_1111}
Peripheral 20	{5'b1_0100, 14'b00_0000_0000_0000}	{5'b1_0100, 14'b11_1111_1111_1111}
Peripheral 21	{5'b1_0101, 14'b00_0000_0000_0000}	{5'b1_0101, 14'b11_1111_1111_1111}
Peripheral 22	{5'b1_0110, 14'b00_0000_0000_0000}	{5'b1_0110, 14'b11_1111_1111_1111}
Peripheral 23	{5'b1_0111, 14'b00_0000_0000_0000}	{5'b1_0111, 14'b11_1111_1111_1111}
Peripheral 24	{5'b1_1000, 14'b00_0000_0000_0000}	{5'b1_1000, 14'b11_1111_1111_1111}
Peripheral 25	{5'b1_1001, 14'b00_0000_0000_0000}	{5'b1_1001, 14'b11_1111_1111_1111}
Peripheral 26	{5'b1_1010, 14'b00_0000_0000_0000}	{5'b1_1010, 14'b11_1111_1111_1111}
Peripheral 27	{5'b1_1011, 14'b00_0000_0000_0000}	{5'b1_1011, 14'b11_1111_1111_1111}
Peripheral 28	{5'b1_1100, 14'b00_0000_0000_0000}	{5'b1_1100, 14'b11_1111_1111_1111}
Peripheral 29	{5'b1_1101, 14'b00_0000_0000_0000}	{5'b1_1101, 14'b11_1111_1111_1111}
Peripheral 30	{5'b1_1110, 14'b00_0000_0000_0000}	{5'b1_1110, 14'b11_1111_1111_1111}
Peripheral 31	{5'b1_1111, 14'b00_0000_0000_0000}	{5'b1_1111, 14'b11_1111_1111_1111}

1. One of these peripherals must be defined as the SPBA. The end address range given is not valid for SPBA internal registers. If <master>_ips_addr[13:0] is greater than the last PRR location, the SPBA will return a transfer error to the master (report to Table 48-2).

48.7.1 SPBA Register Definition

The SPBA control registers (Peripheral Right Registers) are mapped as a virtual shared peripheral using by default the sips_module_en[15] (or another one using verilog parameter during integration phase) which must not be used for shared blocks.

SPBA can support up to 31 shared peripherals. Each of them has its own Peripheral Right Register (PRR) accessible within the SPBA memory mapped registers, and consists of the Requesting Master Owner, the Resource Owner ID and the Resource Access Right fields (using verilog parameters it can be defined which peripheral is present or not, and so if the corresponding PRR exists or not).

Table 48-2. SPBA PRR Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRRn[1] address = ($BASE[2] + n*4)	R	RMOn														ROIn	
	W																
	R														RARn		
	W																

1. for n=0 to 31
2. $BASE = {6'bSPBA_<MASTER>_BASE_ADDR, 5'bx_xxxx, 14'b00_0000_0000_0000}, where 5'bx_xxxx are <master>_ips_addr[18:14] reserved for SPBA registers access. The following table details the value of <master>_ips_addr[13:0] for master access any of the shared PRR

Table 48-3. PRR addresses

Peripheral Rights Registers (SPBA control registers)	<master>_ips_addr[13:0][1]	Note
Peripheral 0 Rights Registers	14'b00_0000_0000_0000	if MODULE0_PRESENT = 1'b1 & MODULE0_IS_SPBA = 1'b0
Peripheral 1 Rights Registers	14'b00_0000_0000_0100	if MODULE1_PRESENT = 1'b1 & MODULE1_IS_SPBA = 1'b0
Peripheral 2 Rights Registers	14'b00_0000_0000_1000	if MODULE2_PRESENT = 1'b1 & MODULE2_IS_SPBA = 1'b0
Peripheral 3 Rights Registers	14'b00_0000_0000_1100	if MODULE3_PRESENT = 1'b1 & MODULE3_IS_SPBA = 1'b0
Peripheral 4 Rights Registers	14'b00_0000_0001_0000	if MODULE4_PRESENT = 1'b1 & MODULE4_IS_SPBA = 1'b0
Peripheral 5 Rights Registers	14'b00_0000_0001_0100	if MODULE5_PRESENT = 1'b1 & MODULE5_IS_SPBA = 1'b0
Peripheral 6 Rights Registers	14'b00_0000_0001_1000	if MODULE6_PRESENT = 1'b1 & MODULE6_IS_SPBA = 1'b0
Peripheral 7 Rights Registers	14'b00_0000_0001_1100	if MODULE7_PRESENT = 1'b1 & MODULE7_IS_SPBA = 1'b0

Table continues on the next page...

Table 48-3. PRR addresses (continued)

Peripheral Rights Registers (SPBA control registers)	<master>_ips_addr[13:0][1]	Note
Peripheral 8 Rights Registers	14'b00_0000_0010_0000	if MODULE8_PRESENT = 1'b1 & MODULE8_IS_SPBA = 1'b0
Peripheral 9 Rights Registers	14'b00_0000_0010_0100	if MODULE9_PRESENT = 1'b1 & MODULE9_IS_SPBA = 1'b0
Peripheral 10 Rights Registers	14'b00_0000_0010_1000	if MODULE10_PRESENT = 1'b1 & MODULE10_IS_SPBA = 1'b0
Peripheral 11 Rights Registers	14'b00_0000_0010_1100	if MODULE11_PRESENT = 1'b1 & MODULE11_IS_SPBA = 1'b0
Peripheral 12 Rights Registers	14'b00_0000_0011_0000	if MODULE12_PRESENT = 1'b1 & MODULE12_IS_SPBA = 1'b0
Peripheral 13 Rights Registers	14'b00_0000_0011_0100	if MODULE13_PRESENT = 1'b1 & MODULE13_IS_SPBA = 1'b0
Peripheral 14 Rights Registers	14'b00_0000_0011_1000	if MODULE14_PRESENT = 1'b1 & MODULE14_IS_SPBA = 1'b0
Peripheral 15 Rights Registers	14'b00_0000_0011_1100	if MODULE15_PRESENT = 1'b1 & MODULE15_IS_SPBA = 1'b0
Peripheral 16 Rights Registers	14'b00_0000_0100_0000	if MODULE16_PRESENT = 1'b1 & MODULE16_IS_SPBA = 1'b0
Peripheral 17 Rights Registers	14'b00_0000_0100_0100	if MODULE17_PRESENT = 1'b1 & MODULE17_IS_SPBA = 1'b0
Peripheral 18 Rights Registers	14'b00_0000_0100_1000	if MODULE18_PRESENT = 1'b1 & MODULE18_IS_SPBA = 1'b0
Peripheral 19 Rights Registers	14'b00_0000_0100_1100	if MODULE19_PRESENT = 1'b1 & MODULE19_IS_SPBA = 1'b0
Peripheral 20 Rights Registers	14'b00_0000_0101_0000	if MODULE20_PRESENT = 1'b1 & MODULE20_IS_SPBA = 1'b0
Peripheral 21 Rights Registers	14'b00_0000_0101_0100	if MODULE21_PRESENT = 1'b1 & MODULE21_IS_SPBA = 1'b0
Peripheral 22 Rights Registers	14'b00_0000_0101_1000	if MODULE22_PRESENT = 1'b1 & MODULE22_IS_SPBA = 1'b0
Peripheral 23 Rights Registers	14'b00_0000_0101_1100	if MODULE23_PRESENT = 1'b1 & MODULE23_IS_SPBA = 1'b0
Peripheral 24 Rights Registers	14'b00_0000_0110_0000	if MODULE24_PRESENT = 1'b1 & MODULE23_IS_SPBA = 1'b0
Peripheral 25 Rights Registers	14'b00_0000_0110_0100	if MODULE25_PRESENT = 1'b1 & MODULE24_IS_SPBA = 1'b0
Peripheral 26 Rights Registers	14'b00_0000_0110_1000	if MODULE26_PRESENT = 1'b1 & MODULE26_IS_SPBA = 1'b0
Peripheral 27 Rights Registers	14'b00_0000_0110_1100	if MODULE27_PRESENT = 1'b1 & MODULE27_IS_SPBA = 1'b0
Peripheral 28 Rights Registers	14'b00_0000_0111_0000	if MODULE28_PRESENT = 1'b1 & MODULE28_IS_SPBA = 1'b0

Table continues on the next page...

Table 48-3. PRR addresses (continued)

Peripheral Rights Registers (SPBA control registers)	<master>_ips_addr[13:0][1]	Note
Peripheral 29 Rights Registers	14'b00_0000_0111_0100	if MODULE29_PRESENT = 1'b1 & MODULE29_IS_SPBA = 1'b0
Peripheral 30 Rights Registers	14'b00_0000_0111_1000	if MODULE30_PRESENT = 1'b1 & MODULE30_IS_SPBA = 1'b0
Peripheral 31 Rights Registers	14'b00_0000_0111_1100	if MODULE31_PRESENT = 1'b1 & MODULE31_IS_SPBA = 1'b0

1. Any accesses to SPBA inexistent location will generate a <master>_ips_xfr_err

48.7.1.1 Peripheral Right Register (SPBA_PRRn)

Peripheral Right Register Diagram (SPBA_PRRn)

SPBA_PRRn[1]				Peripheral Right Register n									Addr $BASE[2] + n*4 Wait State: 0 Access: -			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RMOn														ROIn	
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
														RARn		
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Note:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. for n=0 to 31
2. $BASE = {6'bSPBA_MX_BASE_ADDR, 5'bx_xxxx, 14'b00_0000_0000_0000}, where 5'bx_xxxx are the bit [18:14] of master address bus reserved for SPBA access

RMOn[1:0] - Requesting Master Owner

This 2-bit register field indicates if the corresponding resource is owned by the requesting master or not. This register is reset to 2'b0 if ROI[1:0] = 2'b0.

Table 48-4. RMOn Values

Value	Meaning
00	resource un-owned
10	resource owned by another master
11	resource owned by requesting master

ROIn[1:0] - Resource Owner ID

This 2-bits register field indicates which master (one at a time) can access to the PRR for rights modification. This is a Read-Only register.

Table 48-5. ROIn Values

Value	Meaning
00	un-owned resource
01	resource owned by master A port
10	resource owned by master B port
11	resource owned by master C port

After reset, ROI bits are cleared ("00" -> un-owned resource).

A master performing a write access to the an un-owned PRR will get its ID automatically written into ROI, while modifying RAR bits. It can then read back the RMO, RAR, ROI bits to make sure RMO returns the right value, ROI bits contain its ID and RAR bits are correctly asserted. Then no other master (whom ID is different from the one stored in ROI) will be able to modify RAR fields.

Owner master of a peripheral can assert its dead_owner signal, or write 3'b0 in the RAR to release the ownership (ROI[1:0] reset to 2'b0).

RARn[2:0] - Resource Access Right

This 3-bit register field indicates which master can access to the peripheral. From 0 up to 3 masters can have permission to access a resource (all the master can be granted on a peripheral, but only one access at a time will be granted by SPBA).

RARn[0] - Control and Status bit for master A which is connected to port MA,

access to peripheral is granted.

access to peripheral is not allowed.

RARn[1] - Control and Status bit for master B which is connected to port MB,

access to peripheral is granted.

access to peripheral is not allowed.

RARn[2] - Control and Status bit for master C which is connected to port MC,

access to peripheral is granted.

access to peripheral is not allowed.

After reset all RAR registers are set to value 1. So by default all masters have access granted to all the shared peripherals. This is verified until one master modify this default value.

Chapter 49
System Reset Controller (SRC)

49.1 Introduction

The System Reset Controller (SRC) generates the reset signals for all IPs in the i.MX50 IC. The SRC block diagram is shown in Figure 49-1.

49.1.1 Overview

The reset control is responsible for the generation of all reset signals and boot decoding. The reset control determines the source and the type of reset, such as POR, WARM, and COLD, and performs the necessary reset qualification and stretching sequences. The reset logic generates the reset sequence for the entire IC based on the type of reset. Whenever the chip is powered on, the reset is issued through ipp_reset_b signal and the entire chip is reset.

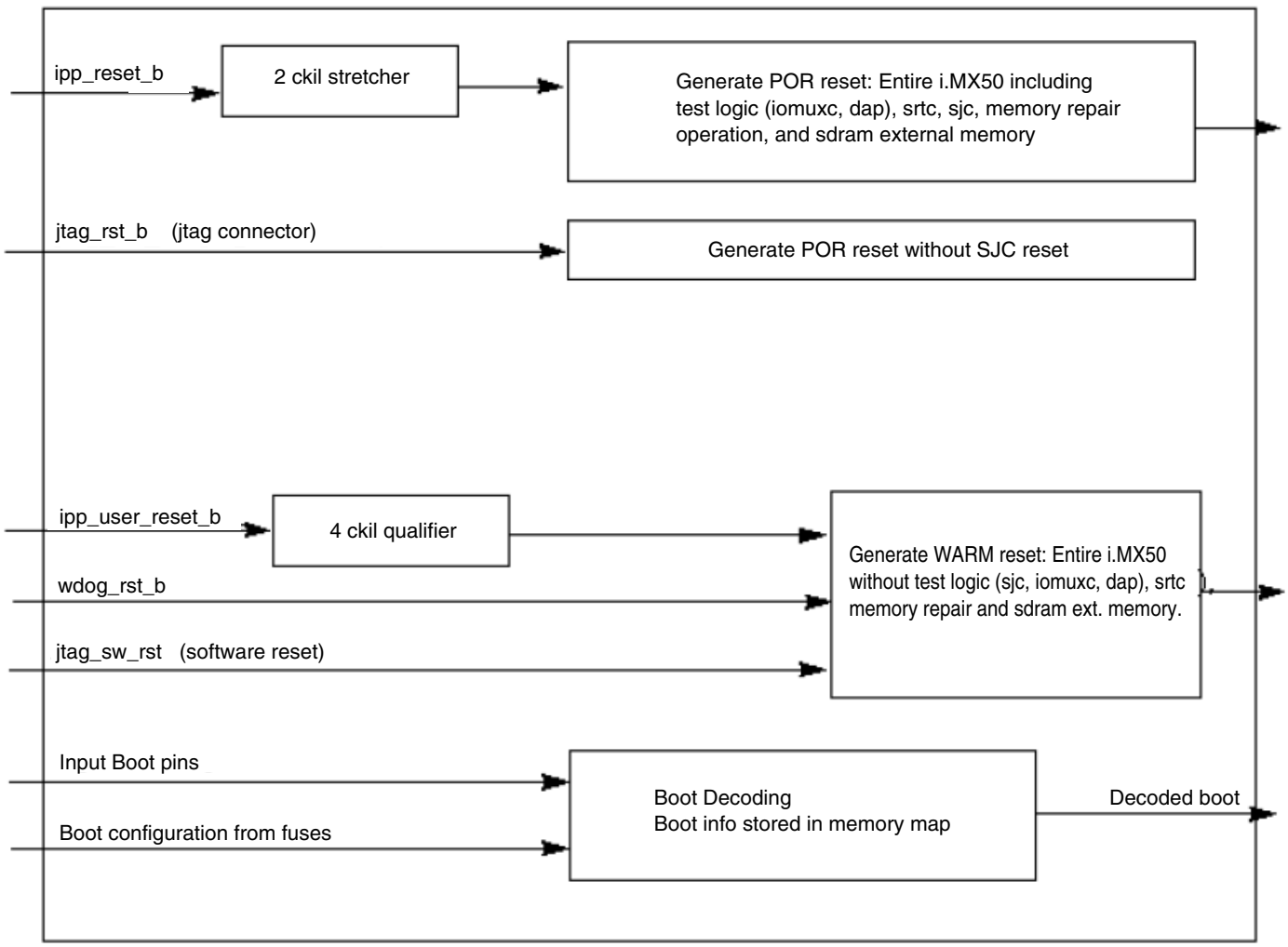


Figure 49-1. SRC High Level Diagram

49.1.2 Features

The SRC includes the following features:

- Receives and handles the resets from all reset sources
- Resets the appropriate domains based on the reset sources and the nature of the reset
- Latches the BOOT_MODE pins and common configuration signals from the internal fuse
- Includes a 32-bit IP bus interface

49.2 Functional Description

49.2.1 Reset Control

This section details the reset control of this device.

49.2.1.1 Reset Inputs and Outputs

The reset control logic receives reset requests from all potential reset sources. All the immediate sources of reset are directly passed to the reset stretching block. The resets which require qualification are passed on to the reset qualification logic before they are sent to the reset stretching block.

All reset inputs and outputs are described in Figure 49-2.

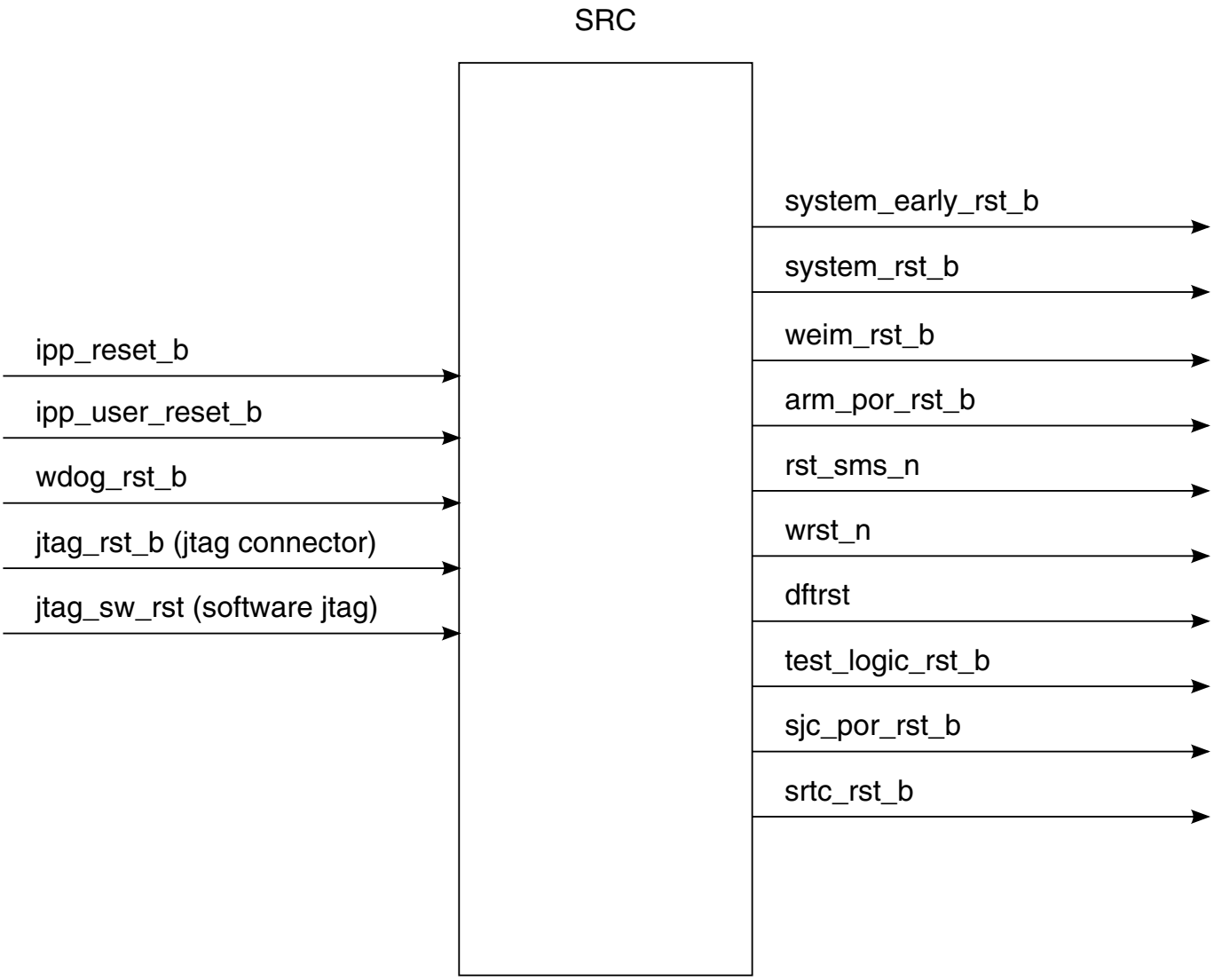


Figure 49-2. SRC Inputs and Outputs

The reset sources, type and behavior are shown in the Table 49-1. Because there is no chip POR, the ipp_reset_b is used to reset the entire chip including test logic and JTAG modules.

NOTE

All resets are expected to be active low except jtag_sw_rst.

Table 49-1. Reset Priorities, Sources, Types, and Behavior

Priority	Sources	Type	Behavior
5	ipp_reset_b	POR	Reset IC = system_early_rst_b + system_rst_b + arm_por_rst+ arm_soc_rst_b + arm_dbg_rst_b + open_vg_rst_b + rst_sms_n + wrst + dftrst + + srtc_rst_b + sjc_por_rst_b

Table continues on the next page...

Table 49-1. Reset Priorities, Sources, Types, and Behavior (continued)

Priority	Sources	Type	Behavior
6	ipp_user_reset_b (qualified 4 ckil's)	Warm	Reset IC - rst_sms_n - wrst - dftrst - test_logic_rst_b - srtc_rst_b- arm_por_rst - arm_dbg_rst_b - sjc_por_rst_b + warm_reset_signal = system_early_rst_b + system_rst_b + arm_soc_rst_b + open_vg_rst_b + warm_reset_signal
7	wdog_rst_b This reset will not be generated if mask_wdog_rst bits are coded to 5.	Warm	Reset IC - rst_sms_n - wrst - dftrst - test_logic_rst_b - srtc_rst_b - arm_por_rst - arm_dbg_rst_b - sjc_por_rst_b + warm_reset_signal = system_early_rst_b + system_rst_b + arm_soc_rst_b + open_vg_rst_b + warm_reset_signal
8	jtag_rst_b (jtag connector)	POR - sjc_por_rst_b	Reset IC - sjc_por_rst_b = system_early_rst_b + system_rst_b + arm_por_rst+ arm_soc_rst_b + arm_dbg_rst_b + open_vg_rst_b + rst_sms_n + wrst + dftrst + srtc_rst_b
9	jtag_sw_rst (software jtag)	Warm	Reset IC - rst_sms_n - wrst - dftrst - test_logic_rst_b - srtc_rst_b - arm_por_rst - arm_dbg_rst_b + warm_reset_signal - sjc_por_rst_b = system_early_rst_b + system_rst_b + arm_rst_b + open_vg_rst_b + warm_reset_signal

The reset priorities are POR (strongest), COLD and WARM (weakest). If a stronger reset is asserted during the sequence of a weaker reset, the weaker sequence will be interfered and the stronger reset sequence will commence. There is no priority inside a reset type group. If a reset is asserted during the sequence of a reset from the same type group, the reset sequence will be interfered and a new reset sequence (from the same type) will commence.

The following table shows the functionality of each of these reset outputs:

Table 49-2. Reset Output Functionality

Reset Output	Functionality
system_early_rst_b	Resets the system modules that need to start first as ccm, pll_ip's, iim, fuseboxes, etc.
system_rst_b	Resets functional modules
arm_rst_b	Resets ARM module (on regular system reset)
open_vg_rst_b	Resets Open_VG module (on regular system reset)
arm_por_rst	Resets ARM por input
arm_soc_rst_b	Reset for arm SOC
arm_dbg_rst_b	Reset debug logic of ARM
test_logic_rst_b	Reset test logic (iomuxc, dap)
sjc_por_rst_b	Reset to SJC
srtc_rst_b	Resets SRTC

The following table shows the reset outputs related to SMS (STAR memory system):

Table 49-3. SMS Reset Output Functionality

SMS Reset Output	Functionality
$\overline{\text{rst\_sms\_n}}$	Resets SMS
dftrst	Resets SMS DFT
$\overline{\text{wrst}}$	Resets JPC wrapper

SRC will also generate rst_sms_n dftrst and wrst_n for specific accelerators. Because those modules are not involved in the memory repair process, they will be similar to the functional resets of those modules. Note that dftrst is active high and its inverted version equals the functional reset.

NOTE

It is assumed that each reset source will deassert after its assertion, either due to a reset generated to the system from SRC, or due to negation of the reset source (in case it came from an external source to the IC). In the later case, it is assumed the reset source is held for at least two ckil cycles so it can be sampled by SRC.

49.2.1.2 Reset Handling

49.2.1.2.1 Reset Qualification

The reset qualification logic qualifies the reset source before sending it out to the chip as a valid reset. All reset sources, except for Warm resets, are immediate resets and are acknowledged by the reset circuitry the moment they are asserted. Warm resets must be qualified before they are sent as valid resets.

A reset source is updated in the reset status register when it becomes valid, provided it is asserted for the minimum amount of time after asserting. All immediate resets are immediately updated in the status register.

Once the reset is qualified, internal resets are asserted appropriately according to the source of the reset.

49.2.1.2.2 Reset Sequence and De-Assertion

The system_early_rst_b and the system_rst_b will assert immediately after any reset source is recognized. After all of the reset sources are released, the SRC will start the following set of events depending on the type of reset that occurred.

49.2.1.2.2.1 IPP_RESET_B(POR)

This is an external signal from the pads. Whenever the chip is powered up, the reset signal is passed through this pin indicating power-up sequence. SRC resets the entire chip including test logic and JTAG module. All SRC registers will be reset on POR sequence as follows:

1. As soon as IPP_RESET_B occurs, all resets are asserted and the entire chip is reset by SRC module. The IPP_RESET_B is stretched for two CKIL cycles, which takes place after two CKIL clocks of IPP_RESET_B pin deassertion.
2. test_logic_rst_b, sjc_por_rst_b and srtc_rst_b signals are deasserted together with IPP_RESET_B signal. Those outputs are also deasserted after the stretching of IPP_RESET_B has deasserted.
3. After the above resets deassert, system_early_rst_b reset is deasserted after two CKIL clocks. The deassertion of system_early_rst_b is used by CCM and DPLLCs to start generating PLL and root clock outputs.
4. Once system root clocks are ready, CCM will assert system_clk_ready signal. This signal is generated during the ignition sequence in CCM and involves the preparation of CKIH/PLLs to generate clock roots for functional operation.
5. Once this signal arives at SRC, it initiates smart sfp operation. This operation will be finished once the SRC receives the assertion on signal "sfp_ready". In parallel with the sfp operation, enables are asserted by the SRC to allow power up clocks into the SoC.
6. After SFP operation is finished, SRC enables IIM and fusebox clocks so that fuses can be loaded to IIM. IIM will notify with iim_ready_flag on completion of the fusebox loading.
7. SRC prepares boot information.
8. SRC waits eight ipg clock cycles.
9. SRC waits eight CKIL clocks.
10. SRC enables GPU2D clocks so that reset will penetrate the module.
11. After eight ipg cycles, SRC dissables open_vg clocks.
12. After eight ipg cycles, resets to all modules are deasserted (system_rst_b, open_vg_rst_b).
13. After eight ipg cycles, enable functional clocks are enabled.
14. If en_tester_control signal is asserted, SRC waits for assertion of tester_ack signal to continue. This signal allows tester to receive determination of reset sequence.
15. After eight ipg cycles, system clocks are enabled (en_system_clk).

NOTE

The memory repair operation is bypassed when the smart_bypass signal is asserted.

49.2.1.2.2.2 COLD RESET

The cold reset sequence is similar to $\overline{\text{IPP\_RESET\_B}}$, except the memory repair operation is not performed and test_logic_reset is not fired.

1. Once the reset source deasserts, system_early_rst_b reset is deasserted after at least two CKIL clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating pll clock outputs and system root clocks.
2. Once system root clocks are ready, CCM asserts system_clk_ready signal. This signal is generated during the ignition sequence in CCM. It involves the preparation of CKIH/PLLs to generate clock roots for functional operation. See Ignition Sequence in the CCM chapter.
3. Once the signal arives at SRC, SRC enables IIM and fusebox clocks so that fuses can be loaded to IIM. IIM will notify with iim_ready_flag on completion of the fusebox loading.
4. SRC prepares the boot information and then deasserts emi_rst_b.
5. SRC waits eight ipg clock cycles, then enables DRAM MC clocks.
6. SRC waits eight CKIL clocks.
7. SRC enables GPU2D clocks so that reset will penetrate the module.
8. After eight ipg cycles, SRC dissables open_vg clocks.
9. After eight ipg cycles, resets to all modules are deasserted (system_rst_b, open_vg_rst_b,).
10. After eight ipg cycles, enable functional clocks is enabled.
11. If en_tester_control signal is asserted, SRC waits for assertion of tester_ack signal to continue. This signal allows tester to receive determinism of the reset sequence.
12. After eight ipg cycles, system clocks are enabled (en_system_clk).

49.2.1.2.2.3 WARM RESET

The handshake between the SRC and the DRAM MC controller for warm reset in the i.MX50 has been disabled. The acknowledge from the DRAM MC controller has been tied off so the warm reset will immediately be acknowledged, thus behaving like a warm reset scenerio. The SRC will not wait for the DRAM MC to be placed into self-refresh.

Warm reset will be enabled only if warm_reset_enable bit is programed. Otherwise, all warm reset sources will generate a cold reset. This bit will be reset only by por reset.

One of the sources of the warm reset is ipp_user_reset_b. If this is the case, it is qualified for four CKIL edges. The warm reset is initiated immediately after four CKIL edges and the system does not come out of reset until the the ipp_user_reset_b is released.

In cases where a warm reset is triggered but has not occurred, a cold reset will be generated after a number of CKIL clocks. The number of CKIL clocks is defined in bit field warm_rst_bypass_count in the SCR register. The default for this bit is 16 CKIL count.

The following is a basic description of the warm reset sequence:

1. ARM sets warm_reset_enable bit to enable the warm reset functionality. If this bit is not set, all warm reset sources will result in cold reset.
2. One of the warm reset sources is asserted.
3. The reset source is qualified in the SRC.
4. SRC asserts system resets
5. The deassertion sequence is exactly the same as in the cold reset, except DRAM MC does not wait for eight CKILs to generate fixed external clock to external memory SDRAM. This stage is not needed in warm reset because SDRAM is held in self-refresh and there is no need to reconfigure it when going out of warm reset.

49.2.1.2.2.4 WARM BOOT

Software saves any needed information in the memory before initiating a warm reset. In this case, software will set the warm_boot bit before initiating warm reset. After the system returns to run mode, the warm_boot bit will still be set, indicating to the software that data was saved in memory and can be reused.

Figure 49-3, Figure 49-4, Figure 49-5, and Figure 49-6 shows the behavior of different resets for POR, Cold and Warm reset cases respectively.

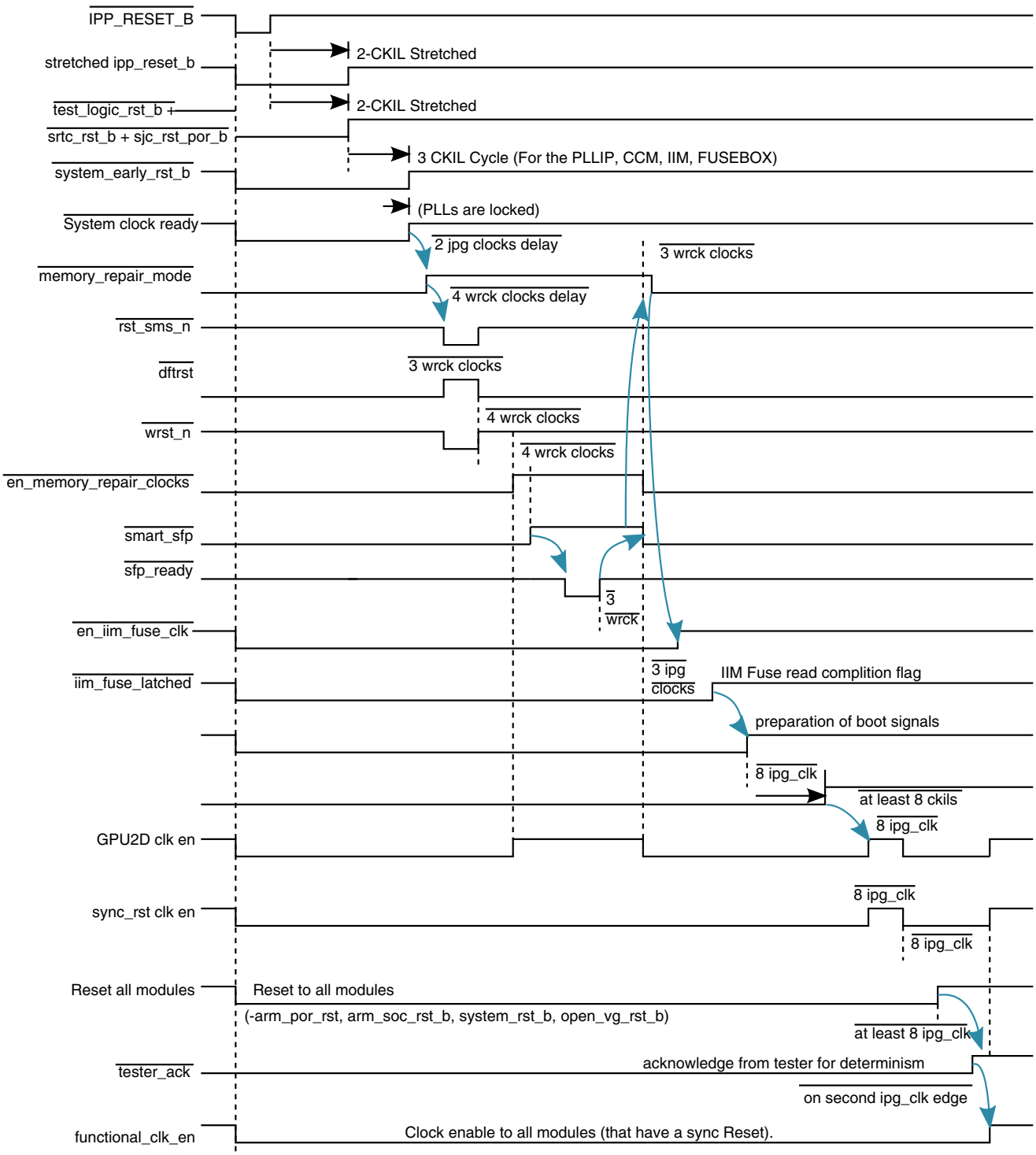


Figure 49-3. Behavior of IPP_RESET_B(POR)

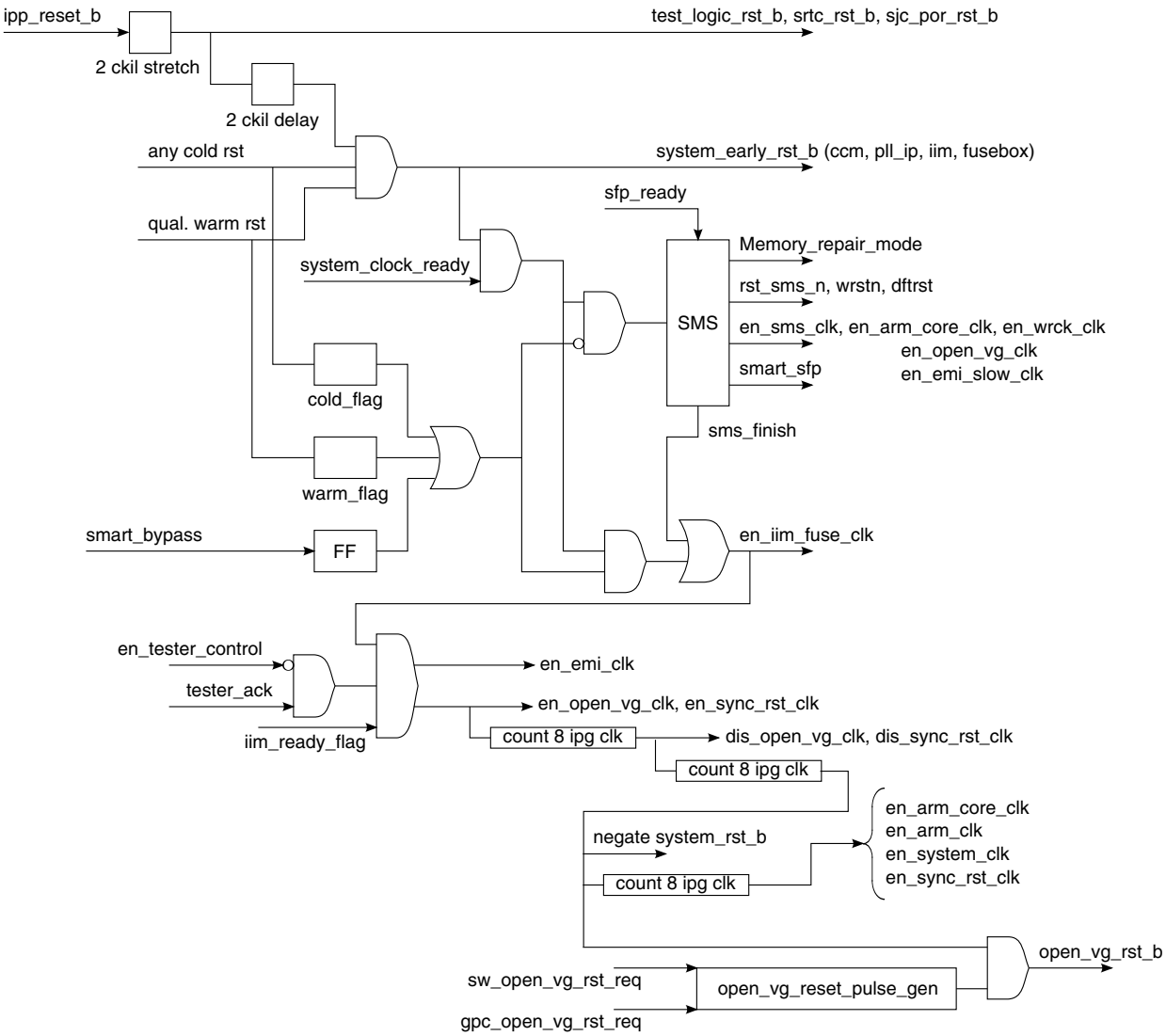


Figure 49-4. Reset Diagram

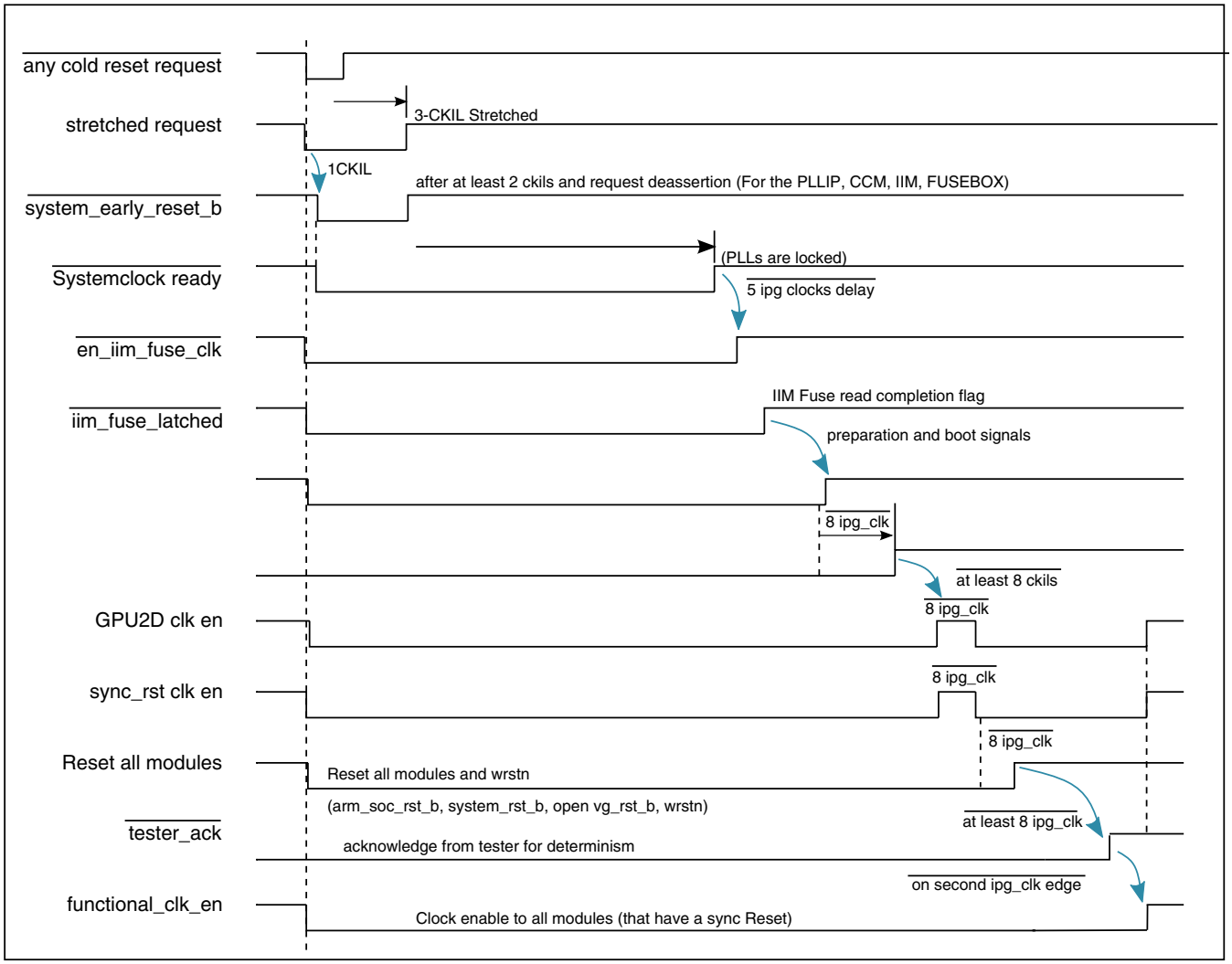


Figure 49-5. Cold Reset Sequence

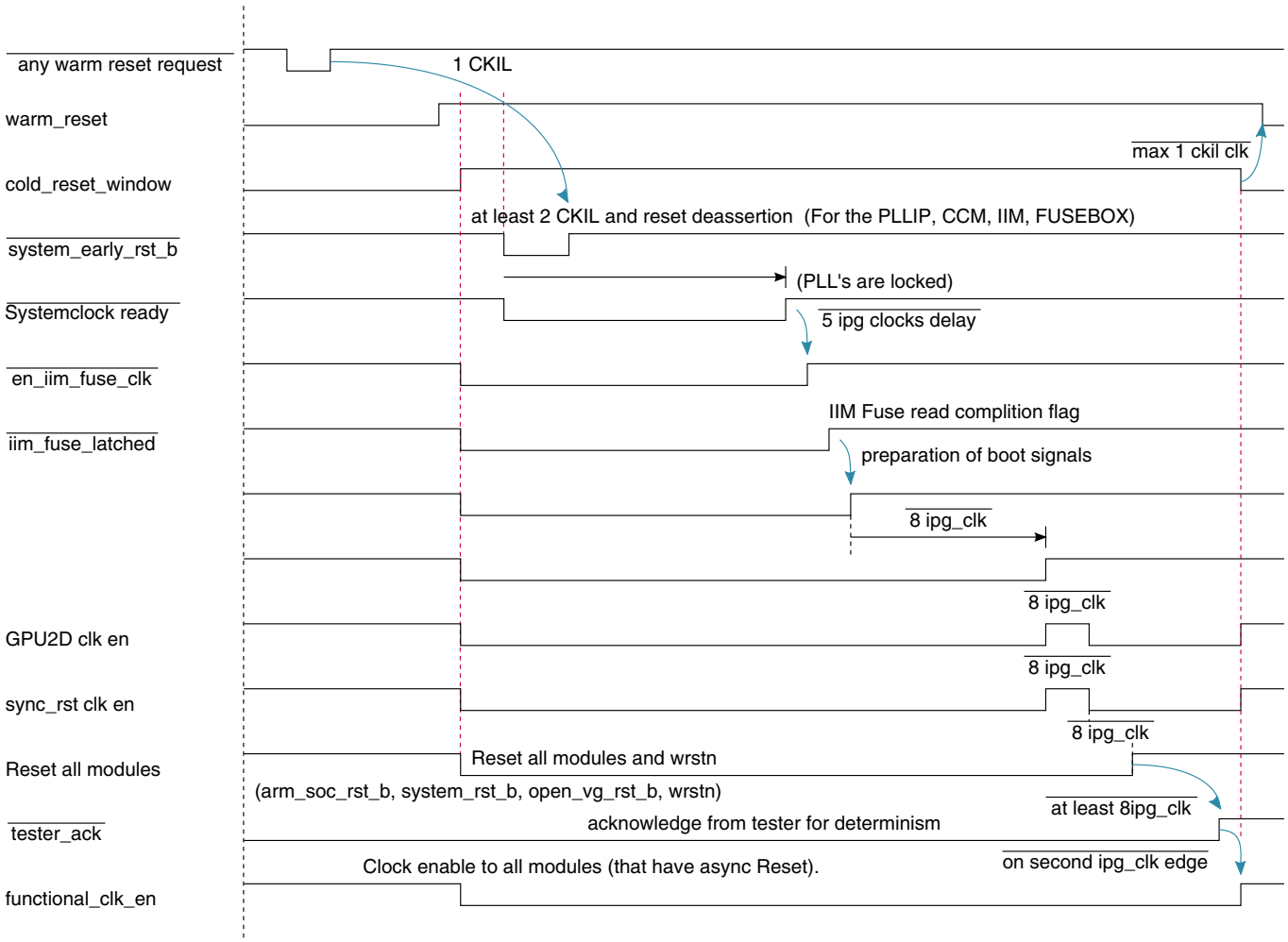


Figure 49-6. Warm Reset Sequence

49.2.1.2.3 SMS Sub Module

The sms sub module manages the memory repair signals during por. The sms sub module will start its operation after system_clock_ready notification from CCM and only if the reset sequence is due to por sequence (ipp_reset_b). The sms sub module operation can be bypassed by smart_bypass assertion signal from the pad. If the smart_bypass is asserted, only the assertion part of sms reset signals will take place, but smart_sfp - sfp_ready sequence will not be performed.

The SMS sequence is as follows:

1. Upon assertion of system_clock_ready signal, count 2 ipg clocks and assert the memory_repair_mode signal.
2. Wait 4 wrck clocks.
3. Assert sms reset signals on wrck clock edge:
   - rst_sms_n = 0

- wrst_n = 0
- dftrst = 1

4. Wait 3 wrck clocks
5. De assert the reset signals on wrck clock edge:
   - rst_sms_n = 1
   - wrst_n = 1
   - dftrst = 0
6. Wait 4 wrck clocks
7. Enable memory repair clocks, i.e. en_arm_core_clk=1, en_emi_slow_clk=1, en_open_vg_clk=1, en_sms_clk=1 and en_wrck_clk=1 on wrck edge.
8. Wait 4 wrck clocks
9. Assert smart_sfp signal on wrck edge.
10. Monitor de assertion and assertion of sfp_ready signal.
11. Once assertion of sfp_ready signal, de assert smart_sfp, de assert clock enables en_arm_core_clk=0 , en_emi_slow_clk=0, en_open_vg_clk=0, en_sms_clk=0 and en_wrck_clk=0 on wrck edge, de assert memory_repair_mode signal. (all on wrck edges).
12. Wait 4 wrck clocks
13. Assert sms_finish to notify that SRC can continue with enable IIM clocks.

Figure 49-7 describes the SMS flow.

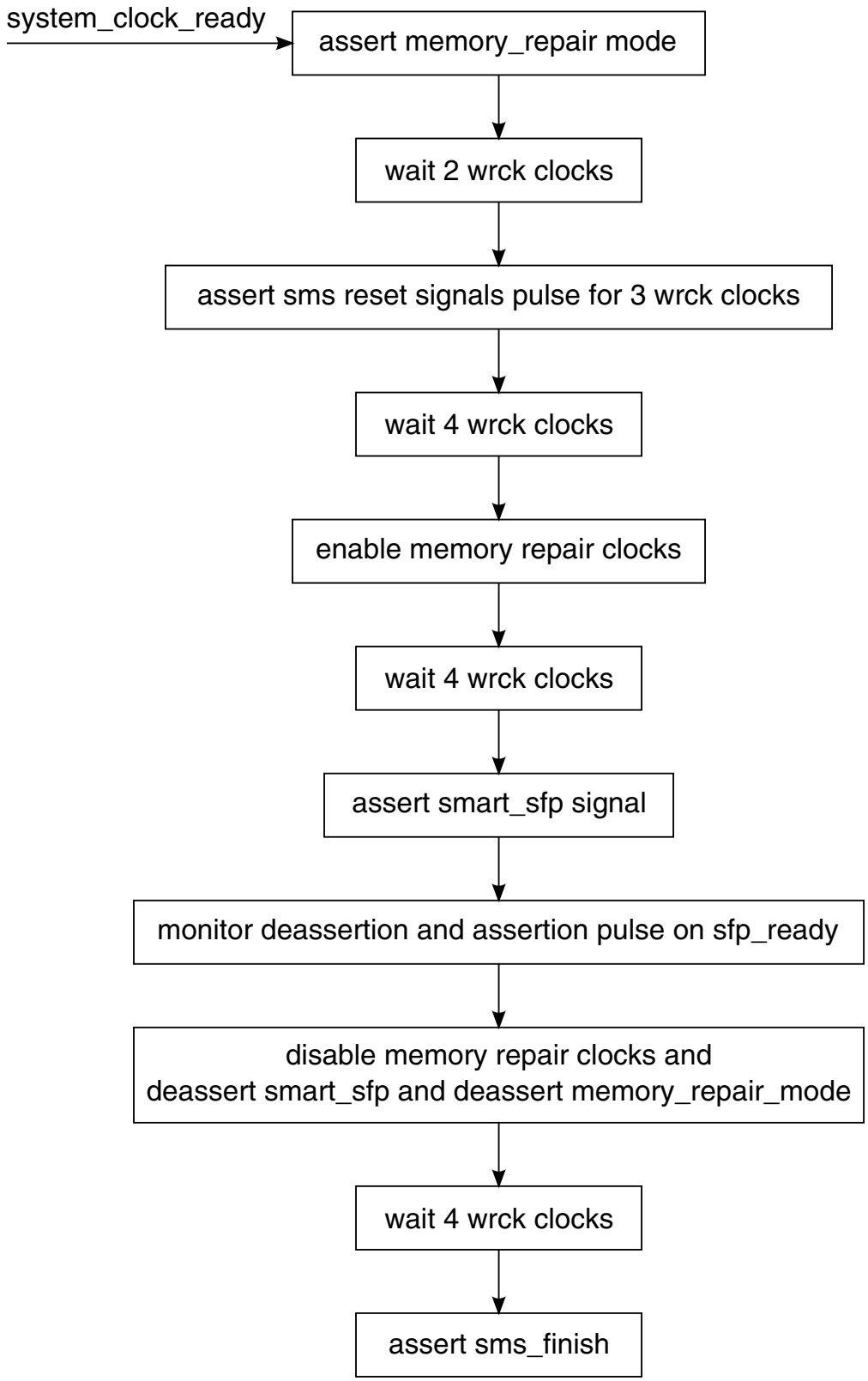


Figure 49-7. SMS Flow Diagram

49.2.1.2.4 SRTC_RST_B generation

SRTC_RST_B will be generated during POR sequence together with test_logic_rst_b.

On the srtc_rst_b path there will be a mux that enables a generation of reset on srtc_rst_b pad from TCU , in case of test_mode. The enable of this mux is test_mode input. When the test_mode input equals '0', the generated srtc_rst_b will be the functional srtc reset (generated during POR scenario). When the test_mode input equals '1', the generated srtc_rst_b will be connected to the tcu_reset input. In this case, tcu will have complete controlability of srtc_rst_b , and will be able to generate a reset for srtc. Figure 49-8 describes this mux.

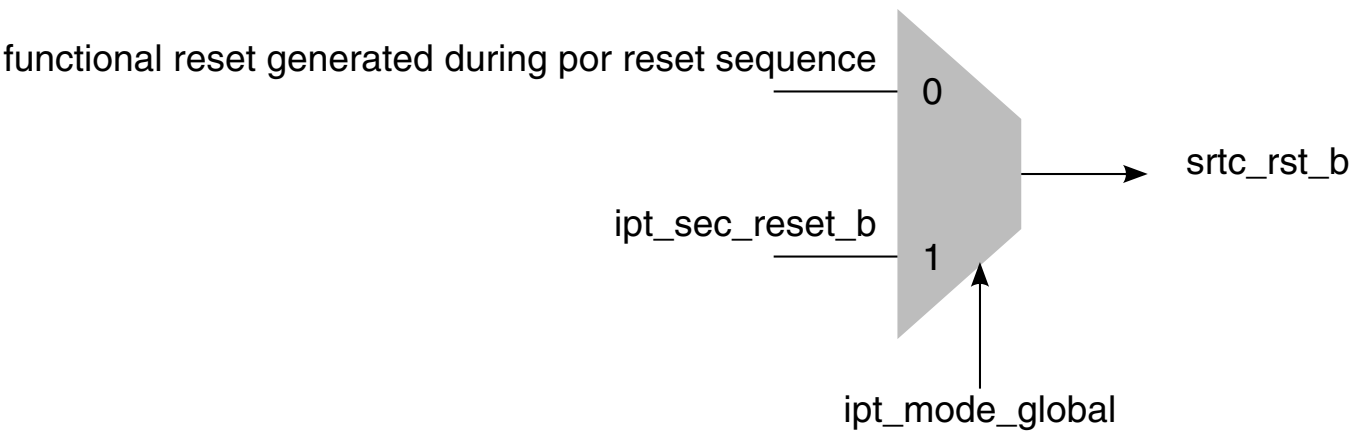


Figure 49-8. srtc_rst_b Generation

49.2.2 SJC_POR_RST_B Generation

SJC will be reseted by sjc_por_rst_b output of SRC. This output will be toggled only on POR sequence together with test_logic_rst_b. It will not be toggled on IEEE reset (jtag_rst_b).

49.2.3 Parallel Reset Requests

This chapter describes the behavior when paralel reset requests are received. In the paralel reset requests case SRC will follow the following rules:

1. The order of strength of resets is POR - strongest, cold - medium, warm - weakest.

2. If a stronger reset is asserted during weaker reset sequence, then the stronger reset will take over and the stronger reset process will comence. the following cases fall into this category:
   - POR reset request in the middle of cold or warm reset process-the cold or warm reset process will be stopped and the POR sequence will start.
   - COLD reset request in the middle of warm reset process-the warm reset process will be stoped and the cold sequence will start.
3. If a weaker reset is asserted during stronger reset sequence, then the stronger reset sequence will continue without interference. If at the end of the stronger reset process the weaker request is still asserted then the weaker sequence will commence. The following cases fall into this category:
   - COLD or WARM reset requests in the middle of POR reset process-the POR process will continue without interference.
   - WARM reset request in the middle of COLD reset process-the COLD process will continue without interference.
4. If a similar reset request is asserted during the process of reset handling, then the process of reset handling will start over (with the same process). The following cases fall into this category:
   - POR reset request in the middle of POR reset process-the POR process will start over.
   - COLD reset request in the middle of COLD reset process-the COLD process will start over.

   There is one exception to this category-WARM reset request inthe middle of WARM reset process-in this case the new WARM reset process cant restart because DRAM MC is reseted and there can't be a handshake with DRAM MC if DRAM MC is reseted. For this case the first WARM reset will continue, and only if the second WARM reset is still asserted after the first one has finished, then the WARM sequence will start again.

49.2.4 DFT Mux on Reset Outputs

All functional reset output described above are muxed before exiting SRC. The mux is controlled by "ipt_mode_global" signal. The mux chooses between the functional reset and the DFT reset.

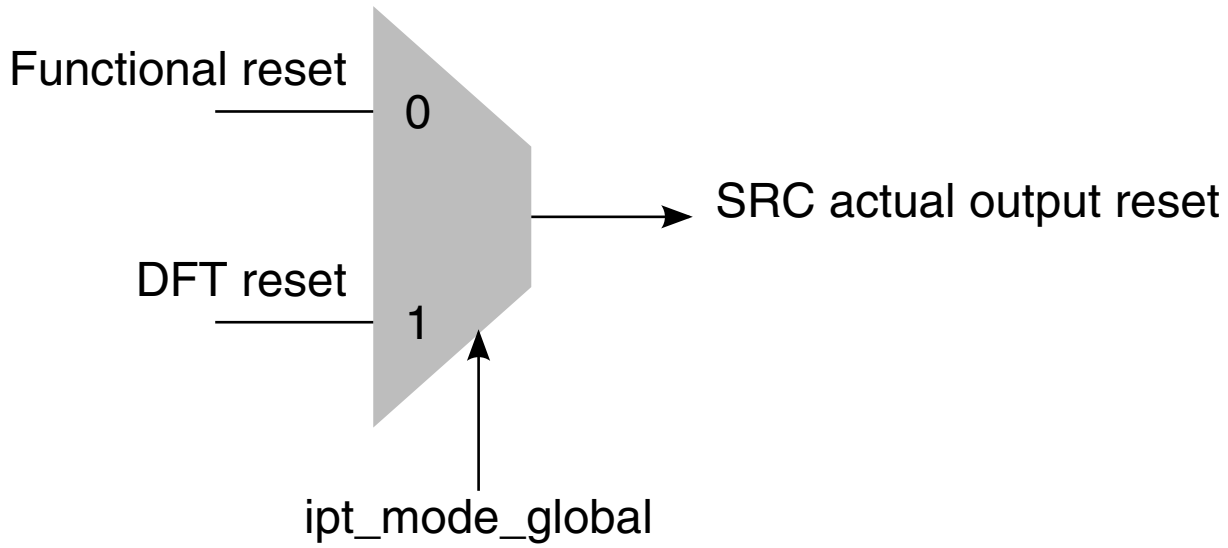


Figure 49-9. MUX between Functional Reset and DFT Reset

Table 49-4. DFT Reset Assigment

SRC Reset Output	DFT Reset Generation
arm_por_rst	~(ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b)
arm_soc_rst_b	ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b
dftrst	~(ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b)
rst_sms_n	ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b
system_early_rst_b	ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b
system_rst_b	ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b
test_logic_rst_b	ipp_reset_b
wrst_n	ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b
wrst_n_open_vg	ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b
srtc_rst_b	**ipt_sec_reset_b**
src_por_b	ipp_reset_b
weim_rst_b	ipp_reset_b & ipp_user_reset_b & ipt_sec_reset_b

NOTE

src_por_b is an internal reset generated for SRC module only. This reset is asserted on por only (ipp_reset_b). It is equal to the functional test_logic_reset_b.

49.2.5 Boot Mode Control

49.2.5.1 BOOT_MODE Pin Latching

The exact boot sequence for the i.MX50 is controlled by the values of the BOOT_MODE pins on this device.

All the boot pins will be sampled at deassertion of system_early_rst_b.

The value of the BOOT_MODE pins will be latched after the IIM asserts the fuse read completion flag. After latching, the values of the BOOT_MODE pins are used to determine the booting options of the core as given in the SBMR register.

The Boot mode general purpose bits can be provided to the SRC from either e-fuses or GPIO signals. The gpio_bt_sel e-fuse defines the source to be used to derive the boot information. When gpio_bt_sel is set, e-fuses are used. When cleared, GPIO signals are used.

When FSL test mode is chosen (01 for BOOT_MODE[1:0]) then the Fuses signals are used.

When Internal Production Boot Mode is chosen (10 for BOOT_MODE[1:0]) then e-fuses signals are used, independent of the gpio_bt_sel e-fuse value.

The Boot information is provided in SBMR register. Figure 49-10 shows the selection of Boot mode information.

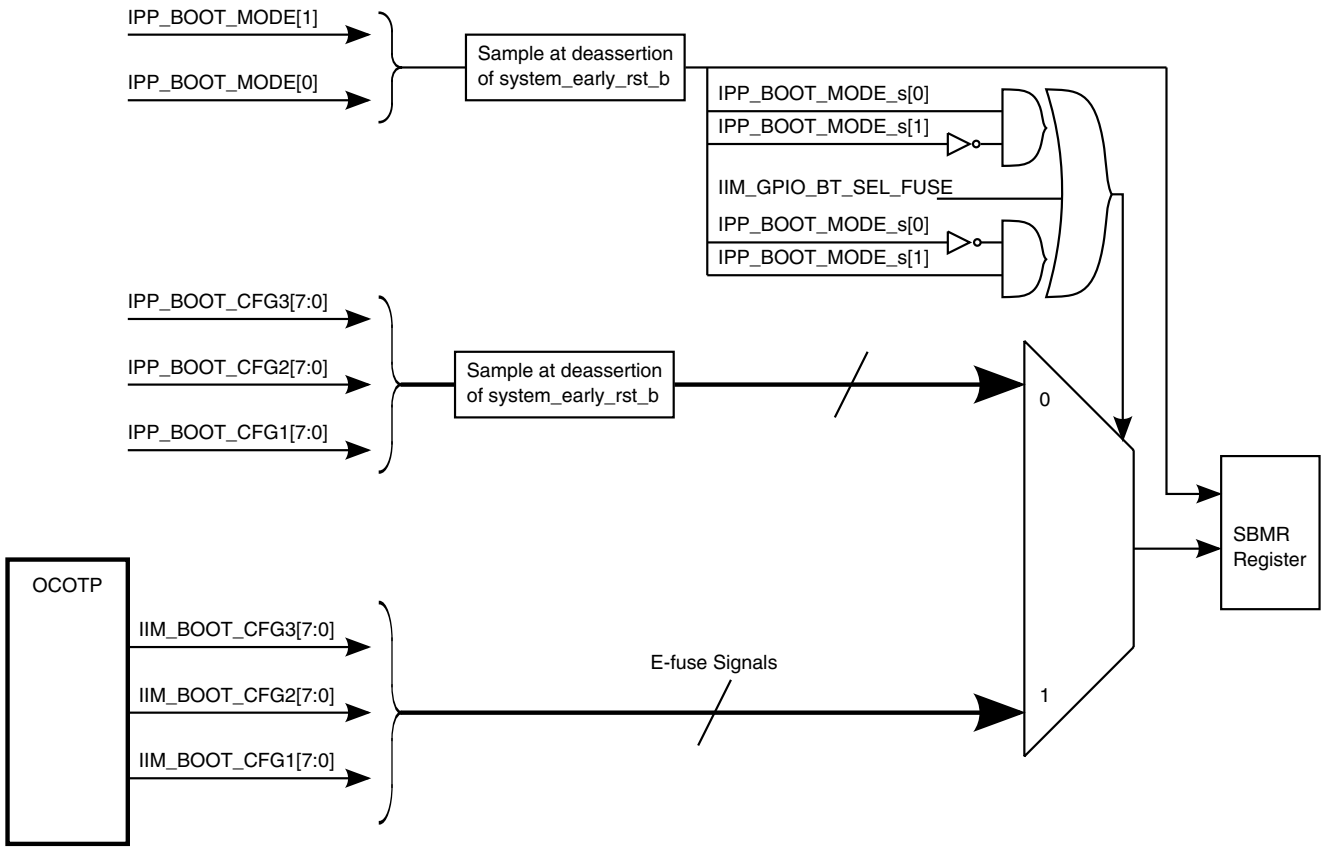


Figure 49-10. Boot Mode Information

NOTE

IPP signals that need to be latched by SRC will be set to the corresponding mode based on system_rst_b signal. When system_rst_b signal equals '0', the IOMUX will shift to the needed mode for transfering boot values on the IO pins. When system_rst_b signal equals '1', the IOMUX will shift to the default mode.

49.2.5.2 Correspondence between SRC SBMR Bit/Signal Names and Fuse Names

The SBMR bit names and boot signals corresponding to those bits were changed in i.MX50 fuse map. Table 49-5 describes the matching between the new fuse names and the ones represented in SBMR register:

Table 49-5. SBMR Fuse/Pad Names

SBMR Bit(s)	Choose Fuses/Pads	SRC Internal Signal	Fuse/Pad Names
31	1	iim_bt_reserved_fuse[1]	BT_LPB_FREQ[1]
	0	ipp_bt_reserved_fuse[1]	SSI_TXFS Pad
30	1	iim_bt_reserved_fuse[0]	BT_LPB_FREQ[0]
	0	ipp_bt_reserved_fuse[0]	SSI_TXC Pad
29	1	test_mode_s[2]	EIM_DA7 Pad
	0		
28	1	test_mode_s[1]	EIM_DA6 Pad
	0		
27	1	test_mode_s[0]	EIM_DA5 Pad
	0		
26	1	iim_bt_virgin_fuse	BT_VIRGIN
	0		
25	1	ipp_boot_mode_s[1]	BOOT_MODE1 Pad
	0		
24	1	ipp_boot_mode_s[0]	BOOT_MODE0 Pad
	0		
23	1	iim_boot_cfg3_fuse[7]	NAND_ROW_ADDRESS_BTYES[1]
	0	ipp_boot_cfg3_s[7]	EIM_RDY Pad
22	1	iim_boot_cfg3_fuse[6]	NAND_ROW_ADDRESS_BTYES[0]
	0	ipp_boot_cfg3_s[6]	EIM_BCLK Pad
21	1	iim_boot_cfg3_fuse[5]	BOOT_SEARCH_COUNT[1]
	0	ipp_boot_cfg3_s[5]	EIM_WAIT Pad
20	1	iim_boot_cfg3_fuse[4]	BOOT_SEARCH_COUNT[0]
	0	ipp_boot_cfg3_s[4]	EIM_EB1 Pad
19	1	iim_boot_cfg3_fuse[3]	PAGES_IN_BLOCK[1]
	0	ipp_boot_cfg3_s[3]	EIM_EB0 Pad
18	1	iim_boot_cfg3_fuse[2]	PAGES_IN_BLOCK[0]
	0	ipp_boot_cfg3_s[2]	EIM_CS0 Pad
17	1	iim_boot_cfg3_fuse[1]	BT_USB_LEGACY
	0	ipp_boot_cfg3_s[1]	EIM_CS1 Pad
16	1	iim_boot_cfg3_fuse[0]	DIR_BT_DIS
	0		
15	1	iim_boot_cfg2_fuse[7]	TM33_PREAMBLE_READ_LATENCY[2]
	0	ipp_boot_cfg2_s[7]	EIM_DA15 Pad
14	1	iim_boot_cfg2_fuse[6]	TM33_PREAMBLE_READ_LATENCY[1]
	0	ipp_boot_cfg2_s[6]	EIM_DA14 Pad

Table continues on the next page...

Table 49-5. SBMR Fuse/Pad Names (continued)

SBMR Bit(s)	Choose Fuses/Pads	SRC Internal Signal	Fuse/Pad Names
13	1	iim_boot_cfg2_fuse[5]	TM33_PREAMBLE_READ_LATENCY[0]
	0	ipp_boot_cfg2_s[5]	EIM_DA13 Pad
12	1	iim_boot_cfg2_fuse[4]	AXI_DDR_FREQ
	0	ipp_boot_cfg2_s[4]	EIM_DA12 Pad
11	1	iim_boot_cfg2_fuse[3]	OSC_FREQ_SEL
	0	ipp_boot_cfg2_s[3]	EIM_DA11 Pad
10	1	iim_boot_cfg2_fuse[2]	BT_DISABLE_ROM_REDUNDANT_BOOT
	0	ipp_boot_cfg2_s[2]	EIM_DA10 Pad
9	1	iim_boot_cfg2_fuse[1]	SEC_CONFIG[1]
	0		
8	1	iim_boot_cfg2_fuse[0]	SEC_CONFIG[0]
	0		
7	1	iim_boot_cfg1_fuse[7]	not clear, value=1
	0	ipp_boot_cfg1_s[7]	EIM_DA7 Pad
6	1	iim_boot_cfg1_fuse[6]	GPMI_CLK_SRC_PLL1
	0	ipp_boot_cfg1_s[6]	EIM_DA6 Pad
5	1	iim_boot_cfg1_fuse[5]	BT_TOGGLEMODE
	0	ipp_boot_cfg1_s[5]	EIM_DA5 Pad
4	1	iim_boot_cfg1_fuse[4]	GPMI_MUX[1]
	0	ipp_boot_cfg1_s[4]	EIM_DA4 Pad
3	1	iim_boot_cfg1_fuse[3]	GPMI_MUX[0]
	0	ipp_boot_cfg1_s[3]	EIM_DA3 Pad
2	1	iim_boot_cfg1_fuse[2]	BT_DEVICE_1.8V
	0	ipp_boot_cfg1_s[2]	EIM_DA2 Pad
1	1	iim_boot_cfg1_fuse[1]	BT_FREQ
	0	ipp_boot_cfg1_s[1]	EIM_DA1 Pad
0	1	iim_boot_cfg1_fuse[0]	BT_MMU_ENABLE
	0	ipp_boot_cfg1_s[0]	EIM_DA0 Pad

49.2.5.3 Output Boot Signals

The output boot signal are as follows:

1. External BOOT is set when the chip is in FSL TEST MODE and BOOT FROM WEIM is selected. Otherwise, internal boot is selected.

2. WEIM_BOOT_CFG bits-Used to determine if the chip uses muxed weim 16 bits low half or unmuxed weim 16 bits high half data. See the i.MX50 Fuse Map for more details.

3. src_boot_add[31:0] -boot address-output to ARM Platform
- IF src_int_boot=0 then src_boot_add[31:0]=32'h F000_0000
- Otherwise src_boot_add[31:0]=32'h 0000_0000

4. ipp_do_boot_mode_inv[1:0] - inversion of IPP_BOOT_MODE[1:0]. Once the boot signals (ipp_boot_mode[1:0]) are sampled in SRC, SRC will generate the inversion of ipp_boot_mode[1:0] on those pins. The IO ring will use system_rst_b to generate the ipp_do_boot_mode_inv[1:0] on the boot pads:

When system_rst_b=0 , the boot pads will be configured as inputs, allowing boot info to propagate to SRC's inputs ipp_boot_mode[1:0].

When system_rst_b=1 , the boot pads will be configured as output, allowing ipp_do_boot_mode_inv[1:0] to be generated to the pads, i.e. it will be noticed on the pads that the value has flipped.

The board will catch this flip and in this way will be notified that the system sequence has changed, and it is allowed to use the boot pads for a different purpose.

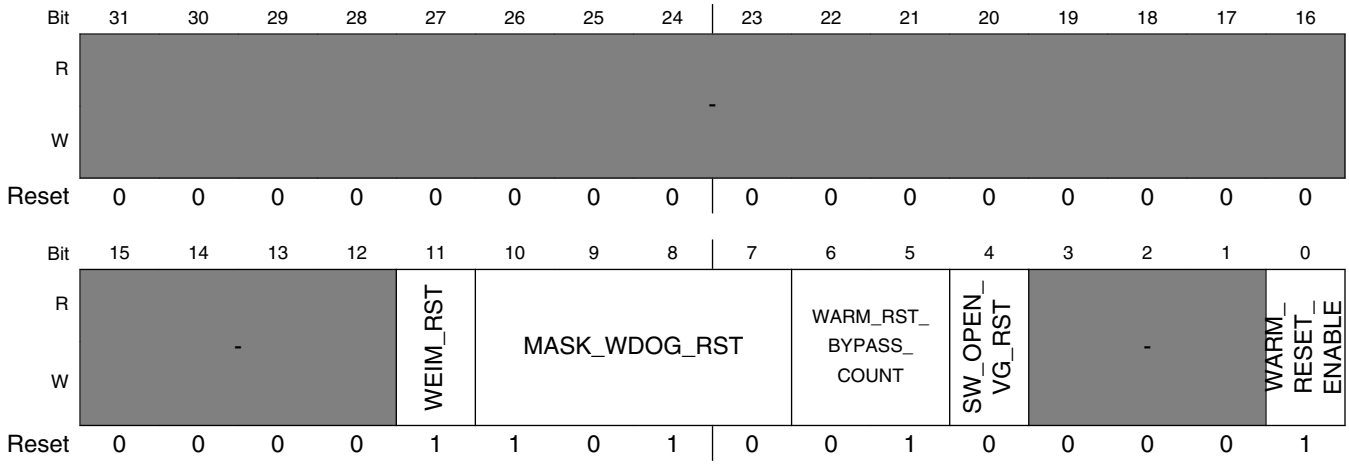
49.3 Programmable Registers

SRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FD_0000	SRC Control Register (SRC_SCR)	32	R/W	0000_0D21h	49.3.1/ 2846
53FD_0004	SRC Boot Mode Register (SRC_SBMR)	32	R	See section	49.3.2/ 2847
53FD_0008	SRC Reset Status Register (SRC_SRSR)	32	R/W	0000_0001h	49.3.3/ 2848
53FD_0014	SRC Interrupt Status Register (SRC_SISR)	32	R	0000_0000h	49.3.4/ 2850
53FD_0018	SRC Interrupt Mask Register (SRC_SIMR)	32	R/W	0000_000Fh	49.3.5/ 2851

49.3.1 SRC Control Register (SRC_SCR)

Address: SRC_SCR is 53FD_0000h base + 0h offset = 53FD_0000h



SRC_SCR field descriptions

Field	Description
31–12 -	Reserved
11 WEIM_RST	EIM reset is needed in order to reconfigure the eim chip select. The software reset bit must de-asserted. The eim chip select configuration should be updated. The software bit must be re-asserted since this is not self-refresh.
10–7 MASK_WDOG_ RST	Mask wdog_rst_b source. If these 4 bits are coded from A to 5 then, wdog_rst_b input to SRC will be masked and wdog_rst_b will not create reset to the IC. Note: During the time the WDOG event is masked using SRC logic, it is likely that WDOG Reset Status Register (WRSR) bit 1 (which indicates WDOG timeout event) will get asserted. SW / OS developer must prepare for this case. Re-enable of WDOG is possible, by un-masking it in SRC, though it must be preceded by the servicing of WDOG. However, if the event has been asserted, the status bit (WRSR bit-1) will remain asserted, regardless of the servicing the WDOG module. (HW reset is the only mean to cause de-assertion of that bit). 0101 wdog_rst_b is masked 1010 wdog_rst_b is not masked (default) Any other code will be coded to 1010 (that is, wdog_rst_b is not masked)
6–5 WARM_RST_ BYPASS_ COUNT	Reserved-EMI acknowledge is disabled for the SRC controller so warm reset will initiate immediately
4 SW_OPEN_VG_ RST	Software reset for open_vg Note: This is a self clearing bit. Once it is set to 1, the reset process will begin, and once it finishes, this bit will be self cleared. Software can determine that the reset has finished once this bit is cleared. Software can also configure SRC to generate interrut once the software has finished. See SISR register for details.

Table continues on the next page...

SRC_SCR field descriptions (continued)

Field	Description
	Note: The reset process will involve 8 open_vg cycles before negating the open_vg reset, allowing reset assertion to propagate into open_vg. 0 do not assert open_vg reset 1 assert open_vg reset
3–1 -	Reserved
0 WARM_RESET_ENABLE	Warm reset enable bit. Warm reset will be enabled only if warm_reset_enable bit is set, otherwise all warm reset sources will generate cold reset. 0 Warm reset disabled 1 Warm reset enabled

49.3.2 SRC Boot Mode Register (SRC_SBMR)

The Boot Mode register (SBMR) contains bits that reflect the status of boot mode pins of the chip. The default values for those bits depends on the values of pins/fuses during the reset sequence. Refer to Table 1 for a mapping of SBMR bits to fuse/pad names.

Address: SRC_SBMR is 53FD_0000h base + 4h offset = 53FD_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BT_ RESERVED		TEST_MODE[2:0]			BT_ FUSE_ SEL	BOOT_ MODE[1:0]		BOOT_CFG3[7:0]							
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BOOT_CFG2[7:0]								BOOT_CFG1[7:0]							
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

SRC_SBMR field descriptions

Field	Description
31–30 Reserved	This field is reserved. Please refer to i.MX50 fuse map.
29–27 TEST_MODE[2:0]	Please refer to i.MX50 fuse map.

Table continues on the next page...

SRC_SBMR field descriptions (continued)

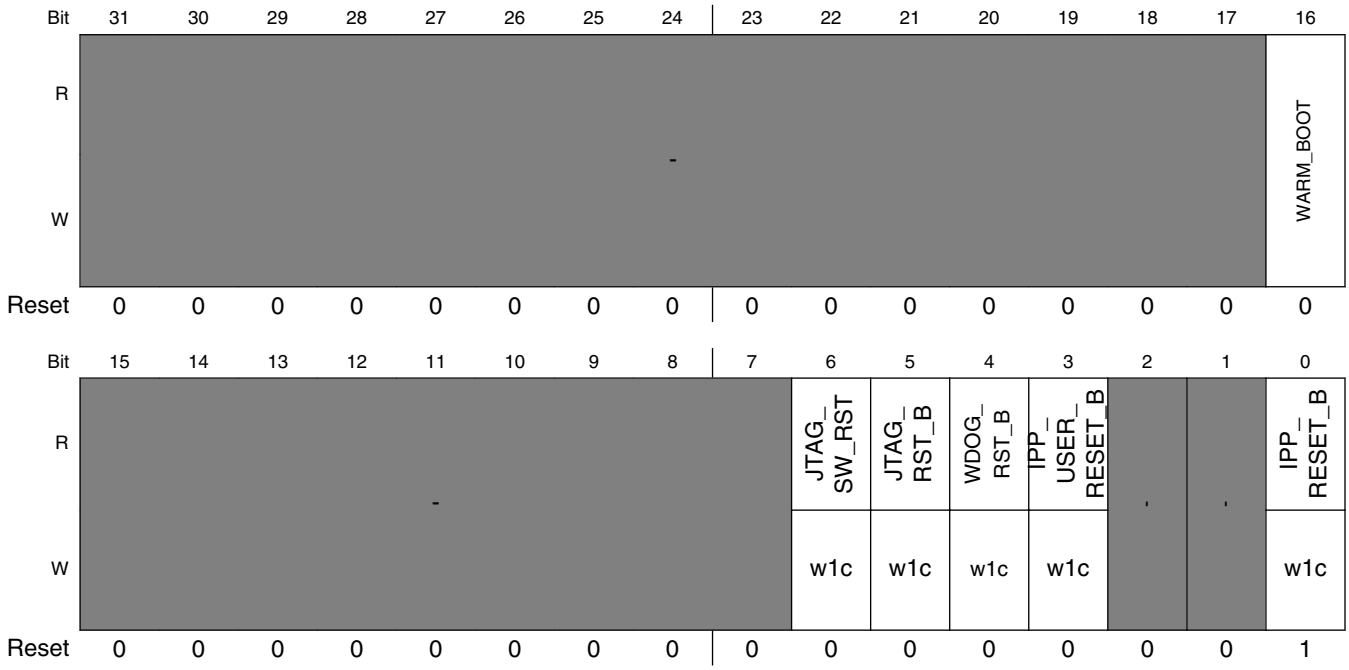
Field	Description
26 BT_FUSE_SEL	Please refer to i.MX50 fuse map.
25–24 BOOT_ MODE[1:0]	Please refer to i.MX50 fuse map.
23–16 BOOT_ CFG3[7:0]	Please refer to i.MX50 fuse map.
15–8 BOOT_ CFG2[7:0]	Please refer to i.MX50 fuse map.
7–0 BOOT_ CFG1[7:0]	Please refer to i.MX50 fuse map.

49.3.3 SRC Reset Status Register (SRC_SRSR)

The SRSR is a write one to clear register which records the source of the reset events for the chip. The SRSR will capture all the reset sources that have occurred. This register is reset on ipp_reset_b and is a read-write register.

For bit fields 6-0, writing zero has no effect. The individual bits can be cleared by writing one to each bit. When the system comes out of reset, this register will have bits set corresponding to all the reset sources that occurred during system reset. Software must clear this register by writing one after every reset that occurs in order for the register to contain the information from the most recent reset.

Address: SRC_SRSR is 53FD_0000h base + 8h offset = 53FD_0008h



SRC_SRSR field descriptions

Field	Description
31–17 -	Reserved
16 WARM_BOOT	Warm boot indication gives software the capability to notify that warm boot was initiated by software. This indicates to the software that it saved the needed info in the memory before initiating the warm reset. In this case, software will set this bit to '1', before initiating the warm reset. The warm_boot bit should be used as indication only after a warm_reset sequence. Software should clear this bit after warm_reset to indicate that the next warm_reset is not done with warm_boot. Please refer to Reset Sequence and De-Assertion for details on warm_reset. 0 warm boot process not initiated by software. 1 warm boot initiated by software.
15–7 -	Reserved
6 JTAG_SW_RST	JTAG SW reset. Indicates whether or not the reset was the result of software reset from JTAG. Connections at chip-level: jtag_sw_rst -> sjc_gpccr_reg_31 0 Reset is not a result of software reset from JTAG. 1 Reset is a result of software reset from JTAG.
5 JTAG_RST_B	HIGH - Z JTAG reset. Indicates whether or not the reset was the result of HIGH-Z reset reset from JTAG. Connections at chip-level: jtag_rst_b -> sjc_ieee_reset_b 0 Reset is not a result of HIGH-Z reset from JTAG. 1 Reset is a result of HIGH-Z reset from JTAG.

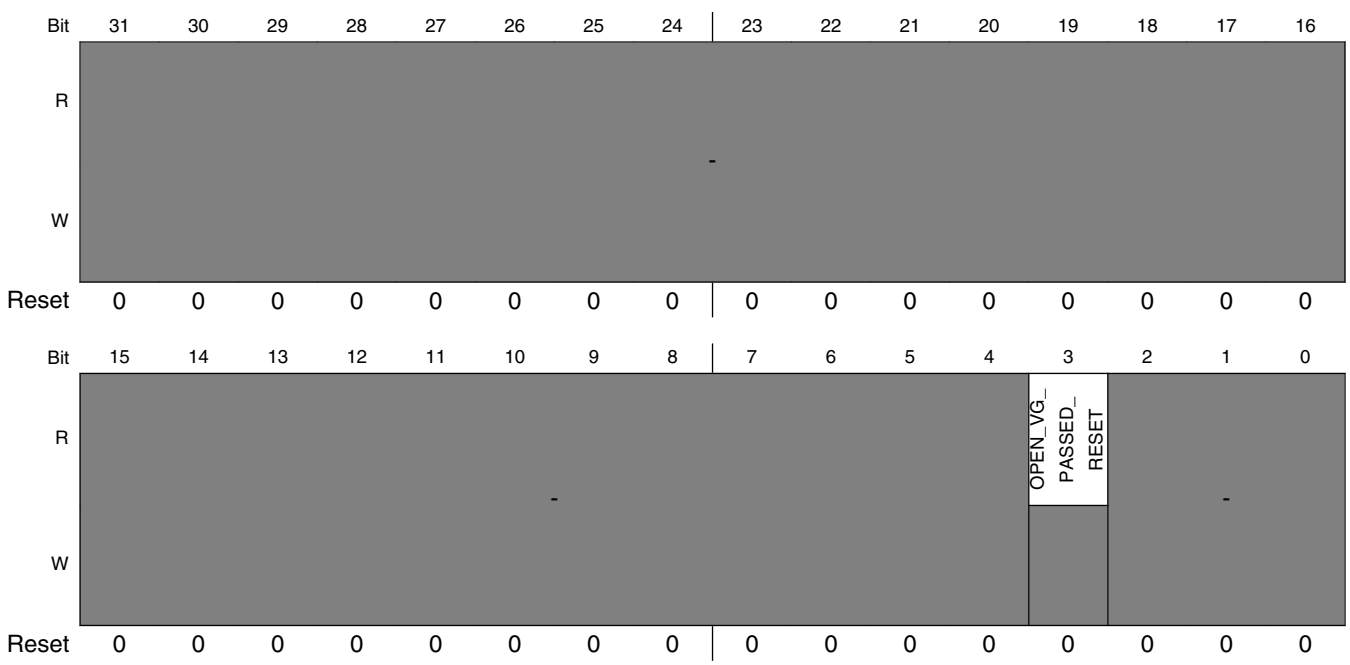
Table continues on the next page...

SRC_SRSR field descriptions (continued)

Field	Description
4 WDOG_RST_B	IC Watchdog Time-out reset. Indicates whether or not the reset was the result of the watchdog time-out event. 0 Reset is not a result of the watchdog time-out event. 1 Reset is a result of the watchdog time-out event.
3 IPP_USER_ RESET_B	Indicates whether or not the reset was the result of the ipp_user_reset_b qulified reset. 0 Reset is not a result of the ipp_user_reset_b qulified as COLD event. 1 Reset is a result of the ipp_user_reset_b qulified as COLD event.
2 -	Reserved.
1 -	Reserved.
0 IPP_RESET_B	Indicates whether or not the reset was the result of ipp_reset_b pin (Power-up sequence) 0 Reset is not a result of ipp_reset_b pin. 1 Reset is a result of ipp_reset_b pin.

49.3.4 SRC Interrupt Status Register (SRC_SISR)

Address: SRC_SISR is 53FD_0000h base + 14h offset = 53FD_0014h



SRC_SISR field descriptions

Field	Description
31–4 -	Reserved
3 OPEN_VG_ PASSED_ RESET	Interrupt generated to indicate that open_vg passed software reset and is ready to be used 0 Interrupt generated not due to open_vg passed reset 1 Interrupt generated due to open_vg passed reset
2–0 -	Reserved

49.3.5 SRC Interrupt Mask Register (SRC_SIMR)

Address: SRC_SIMR is 53FD_0000h base + 18h offset = 53FD_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-												MASK_OPEN_VG_PASSED_RESET	MASK_IPU_PASSED_RESET	MASK_VPU_PASSED_RESET	MASK_GPU_PASSED_RESET
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

SRC_SIMR field descriptions

Field	Description
31–4 -	Reserved
3 MASK_OPEN_ VG_PASSED_ RESET	Mask interrupt generation due to open_vg passed reset 0 Don't mask interrupt due to open_vg passed reset - interrupt will be created 1 Mask interrupt due to open_vg passed reset
2 MASK_IPU_ PASSED_ RESET	Reserved. Set to 1 to mask interrupt
1 MASK_VPU_ PASSED_ RESET	Reserved. Set to 1 to mask interrupt

Table continues on the next page...

SRC_SIMR field descriptions (continued)

Field	Description
0 MASK_GPU_ PASSED_ RESET	Reserved. Set to 1 to mask interrupt

Chapter 50
State Retention Power Gating Controller (SRPGC)

50.1 Introduction

The State Retention Power Gating Controller (SRPGC) is a part of GPC. It controls the save, restore & power gating sequence of a platform implemented using SRPG cells. Power supply to all the logic except the retention latch of the SRPG flop, can be switched off in low power modes. For designs using SRPG flops, a specific sequence of various control signals must be followed in order to save and restore the state of flops correctly. SRPGC provides this sequence based on the various control registers.

50.1.1 Features

The SRPGC features include:

- Provides options to switch power off to a platform.
- Controls the operation of state-retaining/state-restoring circuitry, and stages the retain/restore signals in order to control the IR drop on the power grid during these events.
- Generates Power-up and Power-down control sequences
- Programmable registers to stage the various SRPG and Power control signals
- 32-bit IP Bus interface
- All registers are byte-accessible

50.1.2 Modes of Operation

50.1.2.1 Functional Mode

The SRPGC provides an option to switch OFF power to a platforms. The various power sequence control registers should be programmed to the desired value before powering down a platform.

50.1.2.2 Debug Mode

If the debug bit in the Debug control register is set to one, the SRPGC does not asserts the various outputs in the power down sequence, based on the other control bits in the Debug control register.

50.2 Power Gating Cells

50.2.1 Power Supply Switch Cells

In order to satisfy the power supply requirements of a platform, power switches are placed between the continuous supplies and the gated supplies as shown below. The power switches are composed of parallel power switch cells placed external to the platform.

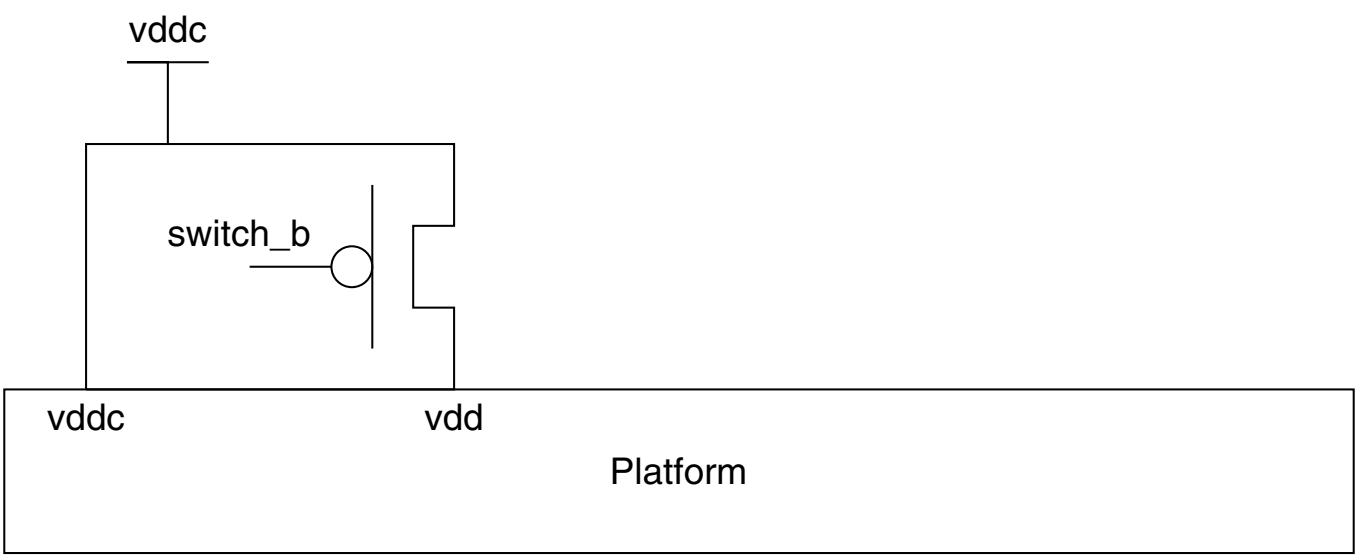


Figure 50-1. Power Switches External to the Platform

The power switch cells are positioned in strategic locations that satisfy the active current requirements of a platform. Table 50-1 describes the pins of the power switch cell.

Table 50-1. Power Switch Cell Pins

Name	Type	Description
vddc	SUPPLY	Continuous supply1 connection
vdd	GATED_SUPPLY	Gated supply1 connection that will be placed into a resistive state when power_en_b is negated.
switch_b	INPUT	Active low power control signal that allows high conductivity between vddc and vdd when asserted. Invokes low conductivity between vddc and vdd when negated.

50.2.2 Power Supply Shorting Cells

Internal to a Platform, there are a number of power supply shorting cells. The power supply shorting cells provide local clamping between the continuous and the gated vdd nodes. Internal clamping ensures that the supply1 reference correlates for both the continuous logic (the state retaining circuitry) and the gated logic (the combinational logic). During periods of high switching activity, IR drops on the continuous supply will be transferred to the gated supply and vice versa. A primary purpose of the shorting cells is to give the Vdd supply access to the nwell-psubstrate diode decoupling capacitance on VDDC.

During power up states, the power supply shorting cells will be placed in a high conductivity configuration by assertions on the SHORT_B signal. Consequently, the power supply shorting cells will be held in a high impedance configuration during power down states.

Figure 50-2 shows a block diagram with a platform power supply shorting cells. The power supply shorting cells are instantiated as a parallel array of cells.

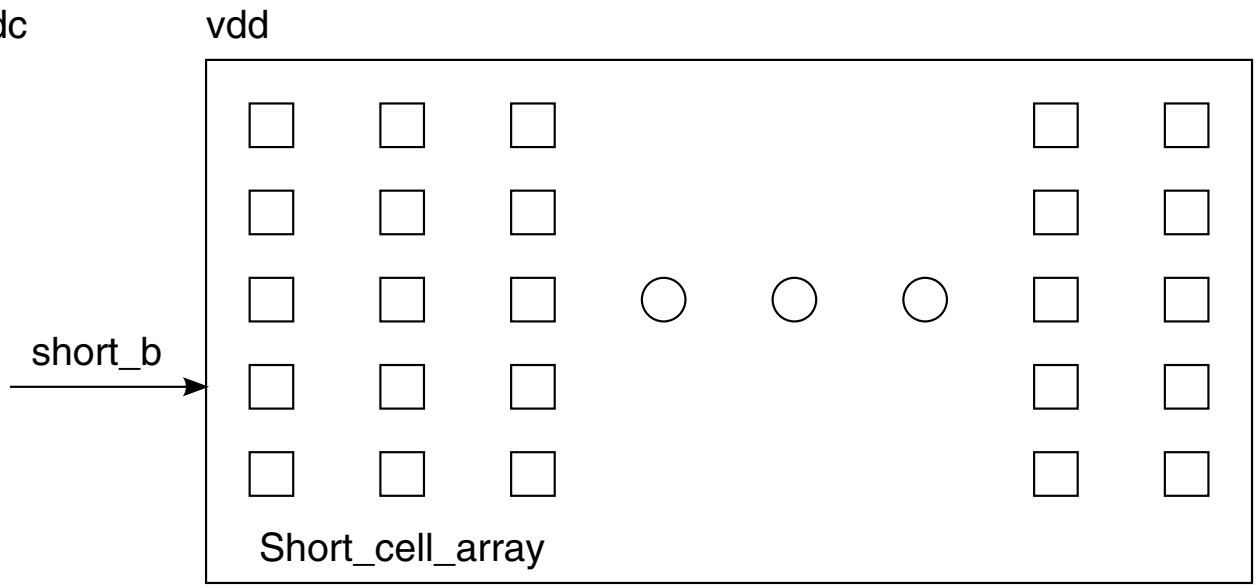


Figure 50-2. Power Supply Shorting Cells Internal to a Platform

Figure 50-3 shows the schematic diagram of power supply shorting cell. This may be combined with a well tie cell.

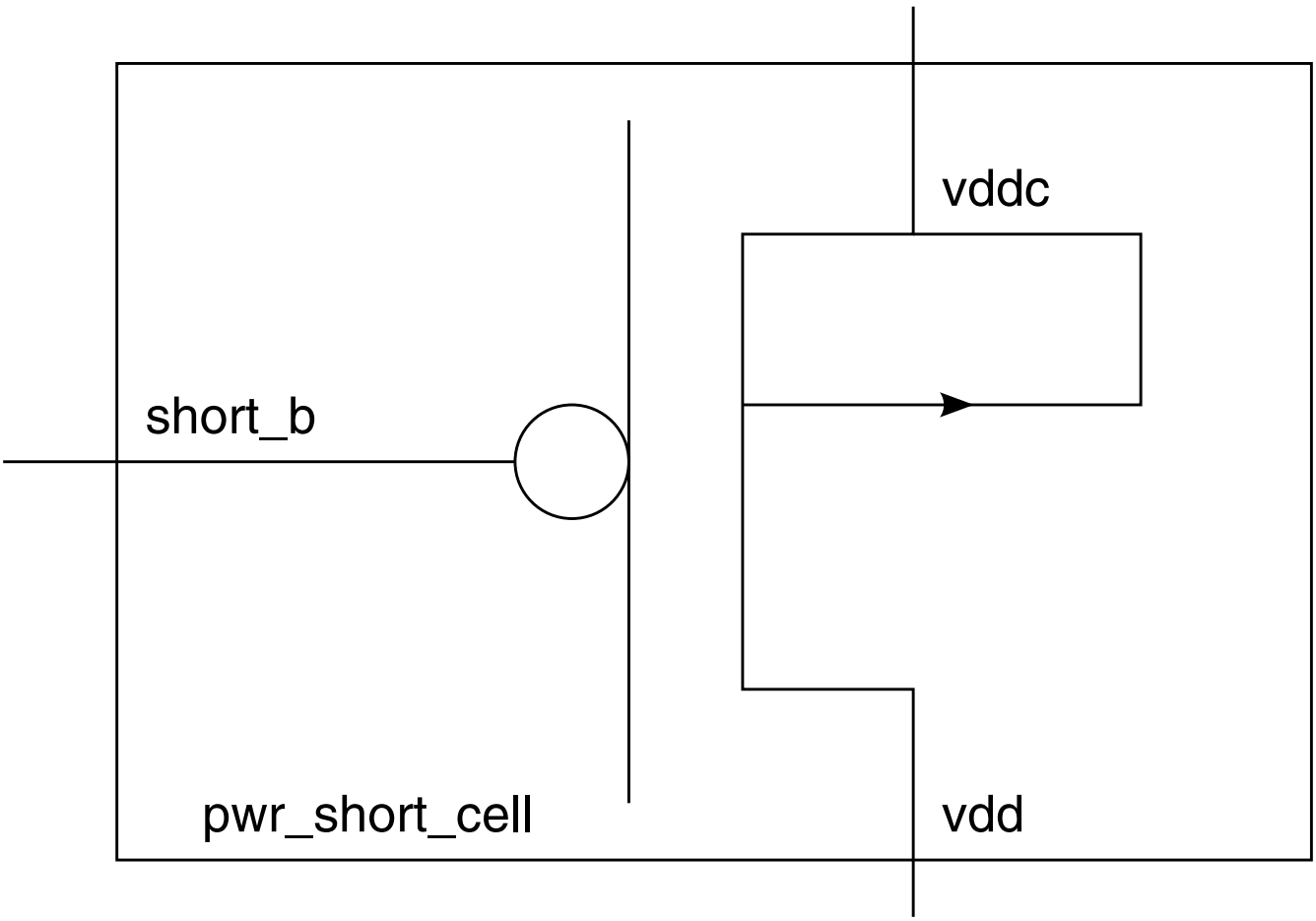


Figure 50-3. Power Supply Shorting Cell Schematic Diagram

Table 50-2 describes the pins of the power supply shorting cell.

Table 50-2. Power Supply Shorting Cell Pins

Name	Type	Description
vddc	SUPPLY	Continuous supply1 connection
vdd	GATED_SUPPLY	Gated supply1 connection that will be placed into a resistive state when short_b is negated.
short_b	INPUT	Active low power control signal that allows high conductivity between vddc and vdd_gtd when asserted. Invokes low conductivity between vddc and vdd_gtd when negated.

50.2.3 State Retention Power Gating (SRPG) Cell

The different platforms have the ability to keep the state of every register element preserved during power gating by using special SRPG flip flop elements. The flip flop consists of a master latch and a slave latch. During active modes, both the master and slave latches are powered up. During power gating, the master latch is completely powered off, but the slave latch remains powered up. This allows the state to be preserved with low leakage. Figure 50-4 shows a simplified version of a normal SRPG flop.

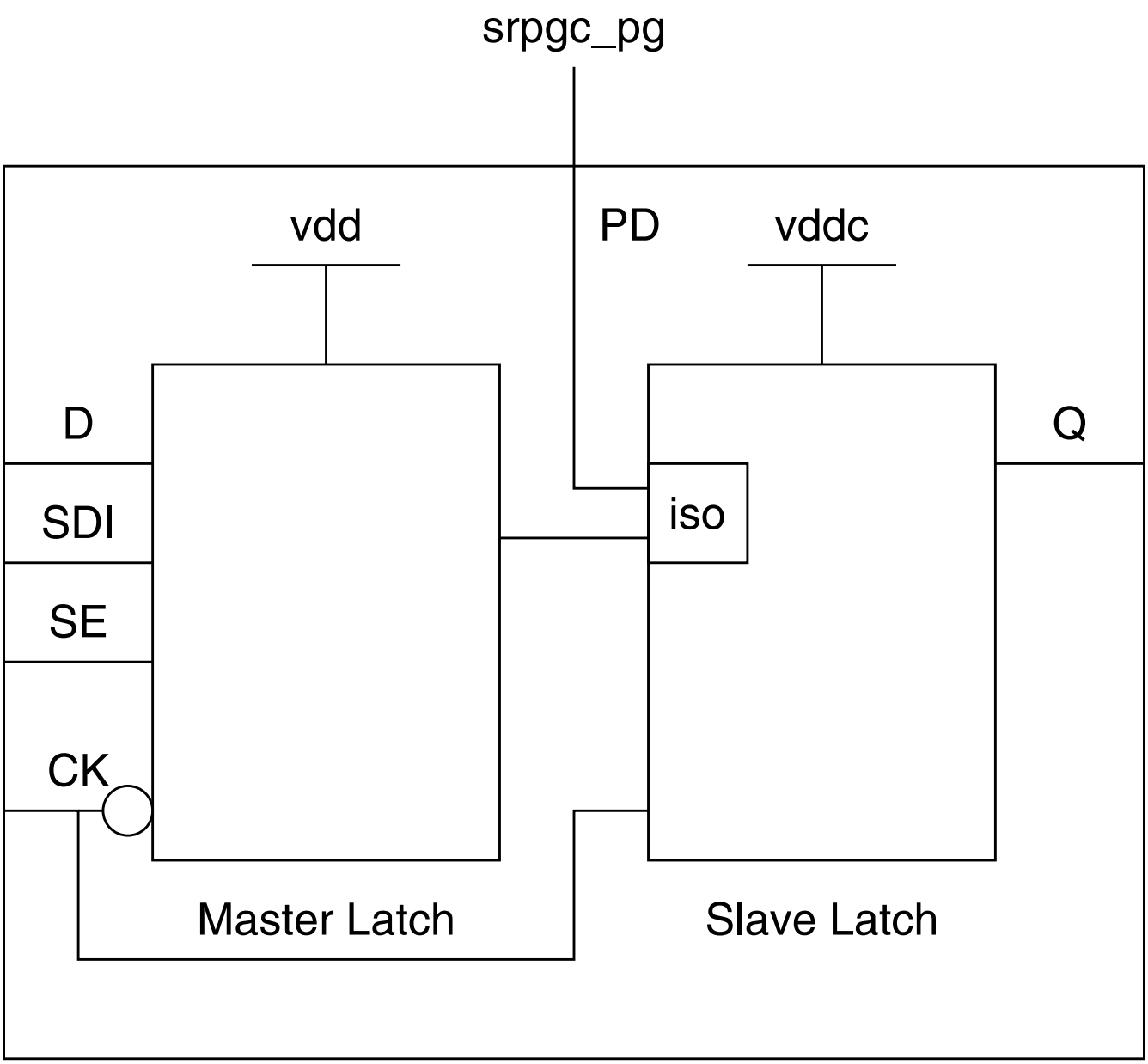


Figure 50-4. Simplified SRPG Cell Diagram

The SRPG flop provides a single extra control signal, PD, that is used to safely isolate the master and the slave latch so that the master latch can be powered down. The PD control signal must remain powered up and asserted during power gating. All other input signals can float including set, reset, clock, etc. The Q output shown in Figure 50-4 and the QN output in an inverted output flop (not shown) will be forced to a value when PD is asserted and vdd is not shut off. This is a result of the internal circuit design and not due to any special requirement. Upon exit from power gating, the PD signal will be deasserted to restore the flip flop to normal operation. For posedge flip-flops, the clock must be low before PD is asserted and held low until PD is deasserted. This also allows for needing only the slave latch state to be saved.

Because there may be many flip flops in a platform, the process of isolating all the flip flops for power gating, and forcing the Q and QN outputs will cause a moment of high instantaneous current, particularly if the Q or QN was previously a different state. Likewise the process of restoring all Q and QN outputs will cause a moment of high instantaneous current. For this reason, the platform must provide multiple PD signals that must be individually asserted/deasserted staged. Staging the negations and assertions on the PD signals provides guard banding against the unacceptable IR drop which might have caused flops to lose state. By transitioning these PD chains at different times during the SRPG_ENTRY and SRPG_EXIT sequences, an acceptable IR drop can be achieved in all cases.

There are several other very important design details which should be adhered to. First, reset synchronizing flops PD signals must be the last PD signals asserted and the first to be deasserted. This ensures proper operation of the reset synchronizing flops which must not have an incorrect Q or QN output prematurely. Second, the PD signals should be asserted/deasserted after the clocks have stopped low for posedge flops. This ensures that the master to slave pass gates are not opened, and is a requirement of the flop design.

The above text describes the SRPG flop architecture called SRPG7. Some variations of the SRPG flop exist. For example, the SRPG2 flop, which is available for LP transistor designs only, does not change Q output when PD is asserted. This lessens the need to stagger PD assertions and eliminates the need to give reset synchronizers their own PD signal. Another flop, applicable only to GP transistor designs, is the SRPG7RS flop. This flop is designed for reset synchronizers and, at the expense of area, does not change the Q output when PD is asserted.

50.3 SRPGC Interface

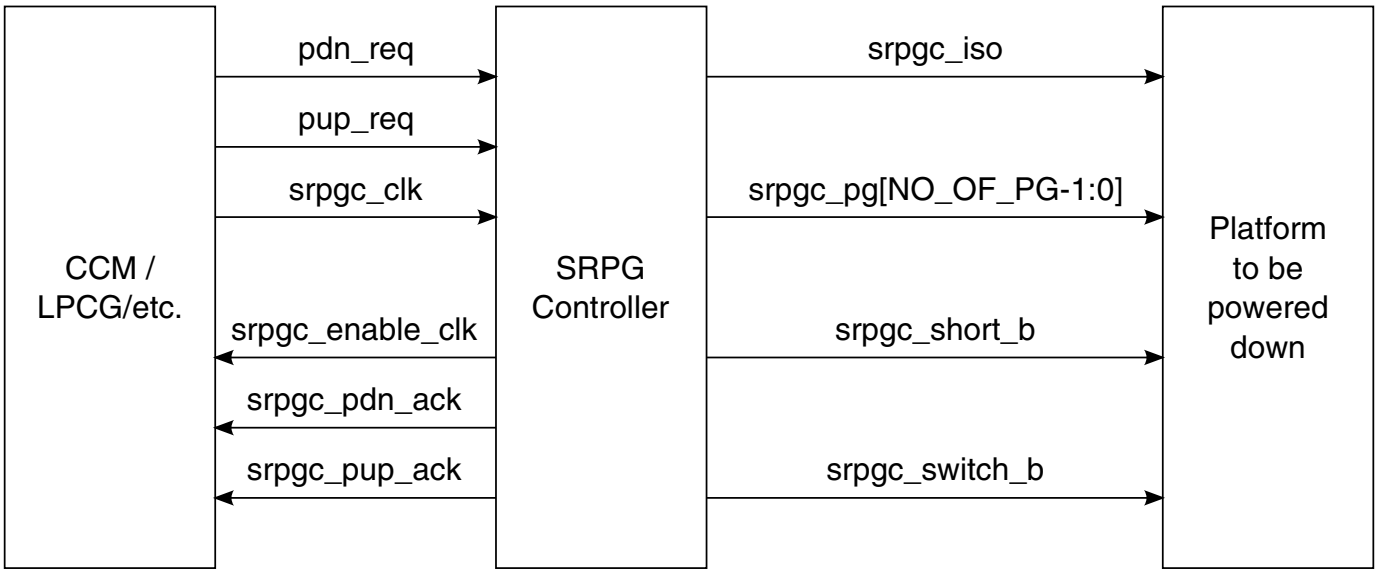


Figure 50-5. SRPGC Interface

50.3.1 Power-Down Sequence (SRPG Entry Sequence)

The power-down sequence control register (SRPGC_PDNSCR) and power-up sequence control register (PUPSCR) must be programmed to the desired value before initiating the "pdn_req" and must not be changed until power up is completed. The SRPG entry sequence is outlined in the following list of steps:

1. Software program PCR bit in the SRPG control register (SRPGC_SRPGCR).
2. CCM stops the clocks to the powering down platform.
3. pdn_req is asserted to SRPGC.
4. Upon detecting the pdn_req assertion, SRPGC asserts "srpgc_enable_clk".
5. SRPGC asserts "srpgc_iso" to isolate the outputs, based on the programming of PDNSCR[5:0].
6. SRPGC asserts "srpgc_pg[0]" to save the state of SRPG flops, based on the programming of PDNSCR[13:8]
7. SRPGC asserts "srpgc_pg[NO_OF_PG-1:1]" to save the state of SRPG flops. "srpgc_pg[1]" is asserted after one "srpgc_clk" of srpgc_pg[0] assertion, "srpgc_pg[2]" is asserted after two "srpgc_clk" of srpgc_pg[0] assertion, "srpgc_pg[NO_OF_PG-1]" is asserted after (NO_OF_PG-1) "srpgc_clk" of srpgc_pg[0] assertion.

8. SRPGC deasserts "srpgc_short_b", based on the programming of PDNSCR[17:16].
9. SRPGC deasserts "srpgc_switch_b", based on the programming of PDNSCR[29:24].
10. SRPGC asserts "srpgc_pdn_ack", one "srpgc_clk" cycle wide pulse.
11. SRPGC deasserts "srpgc_enable_clk".

NOTE

If the "pup_req" is asserted before completion of power down then the SRPGC stops the power down sequence immediately and starts the power-up sequence. "srpgc_pdn_ack" is asserted as soon as the power down sequence stops. If "pup_req" is asserted before or along with "pdn_req" then power down sequence does not start and "srpgc_pdn_ack" is not asserted by SRPGC.

50.3.1.1 Power-Down Sequence Timing Waveforms

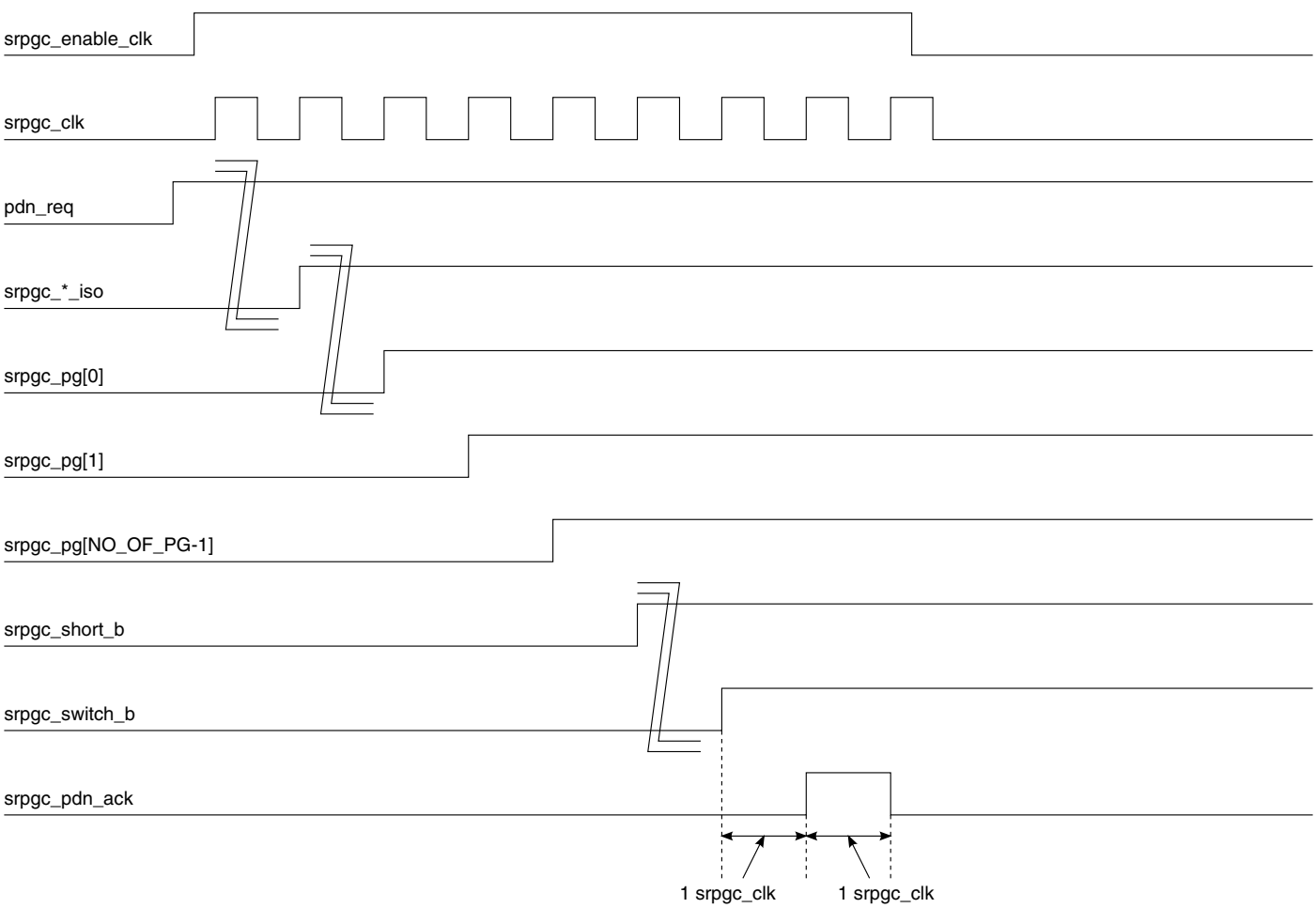


Figure 50-6. Power-down Sequence Timing Waveforms

50.3.2 Power-Up Sequence (SRPG Exit Sequence)

SRPG exit sequence is outlined in the following list of steps:

1. pup_req is asserted to SRPGC.
2. SRPGC asserts "srpgc_enable_clk" to enable the clock.
3. SRPGC asserts "srpgc_switch_b", based on the programming of PUPSCR[5:0].
4. SRPGC asserts "srpgc_short_b", based on the programming of PUPSCR[13:8].
5. SRPGC deasserts "srpgc_pg[NO_OF_PG-1]" to restore the state of SRPG flops, based on the programming of PUPSCR[21:16].
6. SRPGC deasserts "srpgc_pg[NO_OF_PG-2:0]" to restore the state of SRPG flops. "srpgc_pg[NO_OF_PG-2]" is deasserted after one "srpgc_clk" of srpgc_pg[NO_OF_PG-1] deassertion, "srpgc_pg[NO_OF_PG-3]" is deasserted after two "srpgc_clk" of srpgc_pg[NO_OF_PG-1] deassertion, "srpgc_pg[0]" is deasserted after (NO_OF_PG-1) "srpgc_clk" of srpgc_pg[NO_OF_PG-1] deassertion.
7. SRPGC deasserts "srpgc_iso" to remove the isolation from the output of the platform, based on the programming of PUPSCR[25:24].
8. SRPGC asserts "srpgc_pup_ack" after one clock cycles of "srpgc_iso" deassertion. This is a one "srpgc_clk" cycle wide pulse.
9. "pup_req" is deasserted to SRPGC.
10. SRPGC deasserts "srpgc_enable_clk".

NOTE

If the "pup_req" is asserted before completion of power down then the SRPGC stops the power down sequence immediately and starts the power-up sequence. "srpgc_pup_ack" is asserted after completion of power up. If "pup_req" is asserted before "pdn_req" then SRPGC does not initiate the power down sequence and asserts the "srpgc_pup_ack".

50.3.2.1 Power-up Sequence Timing Waveforms

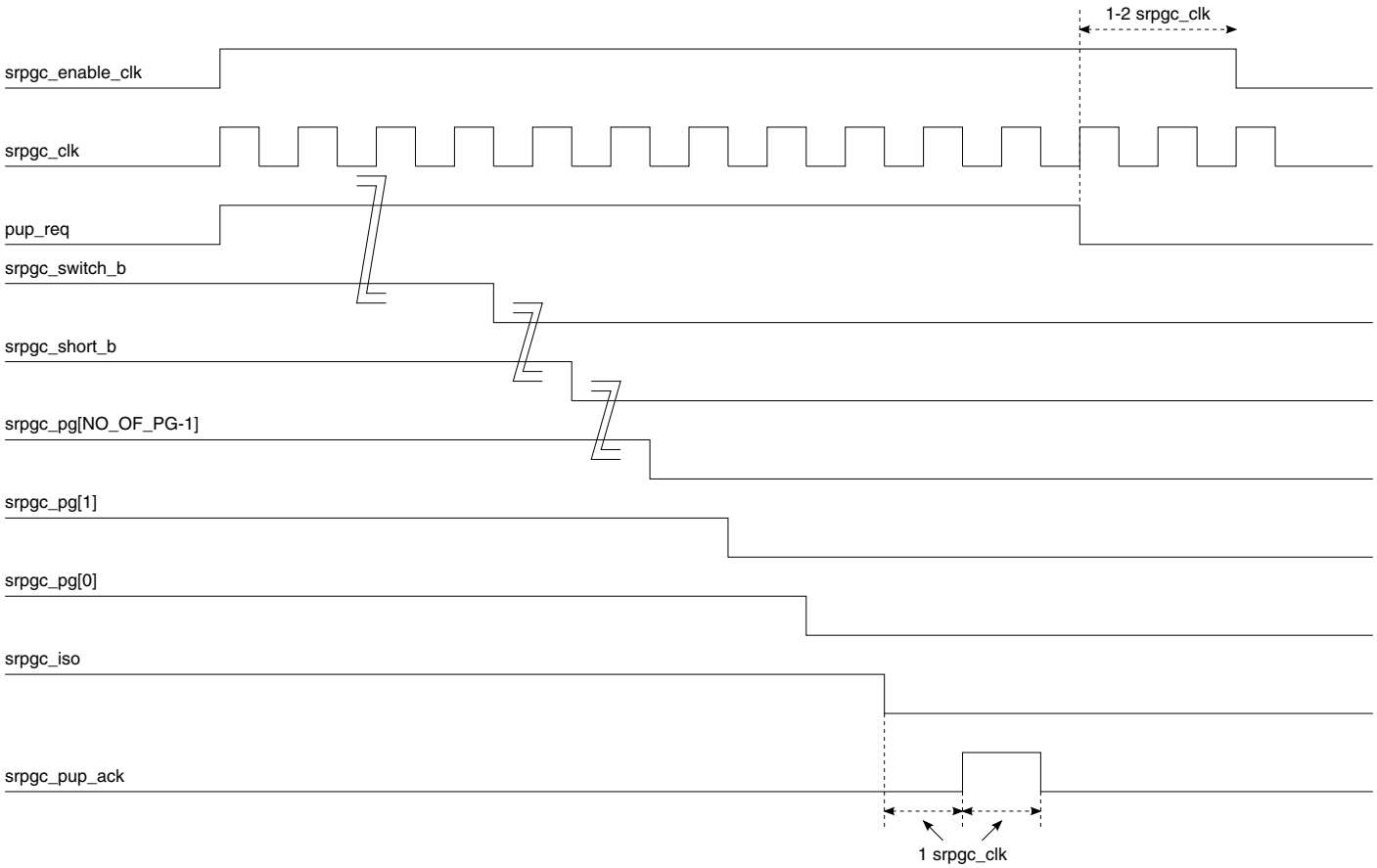


Figure 50-7. Power-up Sequence Timing Waveforms

50.4 Programmable Registers

The SRPGC registers can only be accessed in supervisor mode. Any access in normal mode, or an access to an unimplemented address location can assert transfer error signal "ips_xfr_err", if "resp_sel" is low.

SRPGC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FD_8280	SPRGC Control Register (SRPGC_SRPGCR)	32	R/W	0000_0000h	50.4.1/ 2864

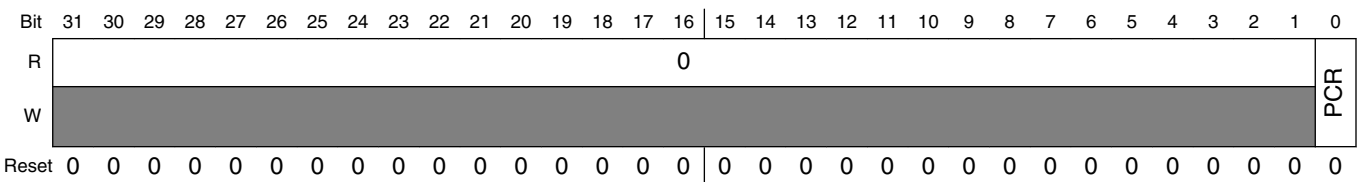
Table continues on the next page...

SRPGC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FD_8284	Power-up Sequence Control Register (SRPGC_PUPSCR)	32	R/W	0102_0F01h	50.4.2/ 2864
53FD_8288	Power-down Sequence Control Register (SRPGC_PDNSCR)	32	R/W	0101_0101h	50.4.3/ 2865
53FD_828C	SPRGC Status Register (SRPGC_SRPGSR)	32	w1c	0000_0000h	50.4.4/ 2866
53FD_8290	SPRGC Debug Register (SRPGC_SRPGDR)	32	R/W	0000_0000h	50.4.5/ 2867

50.4.1 SPRGC Control Register (SRPGC_SRPGCR)

Address: SRPGC_SRPGCR is 53FD_8280h base + 0h offset = 53FD_8280h

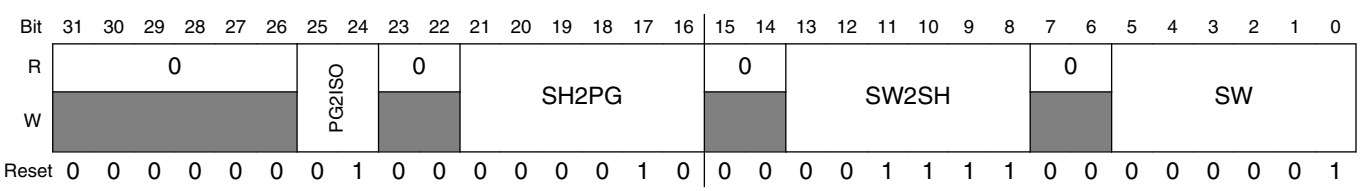


SRPGC_SRPGCR field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 PCR	Power control WARNING: This bit should not change from "pdn_req" assertion until platform is completely powered up. 0 Do Not Switch OFF power even if the pdn_req is asserted 1 Switch OFF Power when pdn_req is asserted

50.4.2 Power-up Sequence Control Register (SRPGC_PUPSCR)

Address: SRPGC_PUPSCR is 53FD_8280h base + 4h offset = 53FD_8284h

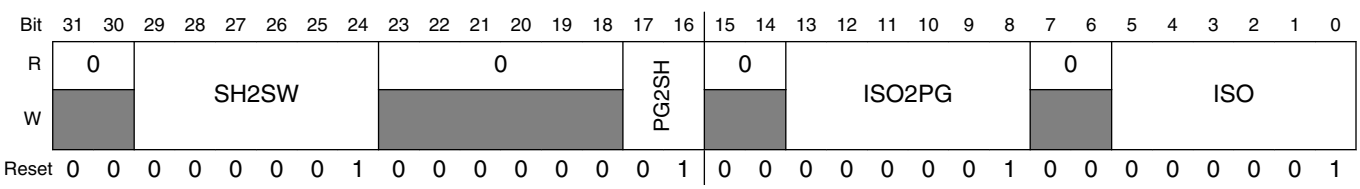


SRPGC_PUPSCR field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value zero. Reserved
25–24 PG2ISO	Number of clocks from Last PG (srpgc_pg[0]) deassertion to ISO deassertion. **NOTE:** It is not recommended to program these bits to zero.
23–22 Reserved	This read-only field is reserved and always has the value zero. Reserved
21–16 SH2PG	Number of clocks from SHORT assertion to First PG (srpgc_pg[NO_OF_PG-1]) deassertion. **NOTE:** It is not recommended to program these bits to zero.
15–14 Reserved	This read-only field is reserved and always has the value zero. Reserved
13–8 SW2SH	Number of clocks from SWITCH assertion to SHORT assertion. **NOTE:** It is not recommended to program these bits to zero.
7–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5–0 SW	Number of clocks to assert SWITCH from Power-up request. WARNING: This should not be programmed to zero. A zero value could cause a glitch on the srpgc_switch_b if the power-up request (pup_req) is asserted immediately after deassertion of srpgc_switch_b.

50.4.3 Power-down Sequence Control Register (SRPGC_PDNSCR)

Address: SRPGC_PDNSCR is 53FD_8280h base + 8h offset = 53FD_8288h



SRPGC_PDNSCR field descriptions

Field	Description
31–30 Reserved	This read-only field is reserved and always has the value zero. Reserved
29–24 SH2SW	Number of clocks from SHORT deassertion to SWITCH deassertion. **NOTE:** It is not recommended to program these bits to zero.
23–18 Reserved	This read-only field is reserved and always has the value zero. Reserved
17–16 PG2SH	Number of clocks from last PG (srpgc_pg[NO_OF_PG-1]) assertion to SHORT deassertion. **NOTE:** It is not recommended to program these bits to zero.

Table continues on the next page...

SRPGC_PDNSCR field descriptions (continued)

Field	Description
15–14 Reserved	This read-only field is reserved and always has the value zero. Reserved
13–8 ISO2PG	Number of clocks from ISO assertion to First PG (srpgc_pg[0]) assertion. **NOTE:** It is not recommended to program these bits to zero.
7–6 Reserved	This read-only field is reserved and always has the value zero. Reserved
5–0 ISO	Number of clocks from Power down request to ISO assertion. WARNING: It is not recommended to Program ISO to zero, if it is programmed to zero, ISO may be deasserted before the power is completely turned on

50.4.4 SPRGC Status Register (SRPGC_SRPGSR)

Address: SRPGC_SRPGSR is 53FD_8280h base + Ch offset = 53FD_828Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

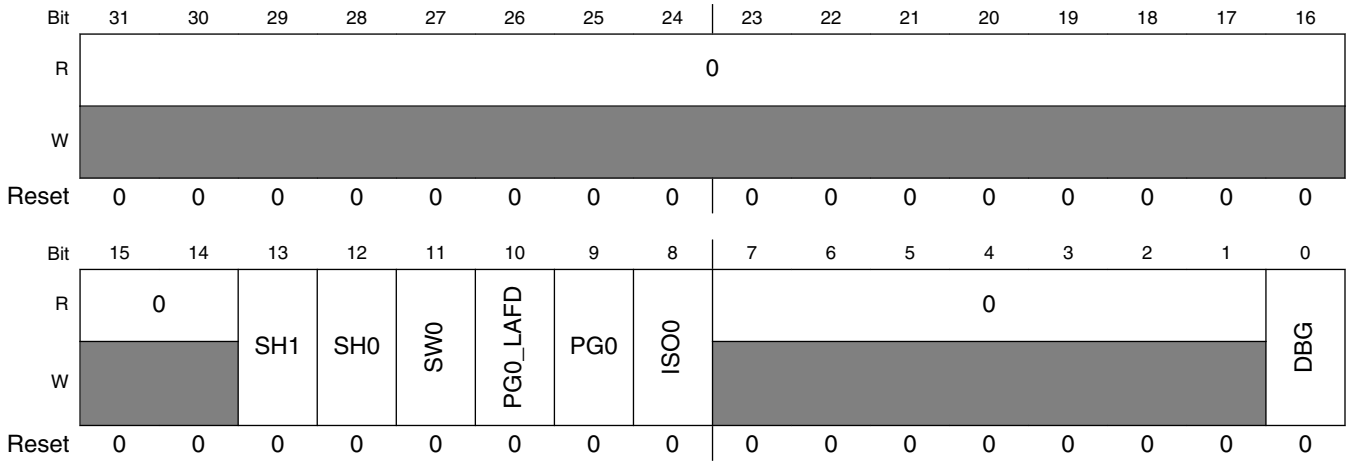
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															PSR
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRPGC_SRPGSR field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 PSR	Power status. Write 1 to clear this bit. User should clear this bit after power-up, otherwise this bit will continue to reflect the power status of the very first power down. 0 was NOT powered down in previous power down request 1 was powered down in previous power down request.

50.4.5 SPRGC Debug Register (SRPGC_SRPGDR)

Address: SRPGC_SRPGDR is 53FD_8280h base + 10h offset = 53FD_8290h



SRPGC_SRPGDR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value zero. Reserved
13 SH1	srpgc_short_b control in debug mode. This bit has no effect if DBG bit is zero. **NOTE:** srpgc_short_b is asserted high as soon as SH1 is written to one if DBG bit is one and SH0 bit is zero. 0 srpgc_short_b is not forced to one. 1 srpgc_short_b is forced to one.
12 SH0	srpgc_short_b control in debug mode. This bit has no effect if DBG bit is zero. **NOTE:** If SH0 is also set when SH1 is set then srpgc_short_b is driven to Zero. 0 srpgc_short_b is not forced to zero. 1 srpgc_short_b is forced to zero.
11 SW0	srpgc_switch_b control in debug mode. This bit has no effect if DBG bit is zero. 0 srpgc_switch_b is not forced to zero. 1 srpgc_switch_b is forced to zero.
10 PG0_LAFD	srpgc_pg[NO_OF_PG-1](Last asserted and First deasserted) control in debug mode. This bit has no effect if DBG bit is zero. **NOTE:** When NO_OF_PG is one then this bit forces srpgc_pg[0] to zero. 0 srpgc_pg[NO_OF_PG-1] is not forced to zero. 1 srpgc_pg[NO_OF_PG-1] is forced to zero.
9 PG0	srpgc_pg[NO_OF_PG-2: 0] control in debug mode. This bit has no effect if DBG bit is zero. **NOTE:** When NO_OF_PG is one then this bit has no meaning.

Table continues on the next page...

SRPGC_SRPGDR field descriptions (continued)

Field	Description
	0 srpgc_pg[NO_OF_PG-2:0] is not forced to zero. 1 srpgc_pg[NO_OF_PG-2:0] is forced to zero.
8 ISO0	srpgc_iso control in debug mode. This bit has no effect if DBG bit is zero. 0 srpgc_iso is not forced to zero. 1 srpgc_iso is forced to zero.
7–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 DBG	Debug control. 0 Debug mode is not selected. 1 Debug mode is selected.

Chapter 51
Secure Real Time Clock (SRTC)

51.1 Overview

This section briefly introduces the block. The full description of the block is in Functional Description.

The SRTC is comprised of two separated sub-blocks:

- Low power domain SRTC (SRTC LP)
- High power domain SRTC (SRTC HP)

SRTC LP is in an always powered up domain and is isolated from the rest of the logic by means of an isolation cell. Isolation cells are library instantiated cells, which insure that the powered up logic does not get corrupted when the power goes down in the rest of the SoC. It is powered by the Coin Cell battery.

SRTC HP is in the normal supply domain. It is powered by the main digital supply powering the SoC.

51.1.1 Low Power SRTC (SRTC LP) Overview

The SRTC LP is a real time clock with enhanced security capabilities. Its purpose is to provide an accurate time read at all times, regardless of the main system power state and without the need to use an external on-board time source such as external RTC. The SRTC LP can wake up the system when preset time alarm is asserted.

It is divided into following units:

- Counters and registers
  - Non-rollover 47-bit time counter
  - 32-bit alarm register

    - General purpose registers
    - Secured monotonic counter
- SRTC Monitor
    - SRTC state machine
    - SRTC power and clock detectors

Figure 51-1 shows the SRTC partitioning.

51.1.2 High Power SRTC (SRTC HP) Overview

The SRTC HP contains all the interfaces to the core (IP Bus Interface). It contains a counter, an alarm and has the capability of generating a number of user-defined periodic interrupts. Access to SRTC LP registers can only be done through the SRTC HP and when SRTC HP is powered up.

SRTC HP is partitioned into following units.

- Address Decode
- Counters and registers
    - Nonsecure 47-bit time counter
    - Programmable nonsecure 47-bit alarm with interrupt
    - Periodic alarm with interrupt

The nonsecure time counter can be synchronized with the secured time counter by writing to a SRTC HP bit (time_sync-Time Synchronization). When written, the content of the secure counter is automatically copied to the nonsecure counter.

The following figure shows the SRTC partitioning.

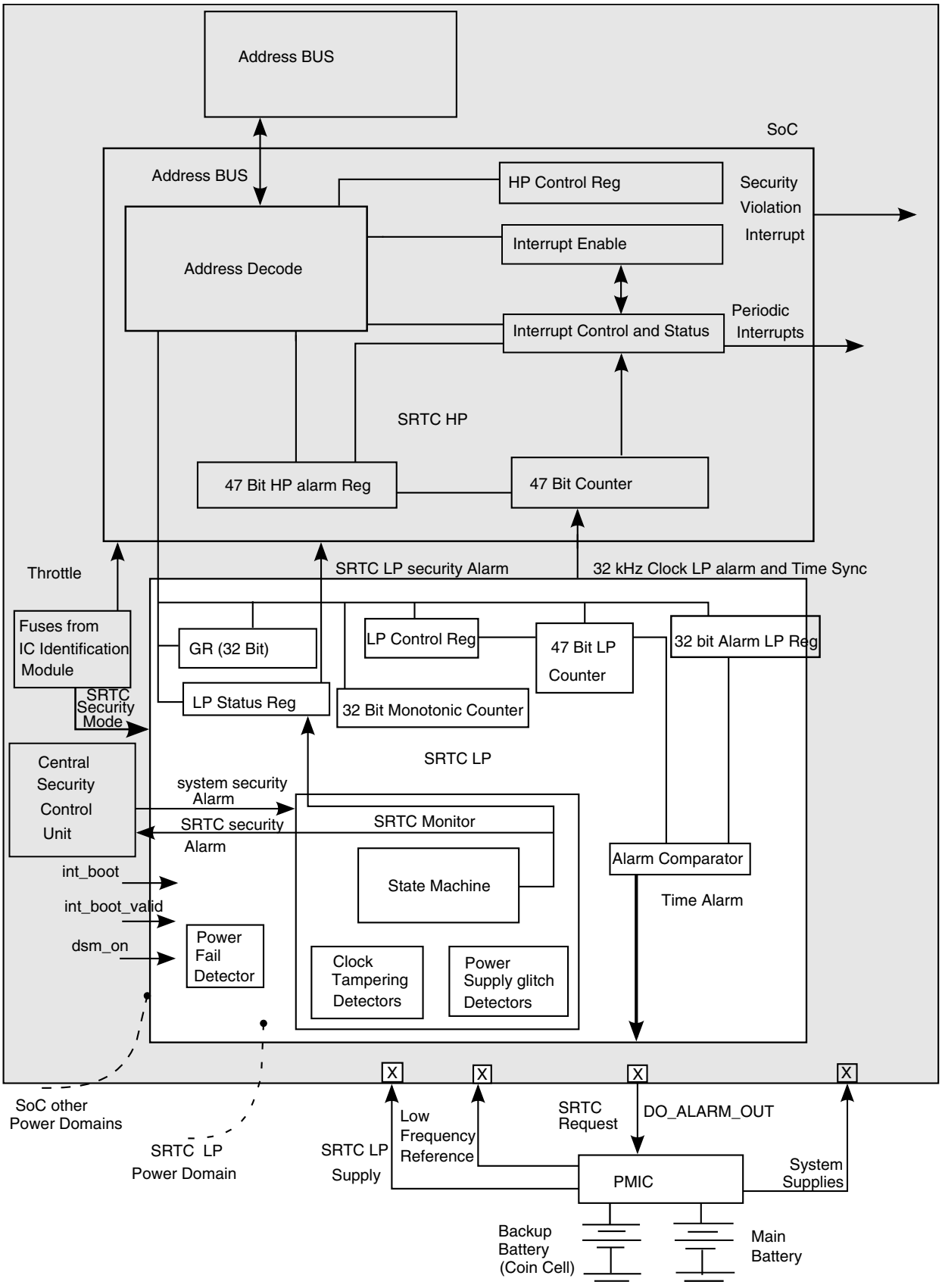


Figure 51-1. Block Diagram

51.1.3 Features

The SRTC includes the following features:

- Secure 47-bit time counter running on 32.768 kHz clock
- Nonsecure 47-bit time counter running on 32.768 kHz clock
- User mode protection- SRTC secured registers cannot be configured by nonsecured SW (see the Programmable Registers section). A dedicated control bit can allow nonsecured SW to update the secured registers. This bit can be set only by secured SW. In any case, SRTC secured registers cannot be configured when SRTC is locked.
- Re-programming protection:
  - SRTC time cannot be altered after SRTC is locked (when SRTC set to the appropriate security mode)
  - SRTC cannot be disabled after SRTC is locked (when SRTC set to the appropriate security mode)
- Power loss protection:
  - Separated power supply
  - Accurate continuous time is kept during power down system states and main power source losses
  - In the event of a total power loss (main power source and backup power source), SRTC time read is invalidated
  - In the event of unstable voltage power source, SRTC time read is invalidated
  - In the event of low voltage power source that disrupts normal time counting, SRTC time read is invalidated
- Clock source protection:
  - Directly connects to a low frequency crystal (32.768 kHz) or to a PMIC 32.768 kHz clock source
  - In the event that SRTC frequency drops, SRTC time read is invalidated
- SRTC time can be invalidated by a system security alarm
- SRTC can assert a system security alarm and fast interrupt request to alert the processor that time read was invalidated
- Programmable secure 32-bit alarm with interrupt (alarm of the secure- LP section). In case of system power up mode, this alarm generates interrupt to alert the processor. In case of system power down mode, this alarm alerts the power management IC (PMIC) via an external pin to instruct the PMIC to power up the system. Note that this feature does not require any PMIC special support. This signal can be connected to an on-board power on button circuit.
- Programmable nonsecure 47-bit alarm with interrupt (alarm of the nonsecure, HP, section)

- Periodic nonsecure alarm with interrupt
- Secured monotonic counter
- General purpose always powered registers
- Ultra low power consumption. SRTC LP power consumption during system power down state is less than 3 µA.(TYP)
- Separate calibration logic for 47 bit HP and LP counters

51.1.4 Modes of Operations

The SRTC has three functional modes, defined by IC Identification Module dedicated fuses, and can be found in one of four states.

The SRTC modes are as follows:

- Mode #1 - High Security
- Mode #2 - Medium Security
- Mode #3 - Low Security

SRTC states are defined as follows:

- Initialization- SRTC is powered up for the first time or after system POR.
- Nonvalid- SRTC time is not set, time read is not authenticated
- Valid- Valid time was set, time read is authentic
- Failure- SRTC logic detected a failure. Time is invalidated.

See SRTC State Machine, for more details.

NOTE

Detailed information about the SRTC security modes is provided in the Security Reference Manual. Contact your Freescale representative for information about obtaining this document.

51.2 External Signal Description

Table 51-1. Signal Properties[1]

Name	PORT	Function	I/O	Reset	Pull Up
IND_CKIL_IN	CKIL	Input 32.768 kHz Low Frequency Reference clock for the LP Section.	I	0	-
DO_ALARM_OUT	SRTC_ALARM	Output wake up alarm to PMIC on time out. Polarity is active low.	O	0	-

1. This table indicates external signal connected to PAD

51.3 Functional Description

The SRTC is a real time clock with enhanced security capabilities. Its purpose is to provide an accurate time read at all times, regardless of the main system power states and without the need to use an external time source such as external RTC. The following sections describe in detail the theory and basic design principals of the SRTC.

51.3.1 Power and Clock Source

The SRTC is divided into two main sub-blocks based on the power supply.

The SRTC LP is the always powered section of the SRTC. SRTC LP is a separate power domain with its own power supply. No other logical sub-blocks within the SoC boundary share the power supply with this section.

SRTC LP power is supplied by a power management IC (PMIC), such as Freescale MC13783 (Atlas). It is assumed that the power management IC is responsible for supplying a stable voltage level at all times from a main power source or from a back-up power source (for example, a coin-cell battery). The SRTC LP is not connected directly to a backup power source and is not responsible for switching between power sources in case of main power source losses: main battery removals or power cuts (momentary lost of power). It is assumed that the PMIC is responsible for switching between main power source and backup power source as required.

The SRTC LP section is separated from the rest of the SRTC and SoC by means of isolation cells. SRTC receives a signal which indicates a power loss to the SoC, and enables the isolation cells. This signal is deactivated once stable power is restored to the SoC.

Figure 51-2 shows low power domain connectivity. SRTC HP section is in the normal power supply domain and receive power along with the rest of the SoC.

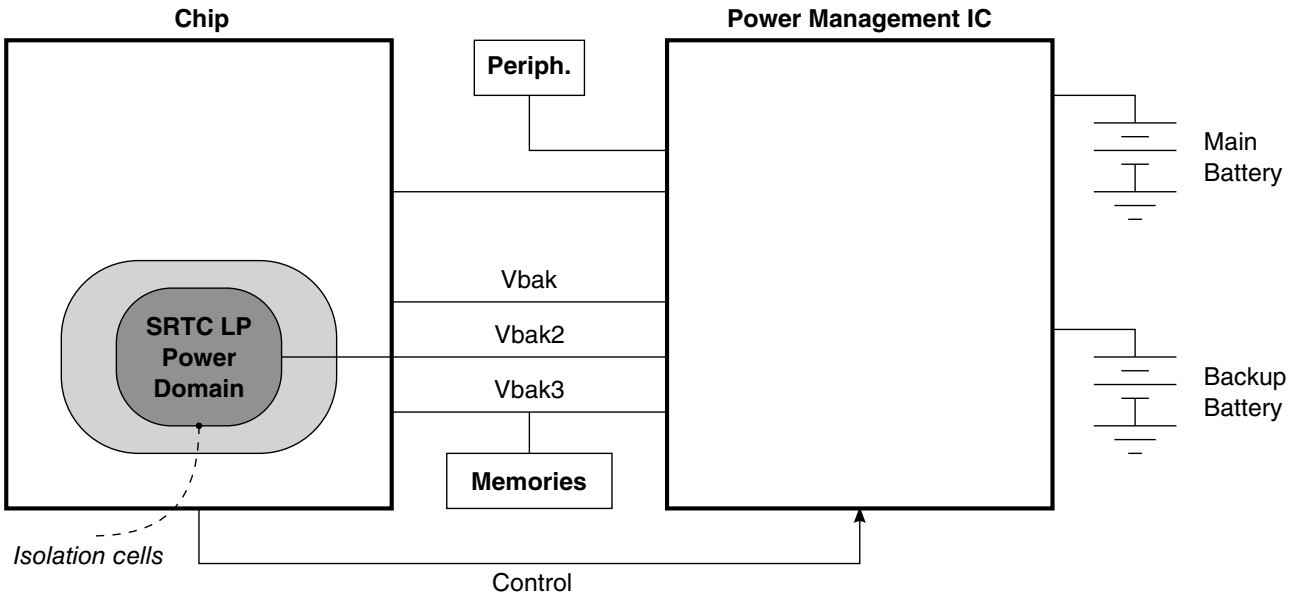


Figure 51-2. SRTC Power Domains

51.3.1.1 Clocks

SRTC runs on two clock sources, high frequency peripherals clock and low frequency timers clock. The high frequency peripheral IP clock is used by the SRTC peripheral bus interface, control and status registers. The low frequency 32.768 kHz clock is the always-active clock used by the SRTC timers and by the PTD and CTD logics. This can be driven directly from an oscillator or from a PMIC clock source.

51.3.2 High Power SRTC (SRTC HP) Description

The SRTC HP contains all the interfaces to the core (IPBus interface). Access to SRTC LP registers can only be done through the SRTC HP and when SRTC HP is powered up. It also contains the following sub- blocks.

51.3.2.1 SRTC HP nonsecured Counter

The SRTC HP incorporates autonomous 47-bit nonsecured time counter. The counter is not active and is reset when system is powered down, however, it remains active during system low-power modes (for example, wait, doze and stop). The nonsecure counter can be used by any application as it has no access restrictions. The SRTC HP counter can be

synchronized with the SRTC LP secure time counter by writing to a specific bit in the SRTC HP control register. After synchronization, time integrity of the nonsecured timer cannot be guaranteed as it has no access limitations

51.3.2.1.1 SRTC HP Counter Calibration

A clock calibration system is provided to adjust the 32,768 cycle counter that generates the 1 Hz timer for SRTC HP timing registers. The general implementation relies on the system processor to measure the 32.768 kHz crystal oscillator against a higher frequency and more accurate system clock such as a TCXO. If the SRTC timer needs a correction, a 5-bit 2-second complement calibration word can be sent via IP to compensate the SRTC for inaccuracy in its reference oscillator as defined in the Table 51-2. The available correction range should be sufficient to ensure that drift accuracy is in compliance with standards for secure time applications like DRM time keeping. Note that the 32.768 kHz oscillator is not affected by SCAL_HP settings; calibration is only applied to the SRTC time base counter. Therefore, the frequency at the clock output Low Frequency Reference Clock is not affected.

The SRTC system calibration is enabled by programming the SCALM_HP for desired behavior by operational mode.

Table 51-2. SRTC HP Calibration Setting

Calibration Settings Code in SCAL_HP[4:0	Correction in Counts per 32768	Relative Correction in ppm
01111	+15	+458
00011	+3	+92
00001	+1	+31
00000	0	0
11111	-1	-31
11101	-3	-92
10001	-15	-458
10000	-16	-488

However, the calibration is only as good as the SCAL_HP data that has been provided, so occasional refreshing is recommended to ensure that any drift-influencing environmental factors have not skewed the clock beyond desired tolerances.

See HP Control Register (SRTC_HPCR), for bit definitions of SCAL_HP and SCALM_HP.

51.3.2.2 SRTC HP nonsecured Counter Alarm

The SRTC HP nonsecured counter has its own 47-bit time alarm register. An alarm is when the 47 bits of HP counter matches with the value of the 47-bit HP alarm register. If X is the value at which the alarm is desired, then this HP Alarm register should be set with value X-1. Note that this register can be updated by any application. The SRTC HP time alarm can generate an interrupt to alert the ARM core to wake it up from one of its low-power modes (for example, wait, doze, stop). Note that this alarm cannot wake up the entire system if it is powered off.

A nonsecure alarm has been provided with mask bit AHP in the SRTC_HPCR.

51.3.2.3 SRTC HP Periodic Alarm

The SRTC HP nonsecure counter can support a generation of a millisecond resolution periodic alarm. The periodic interrupt generation can be set in a range of 1Hz to 32.768 kHz. This interrupt is enabled and set from the SRTC Interrupt enable register. The interrupt source of the periodic alarm is chosen by comparing one of the16 LSB Bits of the 47-bit nonsecured counter (this interrupt is generated when a zero-to-one and one-to-zero transition occurs on the selected bit).

HP SRTC nonsecure counter and its alarms are shown in Figure 51-3.

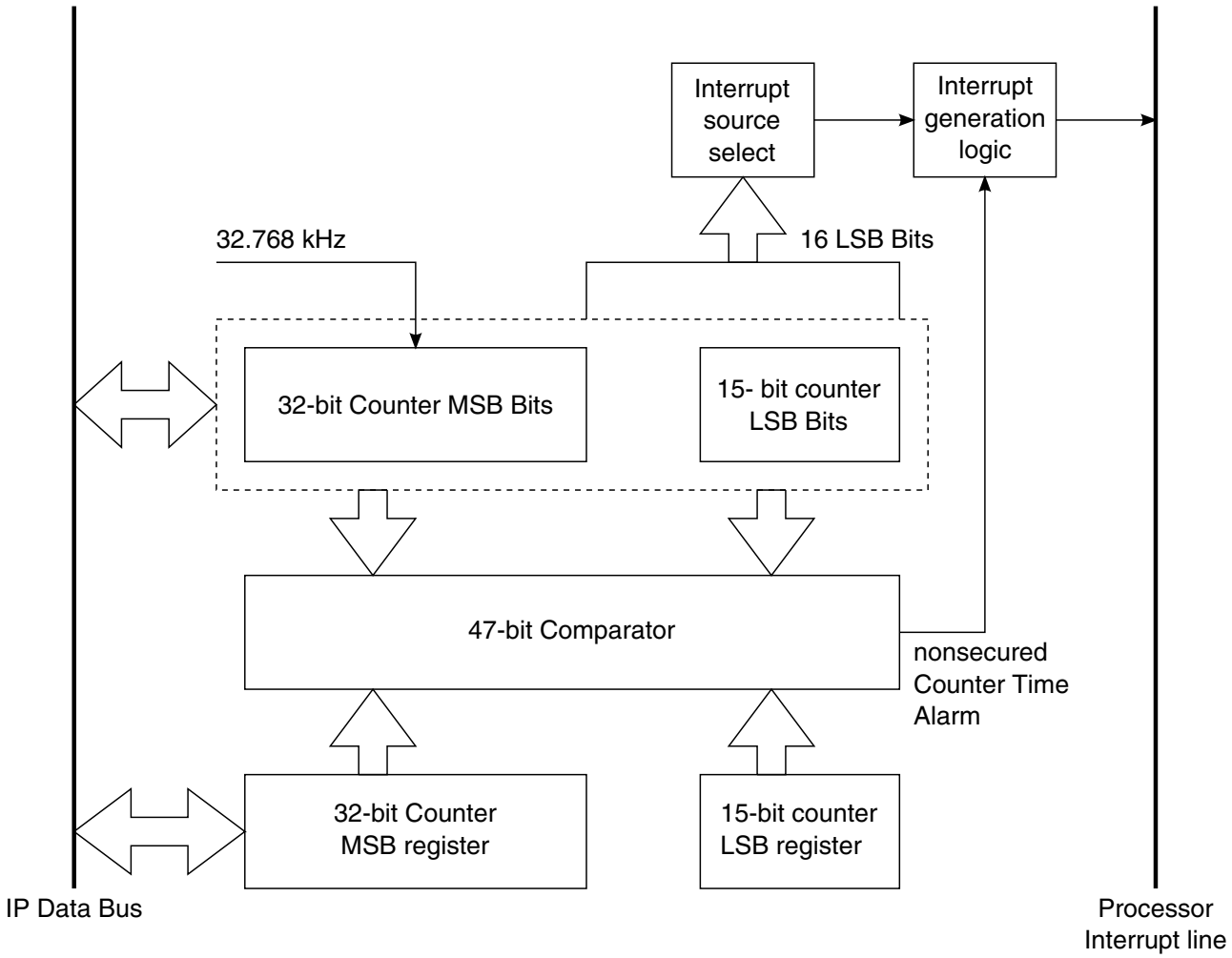


Figure 51-3. SRTC HP nonsecured Counter

51.3.3 Low Power SRTC (SRTC LP) Description

The SRTC LP is a real time clock with enhanced security capabilities. Its purpose is to provide an accurate time read at all times, regardless of the main system power state and without the need to use external on-board time source such as external RTC. The SRTC LP can wake up the system when preset time alarm is asserted. The SRTC LP also contains a monotonic counter and a general purpose register that can be used to store important information while the system is powered down. The following sections describe in detail the theory and basic design principals of the SRTC LP sub-block.

51.3.3.1 SRTC LP Behavior during System Power Down and POR

On system power down the SRTC LP continues to operate normally. It is isolated from the rest of the SoC by means of isolation cells. It keeps the time and retains all parameters stored in its registers. On system power down, the SRTC ignores all system signals (SRTC LP inputs) such as IC Identification Module, SRTC HP, and other input signals. On PoR SRTC LP ignores all system input signals (SRTC LP inputs) until the CCM indicates that the PoR routine ended. This is done to eliminate false signal indications caused by an unstable power state of the system.

51.3.3.2 SRTC LP Secured Counter

The SRTC incorporates autonomous 47-bit secured time counter. Time counter is a non-rollover counter. This means that if the time counter reaches the value of 0xffff_ffff_ffff, it does not rollover. However, the TR bit is set in the SRTC_LPSR and the SRTC enters failure state. This has been done to prevent an attacker from virtually running the time backwards using a higher frequency clock source.

51.3.3.2.1 SRTC LP Counter Calibration

A clock calibration system is provided to adjust the 32,768 cycle counter that generates the 1 Hz timer for SRTC LP timing registers. The general implementation relies on the system processor to measure the 32.768-kHz crystal oscillator against a higher frequency and more accurate system clock such as a TCXO. If the SRTC timer needs a correction, a 5-bit 2-second complement calibration word can be set to compensate the SRTC for inaccuracy in its reference oscillator as defined in the following table.The available correction range should be sufficient to ensure drift accuracy in compliance with standards for secure time applications like DRM time keeping. Note that the 32.768 kHz oscillator is not affected by SCAL_LP settings; calibration is only applied to the SRTC time base counter. Therefore, the frequency at the clock output Low Frequency Reference Clock is not affected.

The SRTC system calibration is enabled by programming the SCALM_LP[1:0] for desired behavior by operational mode.A slight increase in consumption is seen when that calibration circuitry is activated. To minimize consumption and maximize lifetime when the SRTC system is maintained by the coin cell, the SRTC Calibration circuitry can be automatically disabled when main battery contact is lost or if it is so deeply discharged that SRTC power draw is switched to the coin cell (configured with SCALM_LP=01). Because of the low SRTC consumption, SRTC accuracy can be maintained through long periods with the application being shut down, even after the main battery has discharged.

However, calibration is only as good as the SCAL_LP data that has been provided, so occasional refreshing is recommended to ensure that any drift influencing environmental factors have not skewed the clock beyond desired tolerances.

Table 51-3. SRTC LP Calibration Setting

Calibration Settings Code in SCAL_LP[4:0	Correction in Counts per 32768	Relative Correction in ppm
01111	+15	+458
00011	+3	+92
00001	+1	+31
00000	0	0
11111	-1	-31
11101	-3	-92
10001	-15	-458
10000	-16	-488

See LP Control Register (SRTC_LPCR), for bit definition of SCAL_LP and SCALM_LP.

51.3.3.3 SRTC LP Secured Counter Alarm

The SRTC LP section has its own 32-bit time alarm register. An alarm is generated when a 32-bit MSB (bits 46 to 15) value of the LP secure counter (SRTC_LPSCMR) matches with the 32-bit secure alarm register (SRTC_LPSAR). If X is the value at which the alarm is desired, then this SRTC_LPSAR should be set with the value X-1. The alarm register can be updated only by secure application. The SRTC LP time alarm can generate interrupt to alert the core and can wake-up the core from one of its low-power modes (for example, wait, doze, stop). This alarm can also wake-up the entire system by asserting the DO_ALARM_OUT signal to the PMIC. DO_ALARM_OUT is asserted only if the main system is powered off as indicated by power fail circuity or external signal (dsm_on) input to DO_ALARM_OUT is active-low .

The secured alarm has been provided with a separate mask bit, ALP and WAE, in the SRTC_LPCR .

51.3.3.4 Monotonic Counter

Security applications require a monotonic counter that cannot be exhausted or return to any previous value during the life-time of the product.

The monotonic counter is a counter that counts in one direction only. The monotonic counter can never repeat a number implying that it cannot be reset or cycled back to its starting count. The monotonic counter incorporates an innovative mechanism that prevents exhaustive usage of the counter while still allowing short rapid counter increments.

The monotonic counter features are listed below:

- 32-bits counter that can only go up
- Counter increment can only be done by secured SW (by writing any value to the counter's register)
- Average number of updates per predefined period is limited

51.3.3.4.1 Monotonic Counter Roll-Over Protection Mechanism

To prevent a case where a malicious SW rolls over the monotonic counter, a counter update limitation mechanism is implemented. This mechanism assures that the number of updates within a period of 512 seconds is limited. Updates beyond the specified number are ignored and the counter is not incremented.

The number of updates per 512 seconds is set by the IC Identification Module (e-fuse programmed) as follows: (number of updates (average updates per second))

- 8192 (16)
- 4096 (8)
- 2048 (4)
- 1024 (2)
- 512 (1)
- Unlimited

The mechanism gates the increment request signal with output overflow signal of a special 13-bit counter, as shown in the figure below.

Table 51-4 shows the mapping between the IC Identification Module throttle values and number of updates.

Table 51-4. Throttle and Monotonic Counter Updates Mapping

Value of IIM Throttle	Number of Updates per 512 seconds (Average Updates per second)
000	512(1)
001	1024 (2)
010	2048 (4)
011	4096 (8)
100	8192 (16)

Table continues on the next page...

Table 51-4. Throttle and Monotonic Counter Updates Mapping (continued)

Value of IIM Throttle	Number of Updates per 512 seconds (Average Updates per second)
101	8192 (16)
110	8192 (16)
110	8192 (16)
111	Unlimited

On reaching the value of 0xffff_ffff, the monotonic counter does not roll over. The MR bit in SRTC_LPSR is set and SRTC enters failure state.

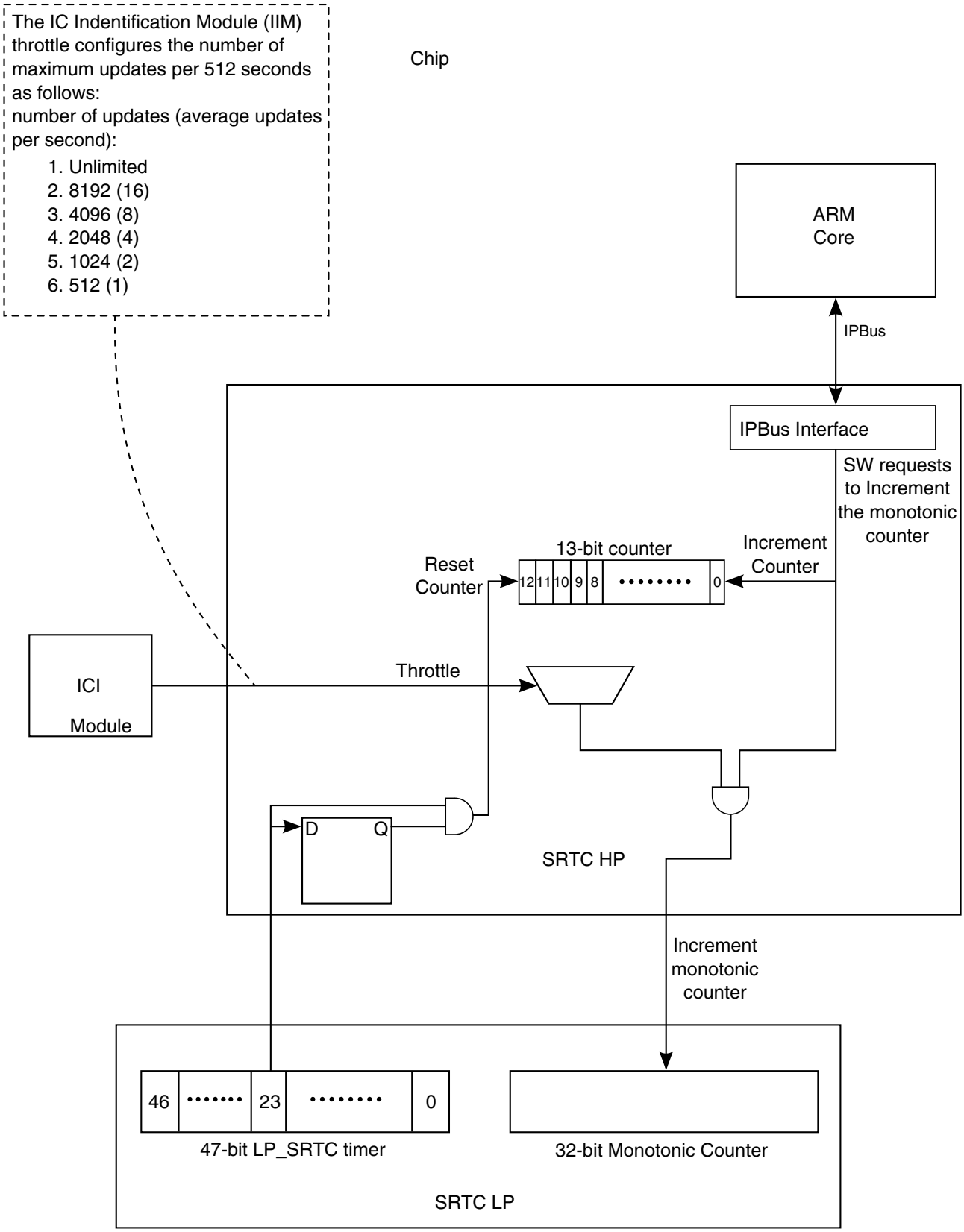


Figure 51-4. Monotonic Counter Roll-Over Protection Mechanism

51.3.3.5 General Purpose Always-Powered Registers

One 32-bits general purpose register that is a part of the SRTC LP and is always powered.

51.3.3.6 SRTC LP Monitor

The SRTC monitor watches for security alarms (SRTC LP and security block alarms) and then shifts the SRTC to failure state, in case of a security violation. The SRTC is comprised of three sub-blocks, SRTC State Machine (SSM), Power Supply Glitch Detector (PGD) and Clock Tampering Detector (CTD).

51.3.3.6.1 SRTC State Machine

The SRTC LP State Machine (SSM) can be found in one of four states:

- Initialization- SRTC failure indication can be cleared. Time read and monotonic counter read are reset and kept reset (except for nonsecure mode). Monotonic counter and secured time cannot be programmed. During this state the power glitch register can be loaded with the 41736166 value. This mode is exited by writing to a IE SRTC control bit. Failure monitors are not active in this state.
- Nonvalid- Time and monotonic counters are not valid and need to be set. Time is not running however time value and monotonic counter can be updated (in nonsecure mode, timer and monotonic counter are not reset and remained active). Failure detectors are active. This mode is exited by writing to a NVE SRTC control bit.
- Valid- Time and monotonic counter read are valid and operational. Re-programing is allowed only in appropriate security mode. Failure detectors are active.
- Failure- SRTC detected a security violation. Time and monotonic read are invalidated. Time and monotonic read operation returns zero (except for nonsecure mode where timer is operational, can be read and reprogrammed).

SRTC states are shown in Figure 51-5.

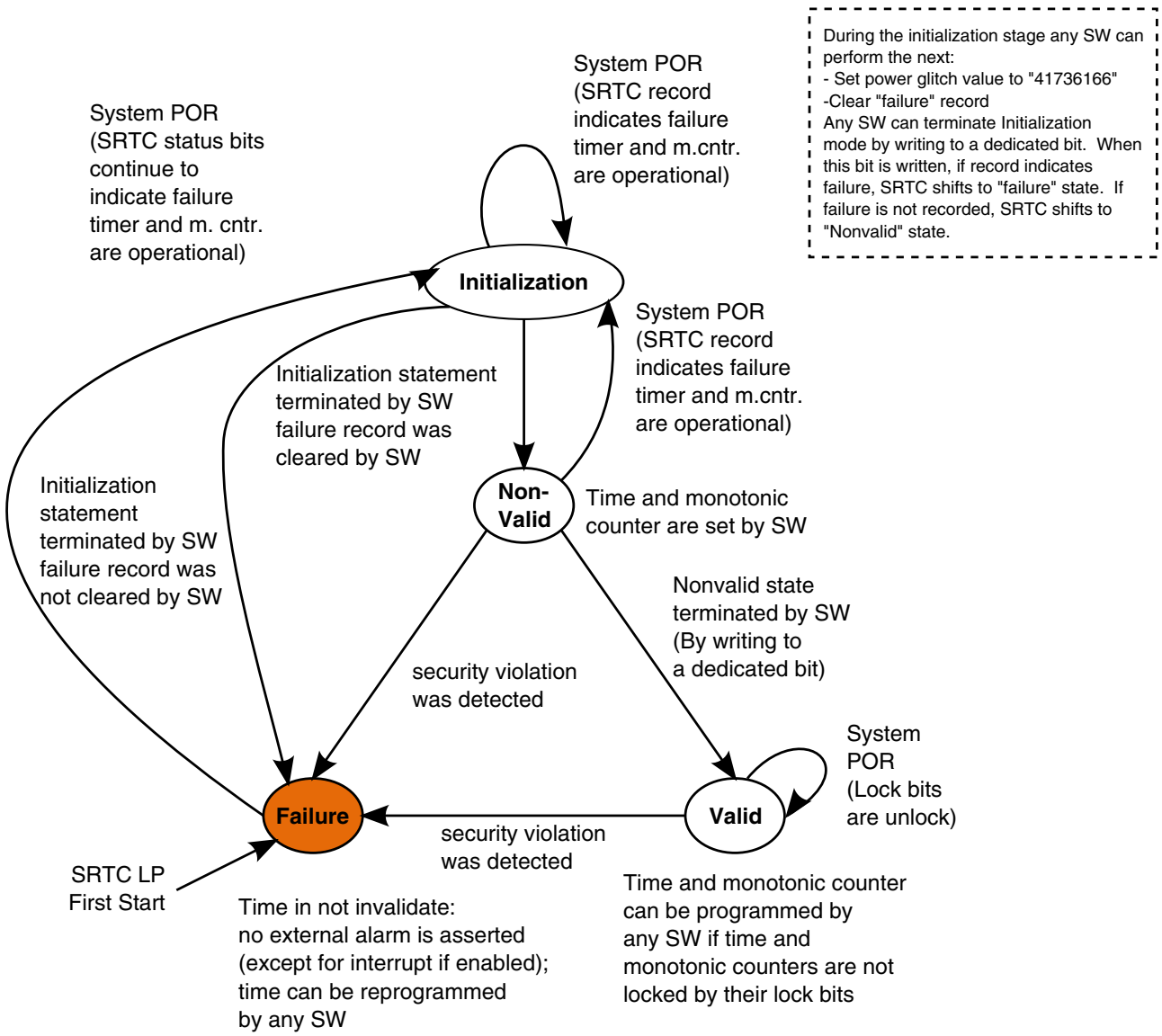


Figure 51-5. SRTC States - Low Security Mode

51.3.3.6.2 Power Supply Glitch Detector (PGD)

SRTC LP power supply can be power gated for short periods to cause state machine or secure counter register value to change. To detect such attack, SRTC incorporates an innovative method to detect and alert the system whenever power supply glitches endanger SRTC registers value. The SRTC glitch detector comprised of a simple register and few logic gates. This register is loaded to a known specific value (41736166) as part of the SRTC initialization process. This register have no reset input and its power-up value is pretty random. Comparing of these register value to a hardened value allow detection of power gating. The power supply glitch detector mechanism is shown in Figure 51-6.

The PGD and the TRI bit are set in the SRTC_LPSR and security alarm is raised to security block on SRTC security alarm (active high) pin and security interrupt (active low) to the core depending upon the SAE bit in the SRTC_LPCR and SI bit HPIENR respectively.

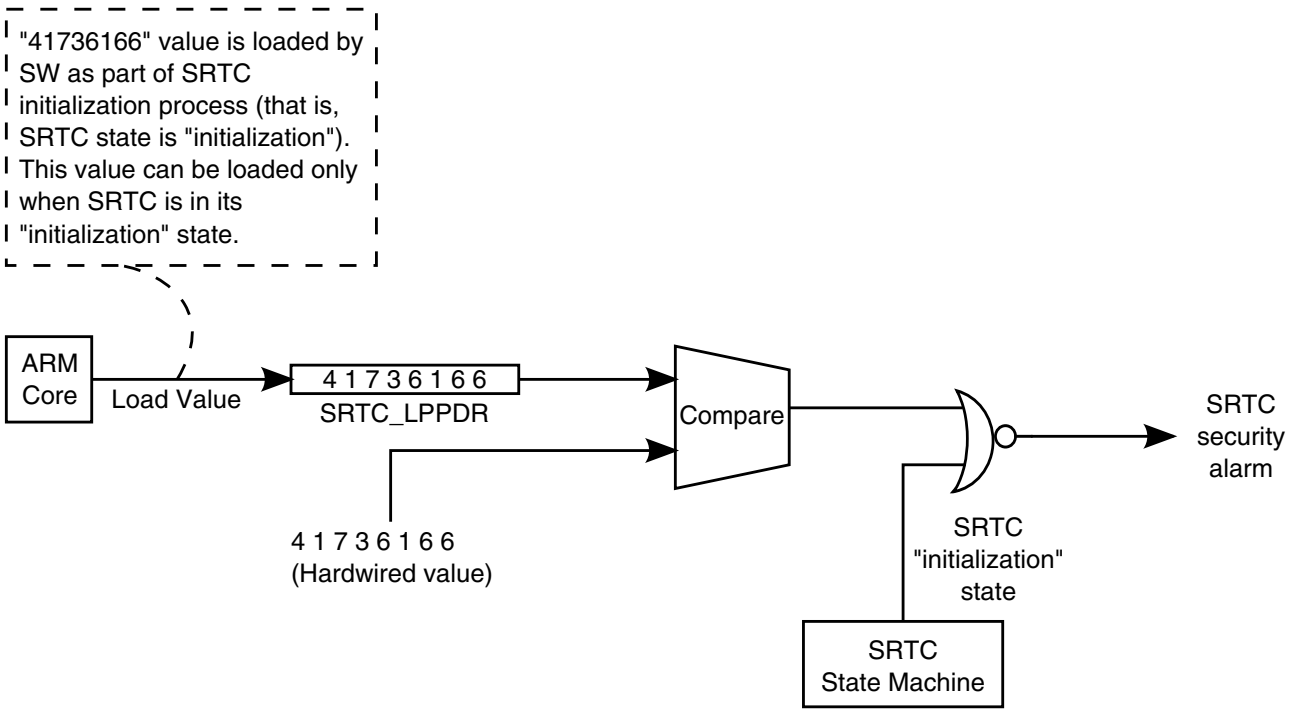


Figure 51-6. Power Supply Glitch Detector

51.3.3.6.3 Clock Tampering Detector (CTD)

External clock removal or frequency reduction can stop or slow down the secure counter time count. Because the clock signal is not generated internally in the SoC, SRTC cannot assure the reliability of the clock source and as consequence, cannot provide absolute assurance of the time read. However, SRTC provides means to detect significant reduction in its clock source frequency. SRTC clock tampering detector can detect frequency drop of 30%-100%. In current SRTC version, only clock frequency drop of 90% or more is detected.

If an attempt to tamper with the SRTC clock is detected, the SRTC enters failure state.

Storing fail indication of the clock monitor is done by SR latch with low, as possible, operating voltage. In this way, even if this register is not functional due to low voltage supply, on system power up, this register recalls the fail indication. The SRTC clock monitor mechanism is shown in Figure 51-7.

The CTD and the TRI bits are set in the SRTC_LPSR, and the security alarm is raised to the security block on the SRTC security alarm (active high) pin and the security interrupt (active low) to the core on the SRTC security Interrupt (Active low) pin SRTC security Interrupt (Active low) depending upon the SAE bit in the SRTC_LPCR and the SI bit in the SRTC_HPIENR, respectively.

51.3.3.6.4 Voltage Level Tampering Detector

Supply voltage of SRTC can be reduced to a value that causes the SRTC FF elements to be stuck at a constant value. Although clock frequency is valid, counter values can be stuck. Such attack has the same effect as clock input removal. Moreover, state machine registers can be also stuck, so even if violation is detected, it is not recorded. To make sure that voltage level is sufficient for proper functionality, the clock monitor does not monitor the raw clock input, but monitors a divided version of the actual real time counter. In this way, the same clock monitor can detect low voltage attack, where time counter LSB FF is stuck. LSB Bit of real time counter should be built from a high threshold transistors (this FF is more sensitive to low voltage than other SRTC FFs).

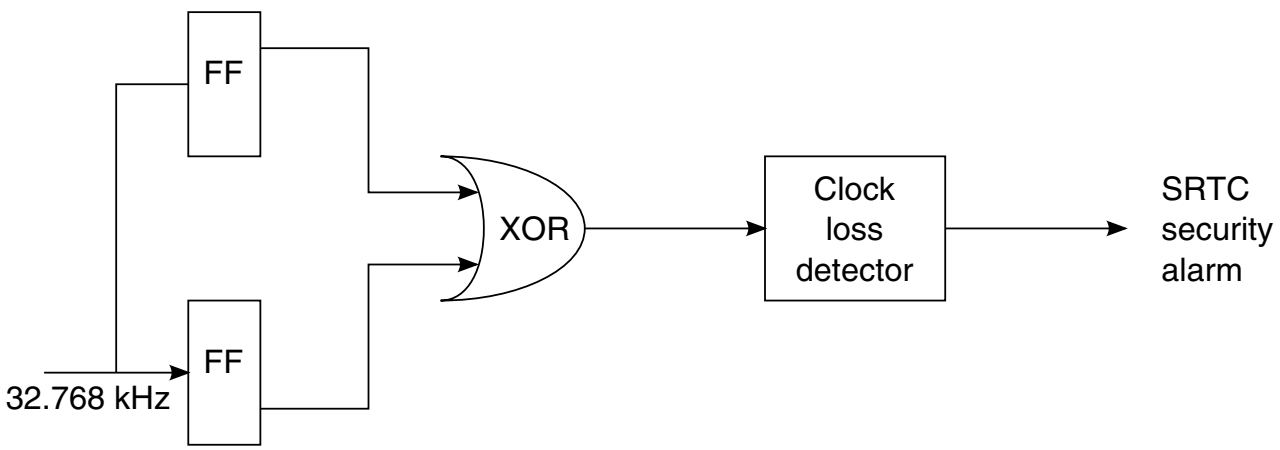


Figure 51-7. Clock Tampering Detector

51.3.3.6.5 Power Fail Detector (PFD)

Power fail detector is provided to detect the power failure in the SoG and isolate the SRTC LP from the rest of the SoG. It is provided with dsm_on input from the SoG. PFD is triggered when the dsm_on is asserted or when the there is power failure in the SoC.

51.4 SRTC Reset and System Power-Up

SRTC LP (always powered up section) is provided with its own POR, which provides power-on-reset as the power is switched on. This POR clears all registers inside SRTC.

SRTC LP section is in the Initialization state of the SRTC state machine upon its POR. Upon System POR the security mode of the SRTC is determined. If the security mode is Low (no security mode) then the SRTC LP counter and monotonic counter can be operational in the Initialization state.

Both the HP and LP section of the SRTC are provided with the their own software reset bit SWR in their respective control registers. The SWR bit of HP section can be set at any time and it resets all the HP section. This is a self clearing bit and always read as zero.

The SWR bit of LP section resets all the LP section. This bit can be set only in the Initialization state of SRTC state machine. This is a self clearing bit and always read as zero.

51.5 SRTC Interrupts and Alarms

The SRTC provides following interrupt and alarm lines

- $\overline{\text{SRTC Interrupt (Active low)}}$: Consolidated (ORed) interrupt pin for the following SRTC interrupts.
  - Periodic interrupt from 1 Hz to 32768 Hz frequency
  - Alarm interrupt of the HP section, see SRTC HP nonsecured Counter Alarm, for more details.
  - Alarm interrupt of the LP section, see SRTC LP Secured Counter Alarm, for more details.
  - Initialization Exit and Non Valid Exit Interrupt. Indicates the exit from the Initialization and Nonvalid state of the SRTC LP state machine.
  - Write done interrupt of the LP and HP section. To take care of the asynchronous Low Frequency Reference Clock, all the write from IP domain is synchronized to the Low Frequency Reference Clock domain. To indicate this delay the write done interrupt are provided for both the LP and HP section. Write done indicates that the consolidated status of the various IP writes and is triggered when all the registers have been updated. A separate interrupt is provided for the IP write to LP and HP section.
- SRTC security interrupt (Active low)- Security interrupt line to the core upon setting of the TRI bit in the LP section.

- SRTC security alarm- Security alarm to the security block upon setting of the TRI bit in the LP section.
- DO_ALARM_OUT- Time alarm of the LP section to the external pad, see SRTC LP Secured Counter Alarm, for more details. This alarm is generated when the system is powered off as indicated by the power fail circuitry or by external signal dsm_on input to the block. DO_ALARM_OUT is active low .

51.6 Initialization Information/Application Information

51.6.1 Flow Chart of SRTC LP Operation

See Figure 51-8 for the illustration of the flow chart of a typical SRTC LP operation.

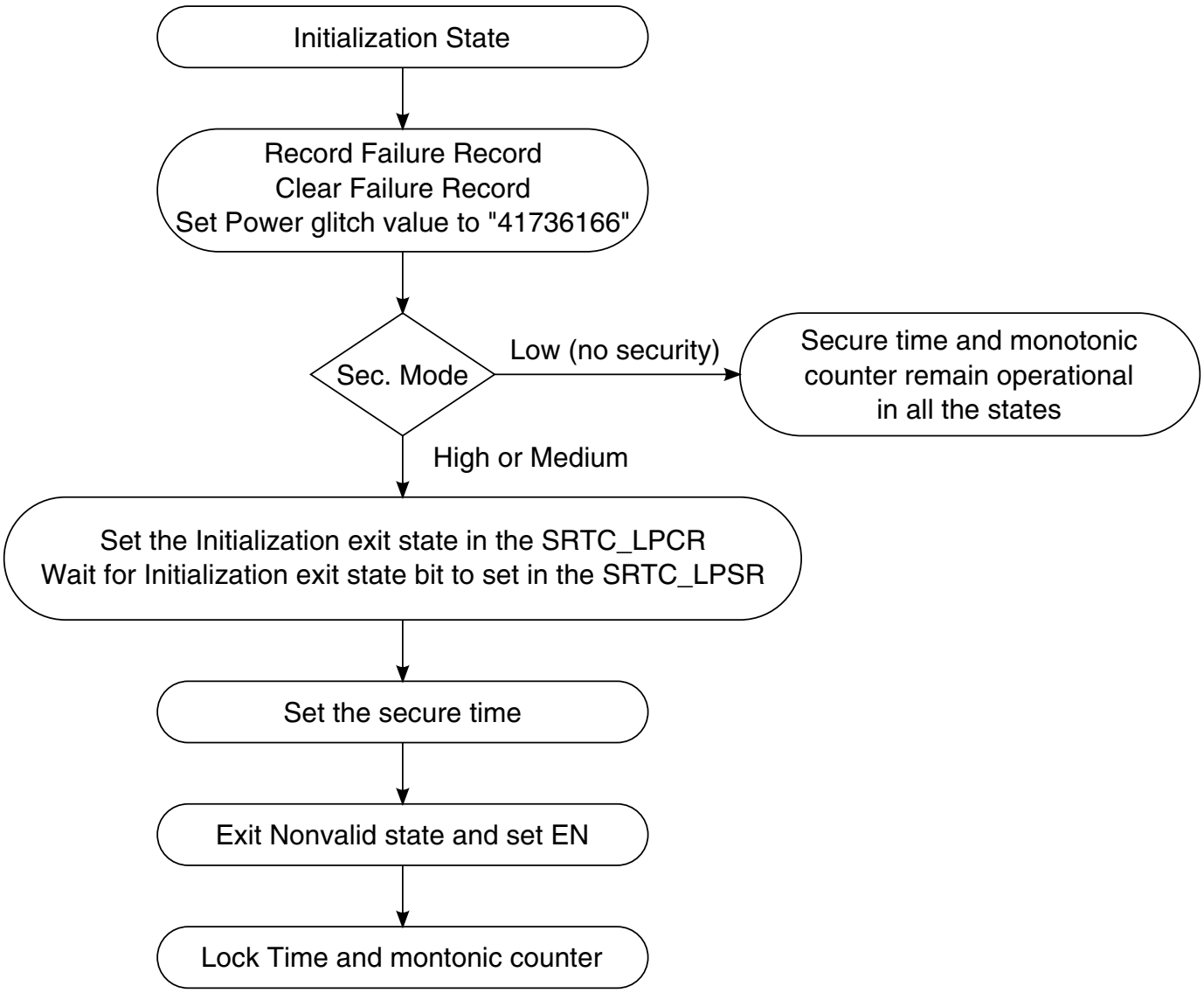


Figure 51-8. Flow Chart of SRTC LP Operations

51.6.2 Flow Chart of SRTC HP Operation

See Figure 51-9 for the illustration of the flow chart of a typical SRTC HP operation.

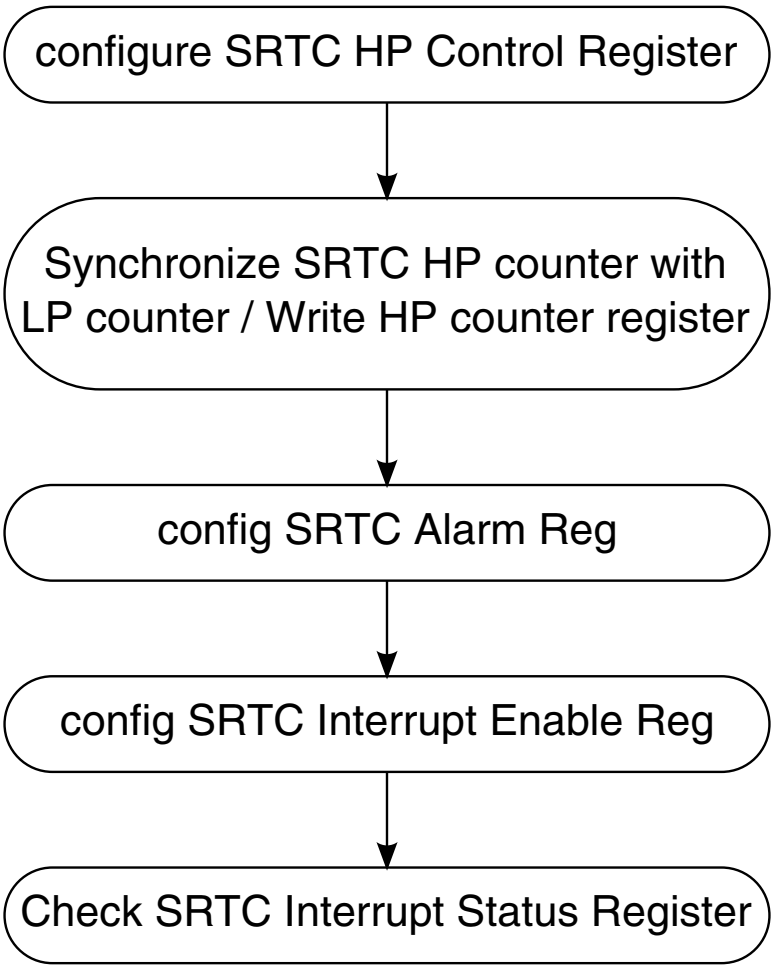


Figure 51-9. Flow Chart of SRTC HP Operations

51.7 Software Restrictions

The SRTC has the following software restrictions.

- The 32-bit MSB part and 15-bit LSB part of all the 47-bit registers should be updated together.
- The registers are not byte-writeable. All Valid byte enable should be asserted high for updating the register.
- LTC bit in the SRTC_LPCR. After it is set, this bit prevents write access to 47 bit SRTC LP secure counter by all application. Once set this bit is reset only upon system POR.

- LMC bit in the SRTC_LPCR. Once set this bit prevents write access to 32 bit SRTC LP secure monotonic counter by all application.Once set this bit is reset only upon system POR.
- SWR bit in the SRTC_LPCR. This bit can set by the secure application only in the Initialization state of the SRTC state machine.
- CTD, PTD and TRI bits in the SRTC_LPSR are unaffected by the System POR and are cleared only in Initialization state of SRTC state machine.
- SRTC LP power supply glitch detector register (SRTC_LPPDR). This register is write accessible only in the Initialization state of the SRTC state machine.
- Security mode. Security mode are retained upon system POR and can only be updated to higher security mode
- HPCR_TS that is, time synchronize operation and write on SRTC_HPCLR and SRTC_HPCMR shall not be performed at the same time. There is one count difference between LP and HP count value after TS has been performed
- There are two Low Frequency Reference Clock delay between the IP write operation on 47 bit counter, and value is updated on these counter.
- For write operation on monotonic counter the SRTC EN shall be enabled by setting the EN_LP. The software then shall wait for write pending LP bit (WPLP in SRTC_HPISR) to be zero before any write operation on Monotonic counter.
- IE and NVE in SRTC_LPCR are used to end the Initialization and Nonvalid states, respectively. They are used in the same sequence, that is, Initialization ends first, followed by Nonvalid exit.

When reading either SRTC_LPSCLR or SRTC_LPSCMR register, the software will read twice. The value of the register is valid when both the read data matches. In case there is mismatch the software will again read it twice.

51.8 Programmable Registers

The SRTC has fifteen 32-bit registers.

SRTC registers are marked with User and Secure access permissions. The distinction between User and Secure accesses does not apply when SRTC is configured for Low Security or the NSA bit is set in the SRTC_LPCR register.

- Supervisor mode.

SRTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FA_4000	LP Secure Counter MSB Register (SRTC_LPSCMR)	32	R/W	Unaffected	51.8.1/ 2894
53FA_4004	LP Secure Counter LSB Register (SRTC_LPSCLR)	32	R/W	See section	51.8.2/ 2894
53FA_4008	LP Secure Alarm Register (SRTC_LPSAR)	32	R/W	Unaffected	51.8.3/ 2895
53FA_400C	LP Secure Monotonic Counter Register (SRTC_LPSMCR)	32	R/W	Unaffected	51.8.4/ 2895
53FA_4010	LP Control Register (SRTC_LPCR)	32	R/W	0000_0000h	51.8.5/ 2896
53FA_4014	LP Status Register (SRTC_LPSR)	32	w1c	0000_0000h	51.8.6/ 2899
53FA_4018	LP Power Supply Glitch Detector Register (SRTC_LPPDR)	32	R/W	Unaffected	51.8.7/ 2902
53FA_401C	LP General Purpose Register (SRTC_LPGR)	32	R/W	Unaffected	51.8.8/ 2902
53FA_4020	HP Counter MSB Register (SRTC_HPCMR)	32	R/W	0000_0000h	51.8.9/ 2903
53FA_4024	HP Counter LSB Register (SRTC_HPCLR)	32	R/W	0000_0000h	51.8.10/ 2904
53FA_4028	HP Alarm MSB Register (SRTC_HPAMR)	32	R/W	0000_0000h	51.8.11/ 2904
53FA_402C	HP Alarm LSB Register (SRTC_HPALR)	32	R/W	0000_0000h	51.8.12/ 2905
53FA_4030	HP Control Register (SRTC_HPCR)	32	R/W	0000_0008h	51.8.13/ 2905
53FA_4034	HP Interrupt Status Register (SRTC_HPISR)	32	w1c	0000_0000h	51.8.14/ 2907
53FA_4038	HP Interrupt Enable Register (SRTC_HPIENR)	32	R/W	0000_0000h	51.8.15/ 2910

51.8.1 LP Secure Counter MSB Register (SRTC_LPSCMR)

The LP secure counter MSB register contains the 32 most significant bits (MSB bits 46 to 15) of the 47-bit LP secure counter.

Address: SRTC_LPSCMR is 53FA_4000h base + 0h offset = 53FA_4000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MLPSC																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

SRTC_LPSCMR field descriptions

Field	Description
31–0 MLPSC	MSB value of the LP secure counter Contains MSB bits 46 to 15 of the 47 Bit LP secure counter.

51.8.2 LP Secure Counter LSB Register (SRTC_LPSCLR)

The LP secure counter LSB register contains the 15 least significant bits LSB (bits 14 to 0) of the 47-bit LP secure counter.

Address: SRTC_LPSCLR is 53FA_4000h base + 4h offset = 53FA_4004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LLPSC																0															
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- u = Unaffected by reset.

SRTC_LPSCLR field descriptions

Field	Description
31–17 LLPSC	LSB value of the LP secure counter Contains LSB bits 14 to 0 of 47 Bit LP secure counter.
16–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

51.8.3 LP Secure Alarm Register (SRTC_LPSAR)

The LP secure alarm register contains the 32-bit value of LP section secure alarm register.

Address: SRTC_LPSAR is 53FA_4000h base + 8h offset = 53FA_4008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LPSALM																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

SRTC_LPSAR field descriptions

Field	Description
31–0 LPSALM	LP Secure alarm Contains 32 bit value of LP secure alarm register, see SRTC LP Secured Counter Alarm, for more details.

51.8.4 LP Secure Monotonic Counter Register (SRTC_LPSMCR)

The LP secure monotonic counter register contains the 32-bit value of LP section monotonic counter. The counter is increment by writing any value to the counters register.

Address: SRTC_LPSMCR is 53FA_4000h base + Ch offset = 53FA_400Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LPSMC																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

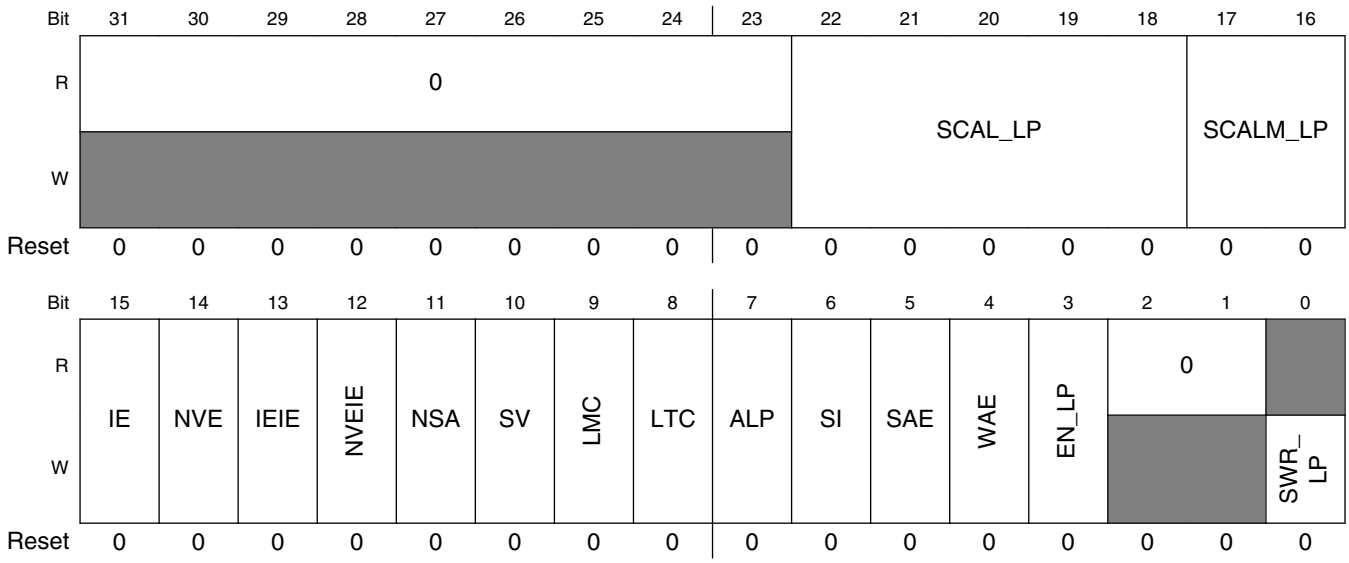
SRTC_LPSMCR field descriptions

Field	Description
31–0 LPSMC	LP Secure Monotonic counter Contains 32 bit value of LP secure monotonic counter.

51.8.5 LP Control Register (SRTC_LPCR)

The LP control register (SRTC_LPCR) contains various control bits of the LP section of SRTC.

Address: SRTC_LPCR is 53FA_4000h base + 10h offset = 53FA_4010h



SRTC_LPCR field descriptions

Field	Description
31–23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22–18 SCAL_LP	SRTC Calibration Value These five bits indicates signed calibration value for SRTC LP timer. In this, SCAL_LP[21:18] represent 4 bit value and SCAL_LP[22] represents the sign. Hence, SCAL_LP[22:18] allow calibration +15 to -16 in SRTC LP timer.
17–16 SCALM_LP	SRTCCALMODEL_LP Decides the SRTC LP calibration mode. 00 SRTC LP Timer calibration disabled. 01 SRTC LP Timer calibration enabled in all modes except coin cell mode 10 SRTC LP calibration disabled (Reserved). 11 SRTC LP Calibration enabled in all modes.
15 IE	Initialization state exit bit When set, this bit ends Initialization state of the SRTC state machine. **NOTE:** IE bit is a control bit to end the Initialization state. For the current state of SRTC LP see the state_lp field in the SRTC_LPSR. **NOTE:** This bit is unaffected by reset.

Table continues on the next page...

SRTC_LPCR field descriptions (continued)

Field	Description
	0 SRTC is in Initialization state. 1 End of Initialization state
14 NVE	Nonvalid state exit bit When set this bit ends the Nonvalid state of the SRTC state machine. **NOTE:** NVE bit is a control bit to end the Non Valid state. For the current state of SRTC LP see the state_lp field in the SRTC_LPSR. **NOTE:** This bit is unaffected by reset. 0 SRTC is in Nonvalid state. 1 End of Nonvalid state.
13 IEIE	Initialization state exit interrupt enable bit This bit enables the interrupt on exit from the Initialization State of the SRTC state machine. **NOTE:** This bit is unaffected by reset. 0 An interrupt is disabled. 1 An interrupt is enabled.
12 NVEIE	Nonvalid state exit interrupt enable bit This bit enables the interrupt on exit from the Nonvalid state of the SRTC state machine. **NOTE:** This bit is unaffected by reset. 0 An interrupt is disabled. 1 An interrupt is enabled.
11 NSA	Nonsecure access When set this bit allows nonsecure application to access secure register. This bit is reset by the system por. 0 Nonsecure access of secure register is not allowed 1 Nonsecure access is allowed
10 SV	Security Violation When set this bit move the SRTC LP to failure state upon termination of Initialization state. Also if the current state is Nonvalid/Valid the SRTC LP is moved to failure state. This bit is reset by the system por. **NOTE:** Setting the SV invalidates the SRTC_LPSCR and SRTC_LPSMCR contents. 0 No security violation 1 Security Violation
9 LMC	Lock monotonic counter When set this bit prevents write access to 32-bit secure LP monotonic counter register (SRTC_LPSMCR) by all application.Once set this bit cannot be reset except by system por. 0 Write access is allowed. 1 Write access is not allowed
8 LTC	Lock time counter

Table continues on the next page...

SRTC_LPCR field descriptions (continued)

Field	Description
	When set this bit prevents write access to secure 47-bit LP counter (SRTC_LPSCMR and SRTC_LPSCLR) by all application.Once set this bit cannot be reset except by system por. 0 Write access is allowed. 1 Write access is not allowed
7 ALP	Alarm Flag of LP Section Enables the alarm interrupt to the core. See SRTC LP Secured Counter Alarm, for more details. **NOTE:** This bit is unaffected by reset. 0 Alarm interrupt disabled. 1 Alarm interrupt enabled.
6 SI	Security Interrupt Enable bit This bit enables the security interrupt to be raised to core on SRTC security Interrupt (Active low). Security interrupt is raised upon setting of the of time read invalidate status bit **NOTE:** This bit is unaffected by reset. 0 Security interrupt disabled. 1 Security interrupt enabled
5 SAE	Security Alarm Enable bit This bit enables the security alarm to be raised to security block on SRTC security alarm (Active high) pin. Security alarm is raised upon setting of the of time read invalidate status bit **NOTE:** This bit is unaffected by reset. 0 Security alarm disabled. 1 Security alarm enabled
4 WAE	Wakeup Alarm Enable bit This bit enables the time alarm to be raised on DO_ALARM_OUT pin. See SRTC LP Secured Counter Alarm, for more details.DO_ALARM_OUT is asserted only if the main system is powered off as indicated by power fail circuity or external signal (dsm_on) input to block. DO_ALARM_OUT's polarity is active low **NOTE:** This bit is unaffected by reset. 0 Time alarm disabled. 1 Time alarm enabled.
3 EN_LP	Enables/Disables the LP secure time and secure monotonic counter When set the secure LP counter and monotonic counter becomes operational. This bit cannot be reset once either of LMC or LTC is set. In high and Medium this bit can only be set in Valid state **NOTE:** This bit is unaffected by reset. 0 Disable the real-time clock and monotonic counter 1 Enable the real-time clock and monotonic counter
2–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 SWR_LP	Software reset

Table continues on the next page...

SRTC_LPCR field descriptions (continued)

Field	Description
	Resets the LP section of the SRTC to its default state.This bit can be set only in the Initialization state of the SRTC state machine. It can be set by a secure application only. This is self clearing bit is always read as zero. 0 No effect 1 Reset the block to its default state

51.8.6 LP Status Register (SRTC_LPSR)

The LP status register (LPSR) register is used by the core to read the status of the various bits.

Address: SRTC_LPSR is 53FA_4000h base + 14h offset = 53FA_4014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IES	NVES	STATE_LP		SM		IT		EAD	TR	MR	ALP	CTD	PGD	TRI	
W	w1c	w1c							w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRTC_LPSR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 IES	Initialization state exit status bit This bit indicates the end of Initialization state of the SRTC state machine. **NOTE:** IES and NVES are w1c status bits corresponding to the Initialization Exit and Nonvalid exit interrupts. **NOTE:** This bit is unaffected by reset. 0 SRTC is in Initialization state. 1 SRTC has exit out of Initialization state
14 NVES	Nonvalid state exit status bit This bit indicate the end of Nonvalid state of the SRTC state machine. **NOTE:** This bit is unaffected by reset. 0 SRTC is in Nonvalid state. 1 SRTC has exit out of Nonvalid state.

Table continues on the next page...

SRTC_LPSR field descriptions (continued)

Field	Description
13–12 STATE_LP	STATE_LP Specify the present state of SRTC LP state machine. These are read-only bits. **NOTE:** This bit is unaffected by reset. 00 Initialize. 01 Nonvalid 10 Valid 11 Failure
11–10 SM	Security Mode Specify the present security mode of the SRTC as determined from the IC Identification Module fuses. These are read-only bits. **NOTE:** This bit is unaffected by reset. 00 Security Mode: Low (no security). Other values: see Security Reference Manual .
9–7 IT	IIM Throttle Specify the 3-bit value of the throttle bits for monotonic counter from the IC Identification Module. These are read-only bits. SeeTable 51-4. **NOTE:** This bit is unaffected by reset.
6 EAD	External alarm detected Indicates that Security alarm generated by the external security block has been detected or external boot is performed in High and Medium security mode. **NOTE:** This bit is unaffected by reset. 0 No security alarm detected. 1 Security alarm detected
5 TR	Time Rollover Indicates that the 47 Bit LP secure counter has reached the value of 0xffff_ffff_ffff. When counter reaches this maximum value, TRI bit is also set and SRTC enters failure state. Once set this bit is not reset by system por, and can be cleared only by secure application in Initialization state of SRTC state machine. **NOTE:** This bit is unaffected by reset. 0 Counter has not reached its maximum value. 1 Counter has reached its maximum value.
4 MR	Monotonic counter Rollover Indicates that the 32 Bit LP secure monotonic counter has reached the value of 0xffff_ffff. When counter reaches this maximum value, TRI bit is also set and SRTC enters failure state. Once set this bit is not reset by system por, and can be cleared only by secure application in Initialization state of SRTC state machine. **NOTE:** This bit is unaffected by reset.

Table continues on the next page...

SRTC_LPSR field descriptions (continued)

Field	Description
	0 Counter has not reached its maximum value. 1 Counter has reached its maximum value
3 ALP	Alarm Flag of LP section Indicates that the SRTC LP secured counter alarm has occured. See SRTC LP Secured Counter Alarm, for more details. **NOTE:** This bit is unaffected by reset. 0 No alarm occurred. 1 An alarm has occurred.
2 CTD	Clock Tampering Detected This bit is set when clock tampering is detected. Once set this bit is not reset by system por, and can be cleared only by secure application in Initialization state of SRTC state machine **NOTE:** This bit is unaffected by reset. 0 No Clock Tampering Detected. 1 Clock Tampering Detected.
1 PGD	Power supply glitch Detected This bit is set when glitch in the power supply is detected. Once set this bit is not reset by system por, and can be cleared only by secure application in Initialization state of SRTC state machine. **NOTE:** This bit is unaffected by reset. 0 No Power supply glitch Detected. 1 Power supply glitch Detected.
0 TRI	Time Read Invalidate This bit is set when SRTC LP state machine is moved to failure by any of the security violation. It indicates the time maintained by the SRTC has been invalidated.Once set this bit is not reset by system por, and can be cleared only by secure application in Initialization state of SRTC state machine. **NOTE:** This bit is unaffected by reset. 0 Time read is validated. 1 Time read is invalidated.

51.8.7 LP Power Supply Glitch Detector Register (SRTC_LPPDR)

The LP power supply glitch detector register (SRTC_LPPDR) provides a 32-bit read write register, which is used for storing power glitch value logic as described in Power Supply Glitch Detector (PGD). This register may be written only in the Initialization state of the SRTC state machine.

Address: SRTC_LPPDR is 53FA_4000h base + 18h offset = 53FA_4018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LPPDR																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

SRTC_LPPDR field descriptions

Field	Description
31–0 LPPDR	LP power supply glitch detector register Contains 32 bit value of LP power supply glitch detector register.

51.8.8 LP General Purpose Register (SRTC_LPGR)

The LP general purpose register can be used to retain data while the system power is down.

NOTE

SW_ISO will not be set by any software except by the power fail routine. Once set it can only be reset by negedge of power fail detector, that is, by removing power from the SoC. This is provided as an additional safeguard over the Power fail detector's late triggering upon power failure in SoC.

NOTE

An alternate way to reset this bit is to assert dsm_on input.

Address: SRTC_LPGR is 53FA_4000h base + 1Ch offset = 53FA_401Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SW_ISO	SEC_BT	LPB_STS	LPGR																												
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

SRTC_LPGR field descriptions

Field	Description
31 SW_ISO	Software isolation bit When set it isolates the SRTC LP portion from the rest of the SoC. It is reset by the negedge of the power fail output of internal power fail detector. 0 SRTC is not isolated. 1 SRTC is isolated from the rest of the SoC.
30 SEC_BT	Secondary boot bit, ROM uses this bit to boot the media using redundant copy of firmware.
29 LPB_STS	This bit shows the boot state. 0 Normal boot mode. 1 Low power boot mode.
28–0 LPGR	LP general purpose register Contains user-specified data.

51.8.9 HP Counter MSB Register (SRTC_HPCMR)

The HP counter MSB register contains the 32 most significant bits (MSB bits 46 to 15) of the 47-bit HP counter.

Address: SRTC_HPCMR is 53FA_4000h base + 20h offset = 53FA_4020h

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

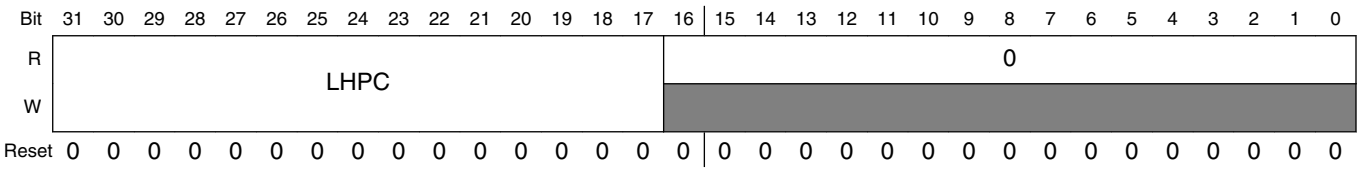
SRTC_HPCMR field descriptions

Field	Description
31–0 MHPC	MSB value of the HP counter Contains MSB bits 46 to 15 of the 47 Bit SRTC HP counter.

51.8.10 HP Counter LSB Register (SRTC_HPCLR)

The HP counter LSB register contains the 15 least significant bits LSB (bits 14 to 0) of the 47-bit SRTC HP counter.

Address: SRTC_HPCLR is 53FA_4000h base + 24h offset = 53FA_4024h



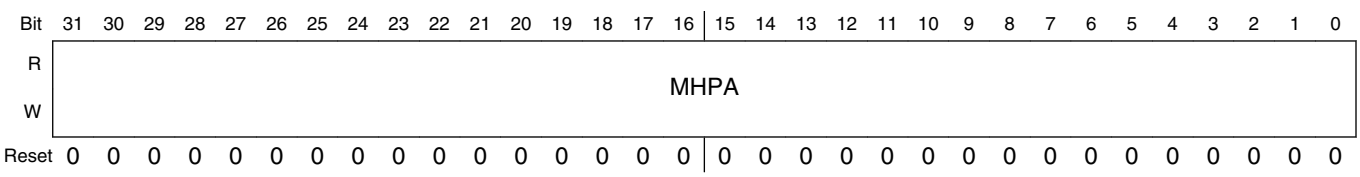
SRTC_HPCLR field descriptions

Field	Description
31–17 LHPC	LSB value of the HP counter Contains LSB bits 14 to 0 of 47 Bit SRTC HP counter.
16–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

51.8.11 HP Alarm MSB Register (SRTC_HPAMR)

The HP alarm MSB register contains the 32 most significant bits (MSB bits 46 to 15) of the 47-bit HP alarm.

Address: SRTC_HPAMR is 53FA_4000h base + 28h offset = 53FA_4028h



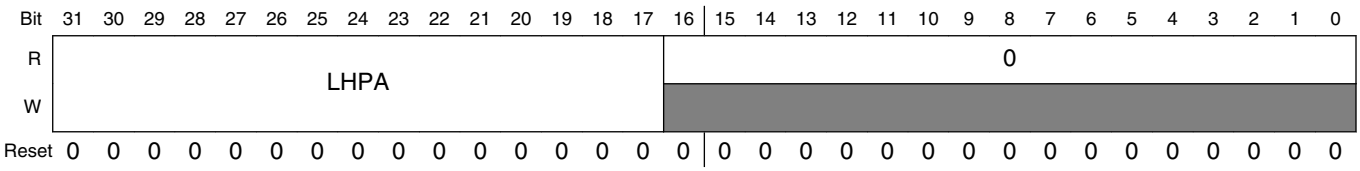
SRTC_HPAMR field descriptions

Field	Description
31–0 MHPA	MSB value of the HP alarm Contains MSB bits 46 to 15 of the 47 Bit SRTC HP alarm. See SRTC HP nonsecured Counter Alarm, for more details.

51.8.12 HP Alarm LSB Register (SRTC_HPALR)

The HP alarm LSB register contains the 15 least significant bits LSB (bits 14 to 0) of the 47-bit SRTC HP alarm.

Address: SRTC_HPALR is 53FA_4000h base + 2Ch offset = 53FA_402Ch



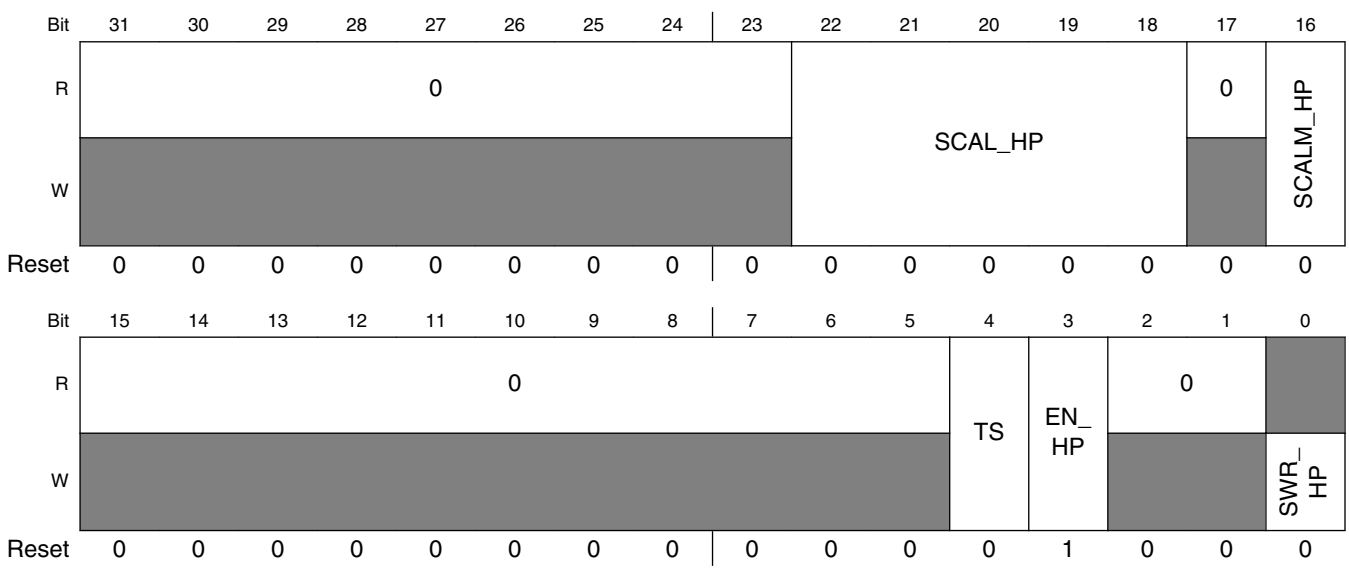
SRTC_HPALR field descriptions

Field	Description
31–17 LHPA	LSB value of the HP alarm Contains LSB bits 14 to 0 of 47 Bit SRTC HP alarm, see SRTC HP nonsecured Counter Alarm, for more details.
16–0 Reserved	This read-only field is reserved and always has the value zero. Reserved

51.8.13 HP Control Register (SRTC_HPCR)

The HP control register (SRTC_HPCR) contains various control bits of the HP section of SRTC.

Address: SRTC_HPCR is 53FA_4000h base + 30h offset = 53FA_4030h



SRTC_HPCR field descriptions

Field	Description
31–23 Reserved	This read-only field is reserved and always has the value zero. Reserved
22–18 SCAL_HP	SRTC Calibration Value These five bits indicates signed calibration value for SRTC HP timer. In this, SCAL_HP[21:18] represent 4 bit value and SCAL_HP[22] represents the sign. Hence, SCAL_HP[22:18] allow calibration +15 to -16 in SRTC HP timer.
17 Reserved	This read-only field is reserved and always has the value zero. Reserved
16 SCALM_HP	SRTCCALMODE_HP Decides the SRTC HP calibration mode. 0 SRTC HP Timer calibration disabled (default). 1 SRTC HP Timer calibration enabled
15–5 Reserved	This read-only field is reserved and always has the value zero. Reserved
4 TS	Time synchronize When set this updates the 47 bit HP counter with the 47 bit LP secure counter. This bit remains set, while the counter is being synchronized, after which it self clears. 0 No Action. 1 Time is synchronized
3 EN_HP	Enables/Disables the HP Time counter 0 Disable the real-time clock 1 Enable the real-time clock
2–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 SWR_HP	Software reset Resets the HP Section of block to its default state. SWR self clears to zero when 0 No effect 1 Reset the block to its default state

51.8.14 HP Interrupt Status Register (SRTC_HPISR)

The HP interrupt status register (SRTC_HPISR) indicates the status of the various periodic, alarm and status interrupts. When an event of the types included in this register occurs, then the bit is set in this register regardless of its corresponding interrupt enable bit. Interrupts may occur while the system clock is idle or in sleep mode. Every interrupt status bit is independent of each other. See SRTC Interrupts and Alarms, for more information on different interrupts.

Address: SRTC_HPISR is 53FA_4000h base + 34h offset = 53FA_4034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										WPLP	WPHP	WDLP	WDHP	0	AHP
W													w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PI15	PI14	PI13	PI12	PI11	PI10	PI9	PI8	PI7	PI6	PI5	PI4	PI3	PI2	PI1	PI0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRTC_HPISR field descriptions

Field	Description
31–22 Reserved	This read-only field is reserved and always has the value zero. Reserved
21 WPLP	Write pending LP Indicates that the write request are pending in the LP section. This is read-only bit. 0 No write requests are pending. 1 Write requests are pending.
20 WPHP	Write pending HP Indicates that the write request are pending in the HP section bit. This is read-only bit. 0 No write requests are pending 1 Write requests are pending
19 WDLP	Write Done LP Indicates that the write request are completed in the LP section. This is w1c bit.

Table continues on the next page...

SRTC_HPISR field descriptions (continued)

Field	Description
	NOTE: The information conveyed by the WPLP and WDLP is about delayed write operation on LP section due to synhronization. WPLP is active high read only bit which indica.tes a pending write operation on LP. WDLP is w1c status bit which is set when the write operation on the LP is completed 0 Write requests are pending. 1 Write requests are completed.
18 WDHP	Write Done HP Indicates that the write request are completed in the HP section. This is w1c bit. **NOTE:** The information conveyed by the WPHP and WDHP is about delayed write operation on HP section due to synhronization. WPHP is active high read only bit which indicates a pending write operation on HP. WDHP is w1c status bit which is set when the write operation on the HP is completed. 0 Write requests are pending. 1 Write requests are completed.
17 Reserved	This read-only field is reserved and always has the value zero. Reserved
16 AHP	Alarm Flag of HP section Indicates that the HP Nonsecured counter alarm has occurred, see SRTC HP nonsecured Counter Alarm, for more details. 0 No alarm interrupt occurred. 1 An alarm interrupt has occurred.
15 PI15	Periodic Interrupt Flag at 32768 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 32768 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
14 PI14	Periodic Interrupt Flag at 16384 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 16384 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
13 PI13	Periodic Interrupt Flag at 8192 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 8192 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
12 PI12	Periodic Interrupt Flag at 4096 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 4096 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
11 PI11	Periodic Interrupt Flag at 2048 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 2048 Hz.

Table continues on the next page...

SRTC_HPISR field descriptions (continued)

Field	Description
	0 No interrupt occurred. 1 An interrupt occurred.
10 PI10	Periodic Interrupt Flag at 1024 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 1024 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
9 PI9	Periodic Interrupt Flag at 512 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 512 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
8 PI8	Periodic Interrupt Flag at 256 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 256 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
7 PI7	Periodic Interrupt Flag at 128 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 128 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
6 PI6	Periodic Interrupt Flag at 64 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 64 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
5 PI5	Periodic Interrupt Flag at 32 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 32 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
4 PI4	Periodic Interrupt Flag at 16 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 16 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
3 PI3	Periodic Interrupt Flag at 8 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 8 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
2 PI2	Periodic Interrupt Flag at 4 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 4 Hz.

Table continues on the next page...

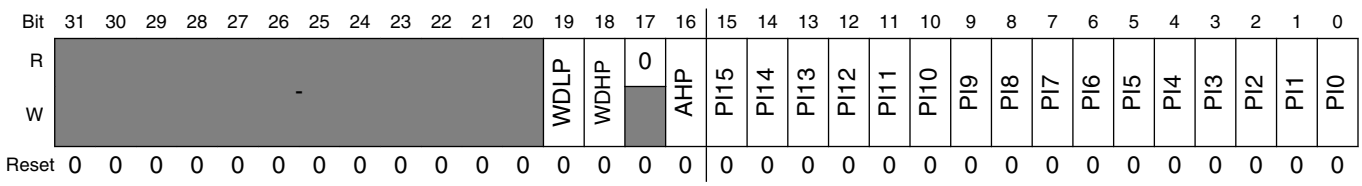
SRTC_HPISR field descriptions (continued)

Field	Description
	0 No interrupt occurred. 1 An interrupt occurred.
1 PI1	Periodic Interrupt Flag at 2 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 2 Hz. 0 No interrupt occurred. 1 An interrupt occurred.
0 PI0	Periodic Interrupt Flag at 1 Hz Frequency Indicates that an interrupt has occurred. If enabled, this bit is periodically set at a rate of 1 Hz. 0 No interrupt occurred. 1 An interrupt occurred.

51.8.15 HP Interrupt Enable Register (SRTC_HPIENR)

The HP interrupt enable register (SRTC_HPIENR) is used to enable/disable the various periodic, alarm and security interrupts. Masking an interrupt bit has no effect on its corresponding status bit. Every interrupt enable bit is independent of the other.

Address: SRTC_HPIENR is 53FA_4000h base + 38h offset = 53FA_4038h



SRTC_HPIENR field descriptions

Field	Description
31–20 -	Reserved
19 WDLP	Write Done LP Enable/Disable the Write done interrupt of LP section. 0 An interrupt is disabled. 1 An interrupt is enabled.
18 WDHP	Write Done HP Enable/Disable the Write done interrupt of HP section. 0 An interrupt is disabled. 1 An interrupt is enabled.
17 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

SRTC_HPIENR field descriptions (continued)

Field	Description
16 AHP	Alarm Flag of HP section Enable/Disable the alarm interrupt of HP section, see SRTC HP nonsecured Counter Alarm, for more details. 0 An interrupt is disabled. 1 An interrupt is enabled.
15 PI15	Periodic Interrupt Flag at 32768 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 32768 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
14 PI14	Periodic Interrupt Flag at 16384 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 16384 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
13 PI13	Periodic Interrupt Flag at 8192 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 8192 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
12 PI12	Periodic Interrupt Flag at 4096 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 4096 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
11 PI11	Periodic Interrupt Flag at 2048 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 2048 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
10 PI10	Periodic Interrupt Flag at 1024 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 1024 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
9 PI9	Periodic Interrupt Flag at 512 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 512 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
8 PI8	Periodic Interrupt Flag at 256 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 256 Hz Frequency Interrupt.

Table continues on the next page...

SRTC_HPIENR field descriptions (continued)

Field	Description
	0 An interrupt is disabled. 1 An interrupt is enabled.
7 PI7	Periodic Interrupt Flag at 128 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 128 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
6 PI6	Periodic Interrupt Flag at 64 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 64 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
5 PI5	Periodic Interrupt Flag at 32 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 32 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
4 PI4	Periodic Interrupt Flag at 16 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 16 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
3 PI3	Periodic Interrupt Flag at 8 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 8 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
2 PI2	Periodic Interrupt Flag at 4Hz Frequency Enable/Disable the Periodic Interrupt Flag at 4 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
1 PI1	Periodic Interrupt Flag at 2 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 2 Hz Frequency Interrupt. 0 An interrupt is disabled. 1 An interrupt is enabled.
0 PI0	Periodic Interrupt Flag at 1 Hz Frequency Enable/Disable the Periodic Interrupt Flag at 1 Hz Frequency Interrupt 0 An interrupt is disabled. 1 An interrupt is enabled.

Chapter 52
Synchronous Serial Interface (SSI)

52.1 Overview

This block guide presents the Synchronous Serial Interface (SSI), and discusses the architecture, the programming model, the operating modes, and initialization of SSI.

The SSI is a full-duplex, serial port that allows the chip to communicate with a variety of serial devices. These serial devices can be standard CODer-DECoder (CODECs), Digital Signal Processors (DSPs), microprocessors, peripherals, and popular industry audio codecs that implement the inter-IC sound bus standard (I2S) and Intel AC97 standard.

SSI is typically used to transfer samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

The figure below illustrates the organization of the SSI. It consists of control registers to set up the port, status register, separate transmit and receive circuits with FIFO registers, and separate serial clock and frame sync generation for the transmit and receive sections. The second set of Tx and Rx FIFOs, replicates the logic used for the first set of FIFOs.

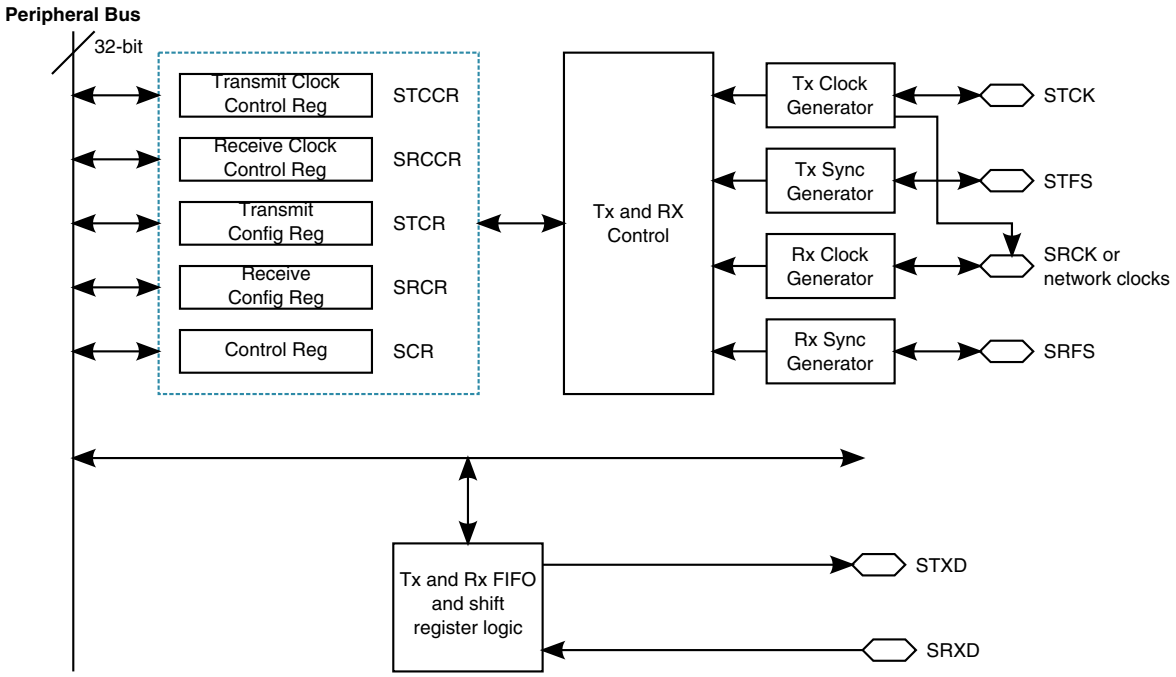


Figure 52-1. SSI Block Diagram

52.1.1 Features

The SSI includes the following features:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in Master or Slave mode.
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots
- Gated Clock mode operation requiring no frame sync
- 2 sets of Transmit and Receive FIFOs. Each of the four FIFOs is 15x32 bits. The two sets of Tx/Rx FIFOs can be used in Network mode to provide 2 independent channels for transmission and reception
- Programmable data interface modes such as I2S, LSB, MSB aligned
- Programmable word length (8, 10, 12, 16, 18, 20, 22 or 24 bits)
- Program options for frame sync and clock generation
- Programmable I2S modes (Master, Slave or Normal). Max audio sampling rate is 196kHz. Min audio sampling rate is 8kHz. Network clock (as an oversampling clock to external device) available as output from SRCK in I2S Master mode
- AC97 support. Max frame rate is 48kHz. Min frame rate is 8kHz.

- Completely separate clock and frame sync selections for the receive and transmit sections. In AC97 standard, the clock is taken from an external source and frame sync is generated internally.
- External network clock input for use in I2S Master mode. Programmable network clock of the sampling frequency available as output in master mode at SRCK, when operated in sync mode.
- Programmable internal clock divider
- Time Slot Mask Registers for reduced ARM platform overhead (for both Tx and Rx)
- SSI power-down feature
- Programmable wait states for ARM platform accesses

52.1.2 Modes of Operation

SSI has the following basic operating modes.

- Normal Mode : Asynchronous protocol, Synchronous protocol
  - Normal Mode Transmit
  - Normal Mode Receive
- Network Mode : Asynchronous protocol, Synchronous protocol
  - Network Mode Transmit
  - Network Mode Receive
- Gated Clock Mode : Synchronous protocol only
- I2S Mode
- AC97 Mode
  - AC97 Fixed Mode (SSI.SACNT[1]=0)
  - AC97 Variable Mode (SSI.SACNT[1]=1)

52.2 External Signal Description

52.2.1 Signals Overview

The Synchronous Serial Interface (SSI) can be connected directly to the external pins or through the Digital Audio Multiplexer (AUDMUX). Refer to the AUDMUX chapter for programming details of the various multiplexing options.

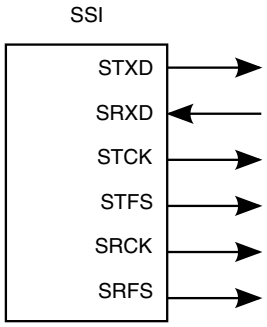
Table 52-1. Off-Chip Block Signals

Name	I/O	Function	Reset State	Pull up
SRCK	I/O	Serial Receive Clock. SRCK can be used as either an input or an output. This clock signal is used by the receiver in asynchronous mode and is always continuos. During synchronous mode, the STCK port is used instead for clocking in data. In SSI synchronous modes, this port can be used as an output for the network clock (oversampling clock) . In I2S master mode, this signal can be used to output the network clock to an external CODEC.	0	Passive
SRFS	I/O	Serial Receive Frame Sync. The SRFS port can be used as either an input or an output. The frame sync is used by the receiver to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. If SRFS is configured as an input, the external device should drive SRFS during the rising edge of STCK or SRCK.	0	Passive
SRXD	I	Serial Receive Data. The SRXD port is an input and is used to bring serial data into the Receive Data Shift Register.	-	-
STCK	I/O	Serial Transmit Clock. The STCK port can be used as either an input or an output. This clock signal is used by the transmitter and can be either continuous or gated. During Gated Clock mode, data on the STCK port is valid only during the transmission of data, otherwise it is pulled to the inactive state. In Synchronous mode, this port is used by both the transmit and receive sections.	0	Passive
STFS	I/O	Serial Transmit Frame Sync. The STFS port can be used as either an input or an output. The frame sync is used by the transmitter to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. In Synchronous mode, this port is used by both the transmit and receive sections. In Gated Clock mode, frame sync signals are not used. If STFS is configured as an input, the external device should drive STFS during the rising edge of STCK if TSCKP is +ve edge triggered. The external device should drive STFS during the falling edge of STCK if TSCKP is -ve edge triggered.	0	Passive
STXD	O	Serial Transmit Data. The STXD port is an output and transmits data from the Serial Transmit Shift Register. The STXD port is an output port when data is being transmitted and is disabled between data word transmissions and on the trailing edge of the bit clock after the last bit of a word is transmitted.	0	Passive

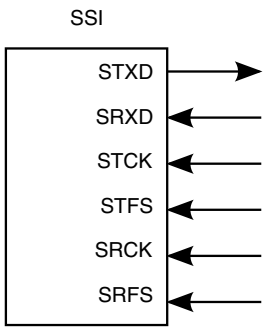
The following figure shows the main SSI configurations. These ports support all transmit and receive functions with continuous or gated clock as shown.

NOTE

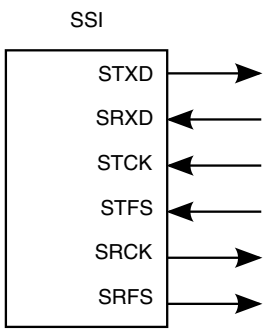
Gated clock implementations do not require the use of the frame sync ports (STFS and SRFS).



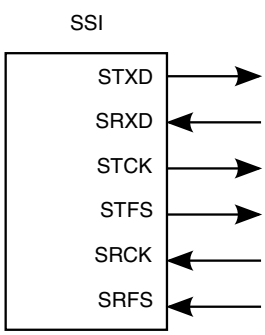
SSI Internal Continuous Clock for TX/RX (RXDIR = 1, TXDIR = 1, RFDIR = 1, TFDIR = 1, SYN = 0)



SSI External Continuous Clock for TX/RX (RXDIR = 0, TXDIR = 0, RFDIR = 0, TFDIR = 0, SYN = 0)



SSI Internal Continuous Clock for RX (RXDIR = 1, TXDIR = 0, RFDIR = 1, TFDIR = 0, SYN = 0)
SSI External Continuous Clock for TX



SSI Internal Continuous Clock for TX (RXDIR = 0, TXDIR = 1, RFDIR = 0, TFDIR = 1, SYN = 0)
SSI External Continuous Clock for RX

Figure 52-2. Asynchronous (SYN=0) SSI Configurations-Continuous Clock

See the following figure for an example of the port signals for an 8-bit data transfer. Continuous and gated clock signals are shown, as well as the bit-length frame sync signal and the word-length frame sync signal.

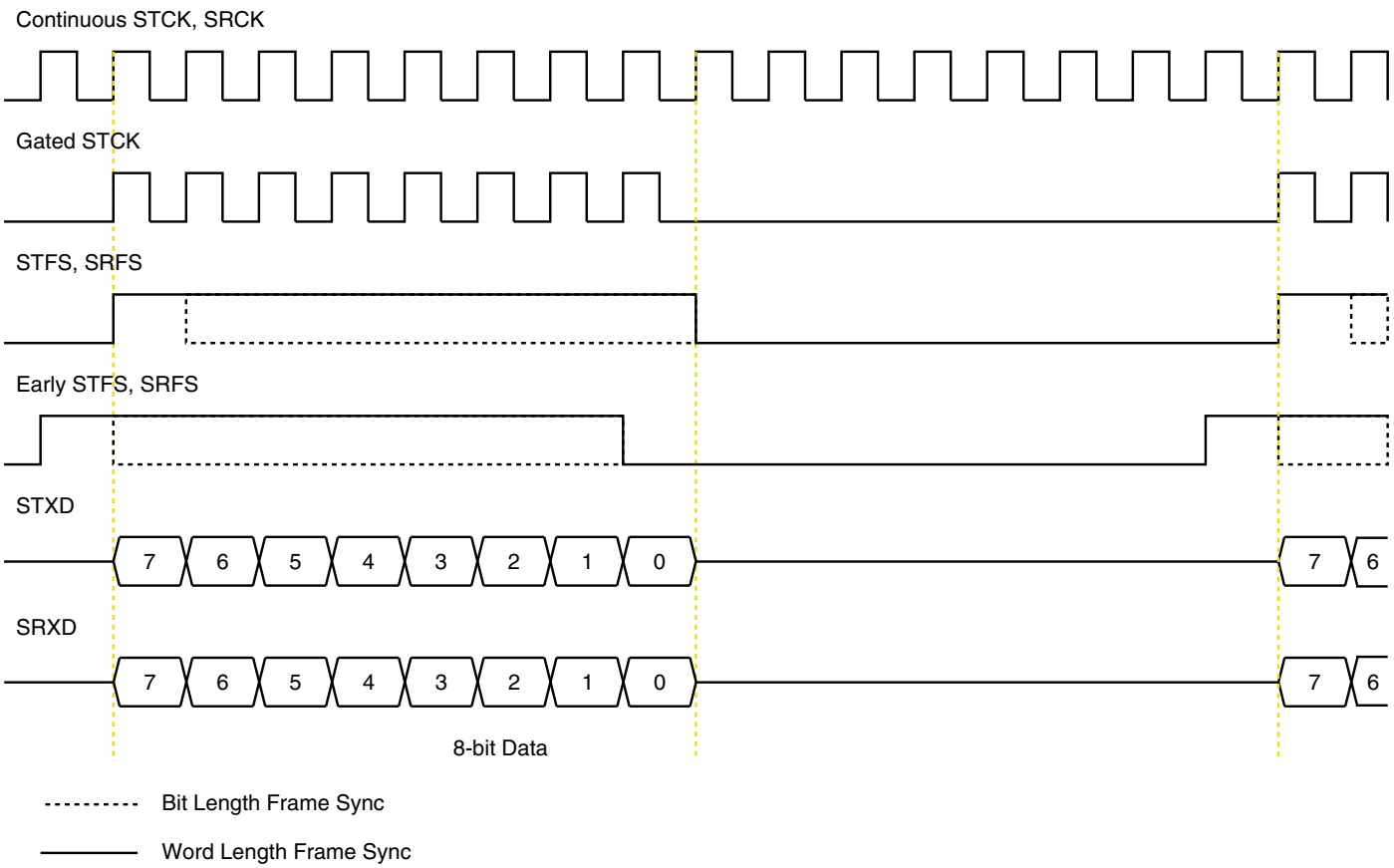


Figure 52-3. Serial Clock and Frame Sync Timing

See the table below for list of clock pin configurations.

Table 52-2. Clock Pin Configurations

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SRCK	STCK	SRFS	STFS
Asynchronous Mode								
0	0	0	0	0	RCK in	TCK in	RFS in	TFS in
0	0	0	0	1	RCK in	TCK in	RFS in	TFS out
0	0	0	1	0	RCK in	TCK in	RFS out	TFS in
0	0	0	1	1	RCK in	TCK in	RFS out	TFS out
0	0	1	0	0	RCK in	TCK out	RFS in	TFS in
0	0	1	0	1	RCK in	TCK out	RFS in	TFS out
0	0	1	1	0	RCK in	TCK out	RFS out	TFS in
0	0	1	1	1	RCK in	TCK out	RFS out	TFS out

Table continues on the next page...

Table 52-2. Clock Pin Configurations (continued)

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SRCK	STCK	SRFS	STFS
0	1	0	0	0	RCK out	TCK in	RFS in	TFS in
0	1	0	0	1	RCK out	TCK in	RFS in	TFS out
0	1	0	1	0	RCK out	TCK in	RFS out	TFS in
0	1	0	1	1	RCK out	TCK in	RFS out	TFS out
0	1	1	0	0	RCK out	TCK out	RFS in	TFS in
0	1	1	0	1	RCK out	TCK out	RFS in	TFS out
0	1	1	1	0	RCK out	TCK out	RFS out	TFS in
0	1	1	1	1	RCK out	TCK out	RFS out	TFS out
Synchronous Mode								
1	0	0	x	0	-	CK in	-	FS in
1	0	0	x	1	-	CK in	-	FS out
1	0	1	x	0	-	CK out	-	FS in
1	0	1	x	1	-	CK out	-	FS out
1	1	0	x	x	-	Gated in	-	-
1	1	1	x	x	-	Gated out	-	-

52.3 SSI Transmit FIFO 0 & 1 Registers

The SSI Transmit FIFO registers are 15x32-bit registers. These registers are not directly accessible by the end user. Transmit Shift Register (TXSR) receives its values from these FIFO registers. Transmitted data is first-in-first-out.

When the Transmit Interrupt Enable (TIE) bit in the SIER and either of the Transmit FIFO Empty Enable (TFE0 or 1) bits in the SIER are set, an interrupt is asserted whenever the number of empty slots exceed or are equal to the selected threshold value of corresponding Tx-FIFO. The threshold value is contained in the corresponding Transmit FIFO Watermark (TFWM0 or 1) field in the SSI FIFO Control/Status Register (SFCSR).

52.4 SSI Transmit Shift Register (TXSR)

The SSI Transmit Shift Register (TXSR) is a 24-bit shift register that contains the data being transmitted. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted out to the Serial Transmit Data (STXD) port by

the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the STXD port by the selected (internal/external) gated clock.

The Word Length control bits (WL[3:0]) in the SSI Transmit and Receive Clock Control Register (STCCR) determine the number of bits to be shifted out of the TXSR before it is considered empty and can be written to again. This word length can be 8, 10, 12, 16, 18, 20, 22 or 24 bits. The data to be transmitted occupies the most significant portion of the shift register if TXBIT0 is '0', otherwise it occupies the least significant portion. The unused portion of the register is ignored. Data is always shifted out of this register with the Most Significant Bit (MSB) first when the SHFD bit of the STCR is cleared. If this bit is set, the Least Significant Bit (LSB) is shifted out first. The figures below show the transmitter loading and shifting operation. The figures show the working for some WL values, the same can be extended for other values.

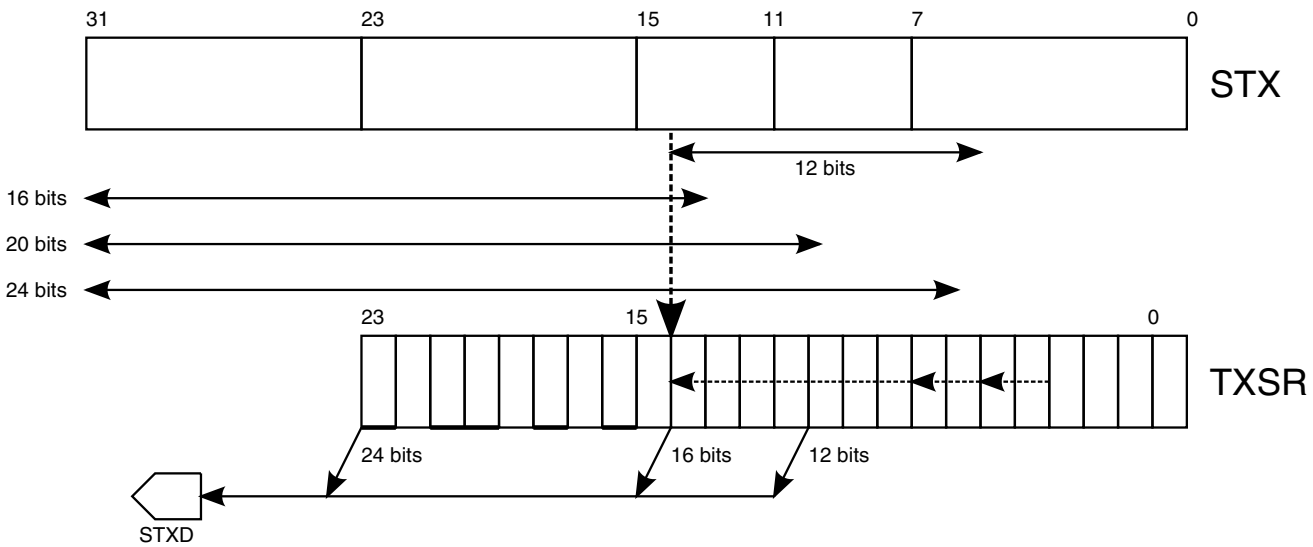


Figure 52-4. Transmit Data Path (TXBIT0=0, TSHFD=0) (MSB Alignment)

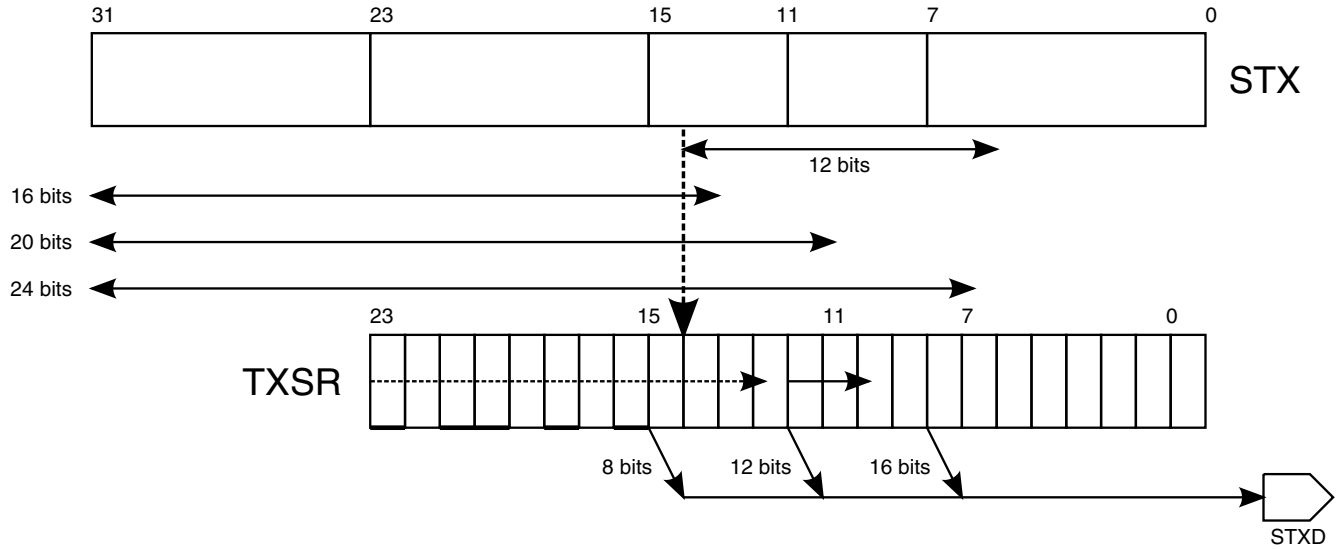


Figure 52-5. Transmit Data Path (TXBIT0=0, TSHFD=1) (MSB Alignment)

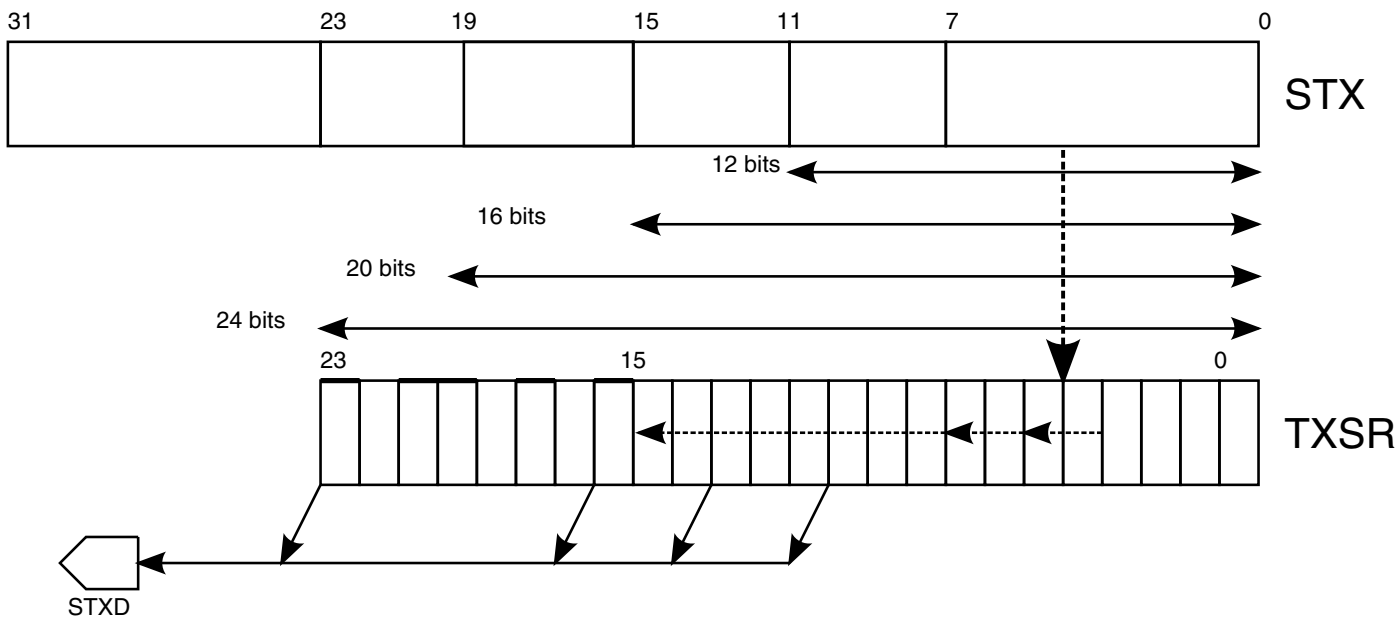


Figure 52-6. Transmit Data Path (TXBIT0=1, TSHFD=0) (LSB Alignment)

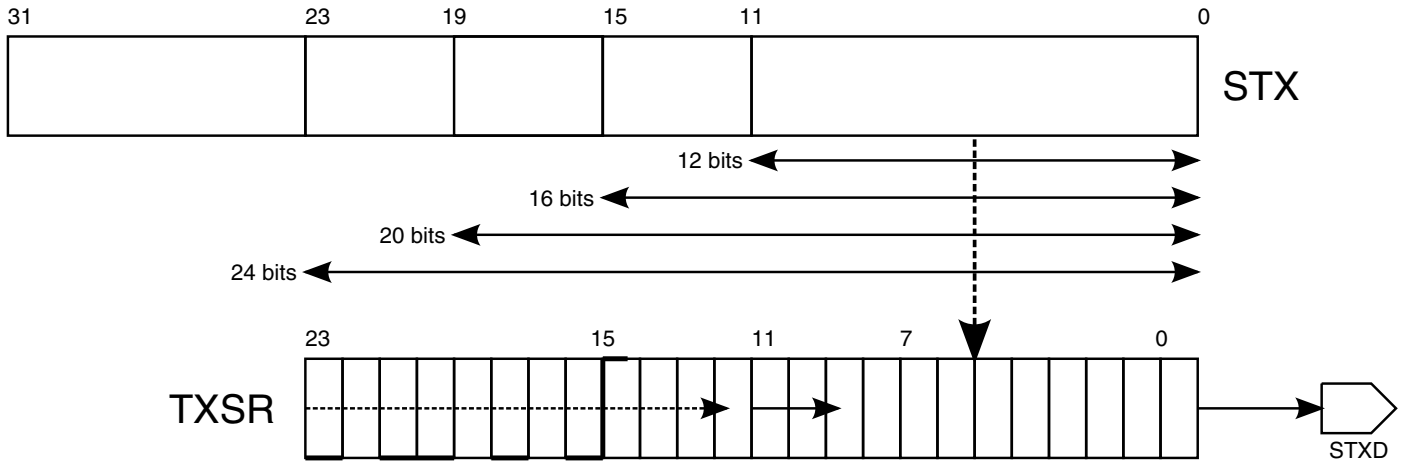


Figure 52-7. Transmit Data Path (TXBIT0=1, TSHFD=1) (LSB Alignment)

52.5 SSI Receive FIFO 0 and 1 Registers

The SSI Receive FIFO registers are 15x32-bit registers. These registers are not directly accessible by the end user. These FIFO registers receive data from the Receive Shift Register (RXSR). Received data is first-in-first-out.

When the Receive Interrupt Enable (RIE) bit in the SIER and either of the Receive FIFO Full Enable (RFF0_EN or RFF1_EN) bits in the SIER are set, an interrupt is asserted whenever the number of full slots exceeds or is equal to the selected threshold value of corresponding Rx-FIFO. The threshold value is contained in the corresponding Receive FIFO Watermark (RFWM0 or 1) field in the SSI FIFO Control/Status Register (SFCSR).

52.6 SSI Receive Shift Register (RXSR)

The SSI Receive Shift Register (RXSR) is a 24-bit, shift register that receives incoming data from the serial receive data SRXD port. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in by the selected (internal/external) gated clock.

Data is assumed to be received MSB first if the SHFD bit of the SSI.SRCR is cleared. If this bit is set, the data is received LSB first. Data is transferred to the appropriate SSI Receive Data Register (SRX0/1) or Receive FIFOs (if the receive FIFO is enabled and the corresponding SRX is full) after 8, 10, 12, 16, 18, 20, 22 or 24 bits have been shifted in depending on the WL[3:0] control bits. For receiving less than 24 bits of data, LSB bits

are appended with zero. The figures below show the receiver loading and shifting operation. These figures show the operation for several values of WL and the same can be extended for other values.

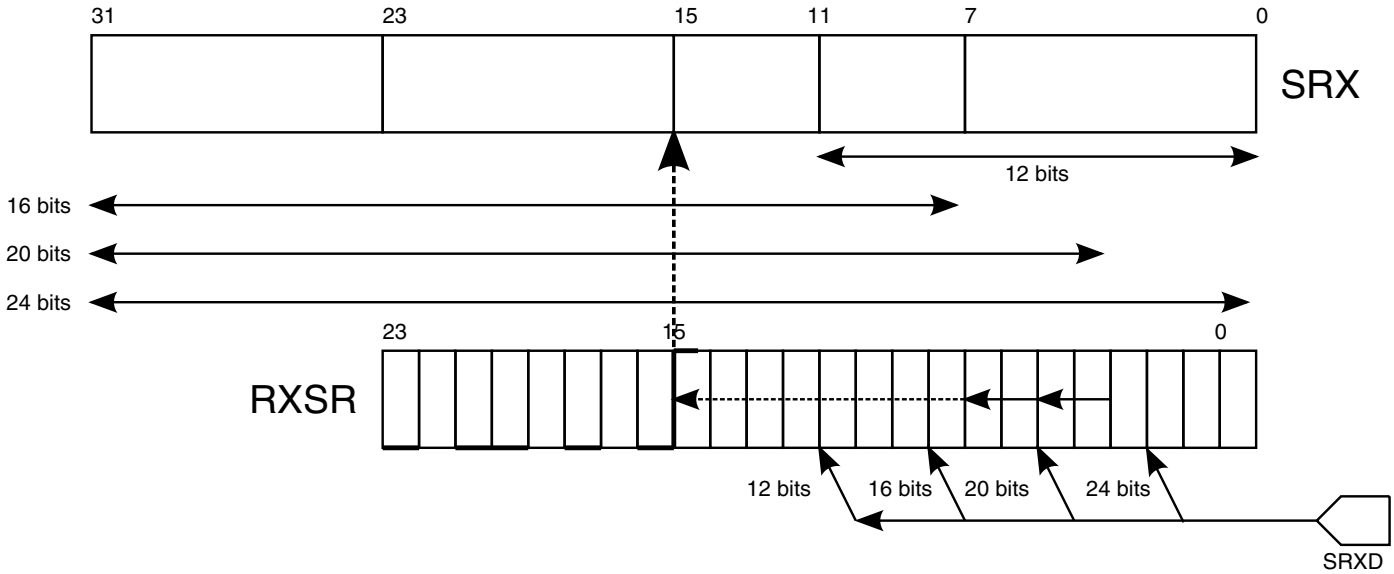


Figure 52-8. Receive Data Path (RXBIT0=0, RSHFD=0) (MSB Alignment)

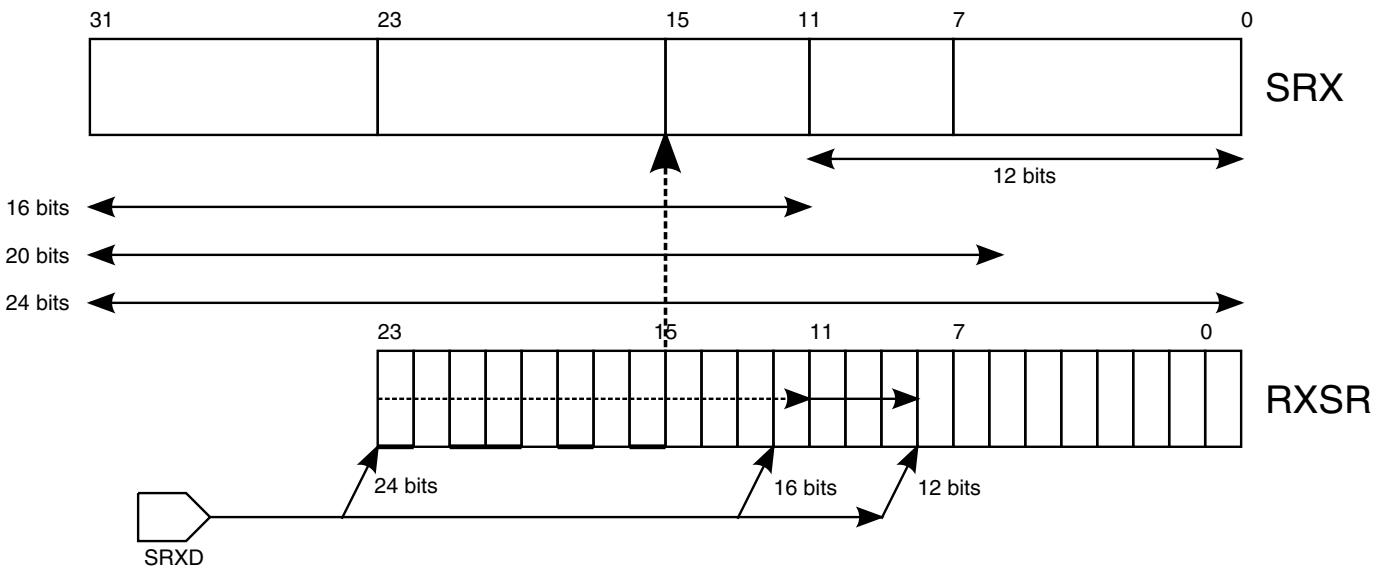


Figure 52-9. Receive Data Path (RXBIT0=0, RSHFD=1) (MSB Alignment)

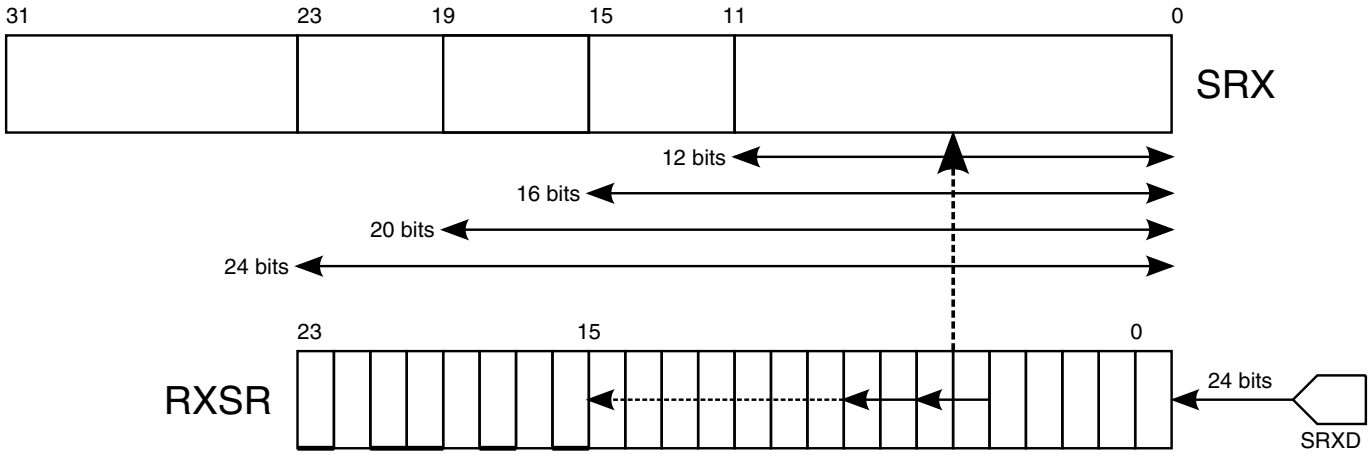


Figure 52-10. Receive Data Path (RXBIT0=1, RSHFD=0) (LSB Alignment)

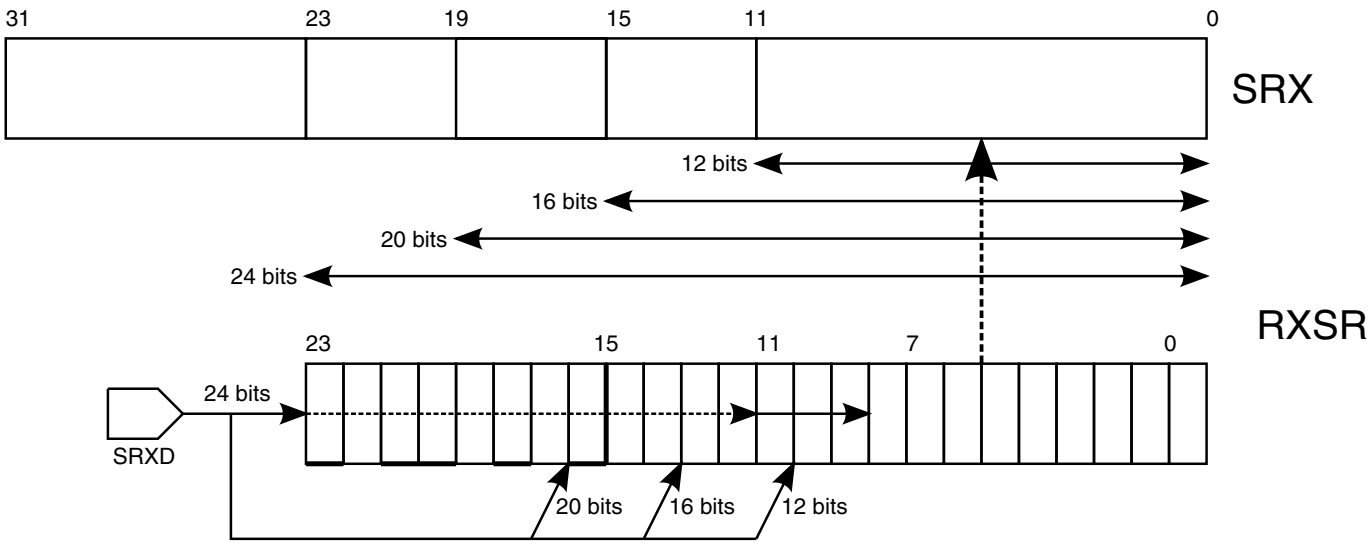


Figure 52-11. Receive Data Path (RXBIT0=1, RSHFD=1) (LSB Alignment)

52.7 Functional Description

52.7.1 Operating Modes

Different modes can be programmed by several bits in the SSI control registers.

See the table below for the list of SSI operating modes and some of the typical applications in which they can be used:

Table 52-3. SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Application
Asynchronous	Continuous	Normal	Multiple synchronous CODECs
Asynchronous	Continuous	Network	TDM CODEC or DSP networks
Synchronous	Continuous	Normal	Multiple synchronous CODECs
Synchronous	Continuous	Network	TDM CODEC or DSP network
Synchronous	Gated	Normal	SPI-type devices; DSP to ARM platform

The transmit and receive sections of the SSI can be synchronous or asynchronous. In Synchronous mode, the transmitter and the receiver use a common clock and frame synchronization signal. Masking of slots for Transmit and Receive section can differ in synchronous mode. Also the shifting of data in independent and for receive section depends on RXBIT0 and RSHFD bits in SSI.SRCR register. In Asynchronous mode, the transmitter and receiver each has its own clock and frame synchronization signals.

Normal or Network mode can be selected. In Normal mode, the SSI functions with one data word of I/O per frame. In Network mode, any number from two to thirty-two data words of I/O per frame can be used. Network mode is typically used in star time division multiplex networks with other processors or CODECs, allowing interface to time division multiplexed networks without additional logic. Gated clock mode option can be selected in Normal synchronous mode only. During Gated mode the clock is not-continuous and runs only during data-transmission. These distinctions result in the basic operating modes that allow the SSI to communicate with a wide variety of devices.

The SSI supports both Normal and Network modes, and these can be selected independently of whether the transmitter and receiver are synchronous or asynchronous. Typically these protocols are used in a periodic manner, where data is transferred at regular intervals, such as at the sampling rate of an external CODEC. Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC[4:0] bits in either the SSI.SRCCR or SSI.STCCR register, depending on whether data is being transmitted or received. The number of words transferred per frame depends on the mode of the SSI.

In Normal mode, one data word is transferred per frame. In Network mode, the frame is divided into anywhere between two and thirty-two time slots, where in each time slot one data word can optionally be transferred.

Apart from the above basic modes of operation, SSI supports the following modes which require some specific programming.

- I2S mode
- AC97 mode
    - AC97 Fixed mode
    - AC97 Variable mode

In (non-I2S) slave modes (external frame sync), the programmed word length setting of the SSI should be equal to the word length setting of the master. In I2S slave mode, the programmed word length setting of the SSI can be lesser than or equal to the word length setting of the I2S master (external CODEC).

In slave modes, the programmed frame length setting (DC bits) of the SSI can be lesser than or equal to the frame length setting of the master (external CODEC).

The following sections provide detailed descriptions of the above modes.

52.7.1.1 Normal Mode

Normal mode is the simplest mode of the SSI. It is used to transfer data in one time slot per frame. A time slot is a unit of data and the WL[3:0] bits define the number of bits in a time slot. In Continuous Clock mode, a frame sync occurs at the beginning of each frame.

The length of the frame is determined by the following factors:

- The period of the Serial Bit Clock (DIV2, PSR, PM[7:0] bits for internal clock or the frequency of the external clock on the STCK port)
- The number of bits per time slot (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

If Normal mode is configured with more than one time slot per frame, data is transferred only in the first time slot. No data is transferred in subsequent time slots. In Normal mode, DC[4:0] values corresponding to more than a single time slot in a frame, only results in lengthening of the frame.

52.7.1.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in Normal mode are:

- SSI Enabled (SSIEN = 1)
- Write data to Transmit Data Register (STX)
- Transmitter Enabled (TE = 1)
- Frame sync active (for continuous clock case)
- Bit clock begins

When the above conditions occur in Normal mode, the next data word is transferred into the Transmit Shift Register (SSI.TXSR) from the Transmit Data Register 0 (SSI.STX0), or from the Transmit FIFO 0 Register, if transmit FIFO 0 is enabled. The new data word is transmitted on arrival of frame-sync preceded by clocks in continuous clock mode. In gated-external mode, data word in transmitted on arrival of frame-sync. In gated-internal mode, data word is transmitted whenever data is available in Tx-FIFO.

If Transmit FIFO 0 is not enabled and the transmit data register empty enable (TDE0_EN) and transmit interrupt enable (TIE) bits are set, transmit interrupt occurs when the word in SSI_STX0 is shifted to Transmit Shift (SSI.TXSR) register for shifting.

If Transmit FIFO 0 is enabled and the transmit fifo empty enable (TFE0_EN) and transmit interrupt enable (TIE) bits are set, transmit interrupt occurs when the number of empty slots in Transmit Fifo 0 exceed or are equal to the selected threshold value i.e. Transmit Fifo 0 Watermark (TFWM0) value. If transmit FIFO 0 is enabled and filled with data, 15 data words can be transferred before the core must write new data to the SSI.STX0 register.

The STXD port is disabled except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not disabled, even if both receiver and transmitter are disabled.

52.7.1.1.2 Normal Mode Receive

The conditions for data reception from the SSI are:

- SSI enabled (SSIEN = 1)
- Receiver enabled (RE = 1)
- Frame sync active (for continuous clock case)
- Bit clock begins

With the above conditions in Normal mode with a continuous clock, each time the frame sync signal is generated (or detected) a data word is clocked in. With the above conditions and a gated clock, each time the clock begins, a data word is clocked in.

If Receive FIFO 0 is not enabled and Receive Interrupt enable (RIE) and Received Data 0 Ready enable (RDR0_EN) bits are set, receive interrupt occurs when received data word is transferred from the Receive Shift Register (SSI.RXSR) to the Receive Data Register 0 (SSI.SRX0), thus setting the Receive Data Ready 0 (RDR0) flag.

If Receive FIFO 0 is enabled, and Receive Interrupt enable (RIE) and Received Fifo 0 full enable (RFF0_EN) bits are set, receive interrupt occurs when the received data word is transferred to the Receive FIFO 0 and Receive FIFO 0 reaches the selected threshold and results in Receive FIFO Full 0 (RFF0) flag to get set.

The core program has to read the data from the Receive Data Register 0 (SSI.SRX0) (in case Receive FIFO0 is disabled) before a new data word is transferred from the Receive Shift Register (SSI.RXSR), otherwise the Receive Overrun Error 0 (ROE0) bit is set. If receive FIFO 0 is enabled, the Receive Overrun Error 0 (ROE0) bit is set when the Receive FIFO 0 data level reaches the selected threshold and a new data word is ready to be transferred to the Receive FIFO 0.

See the following figure for an illustration of transmitter and receiver timing for an 8-bit word in the first time slot in Normal mode, continuous clock with a late word length frame sync. The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register, which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and, at the end of the time slot, this data is transferred to the Rx Data Register.

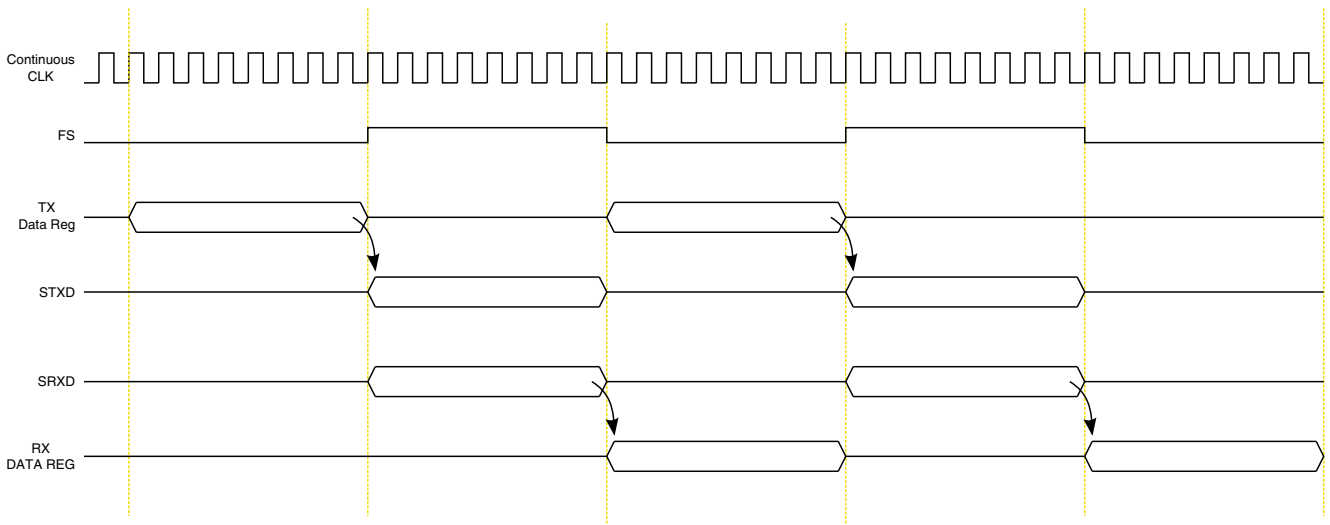


Figure 52-12. Normal Mode Timing - Continuous Clock

The following figure shows a similar case for internal (SSI generates clock) gated clock mode and Figure 52-14 shows a case for external (SSI receives clock) gated clock mode.

NOTE

A pull-down resistor is required in the gated clock case because the clock port is disabled between transmissions.

The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register, which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and, at the end of the time slot, this data is transferred to the Rx Data Register. In case of Internal Gated clock mode, the Tx Data line and clock output port are put in the high-impedance state at the end of transmission

of the last bit (at the completion of the complete clock cycle), whereas, in External Gated clock mode, the Tx Data line is tri-stated at the last inactive edge of the incoming bit clock (during the last bit in a data word).

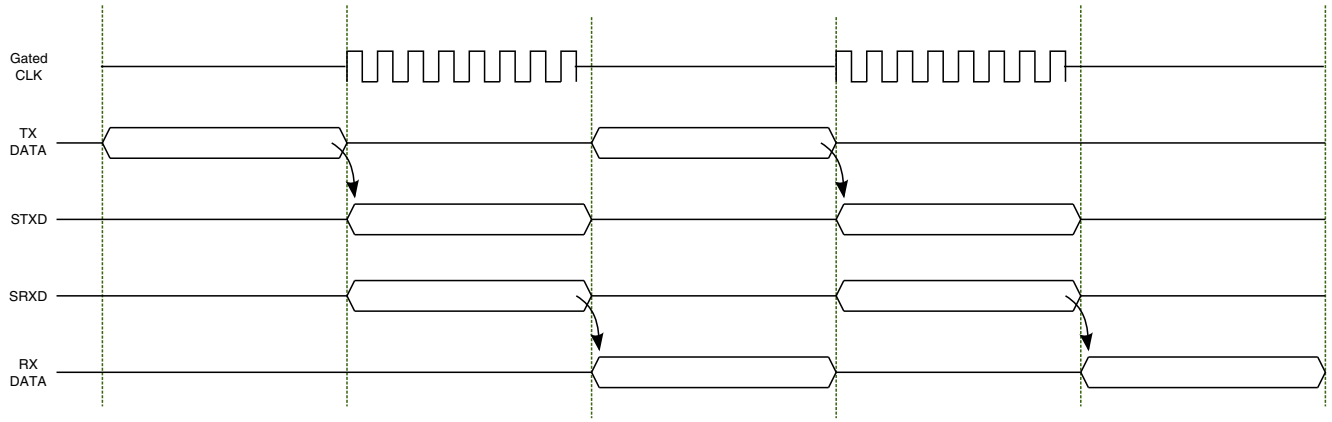


Figure 52-13. Normal Mode Timing - Internal Gated Clock

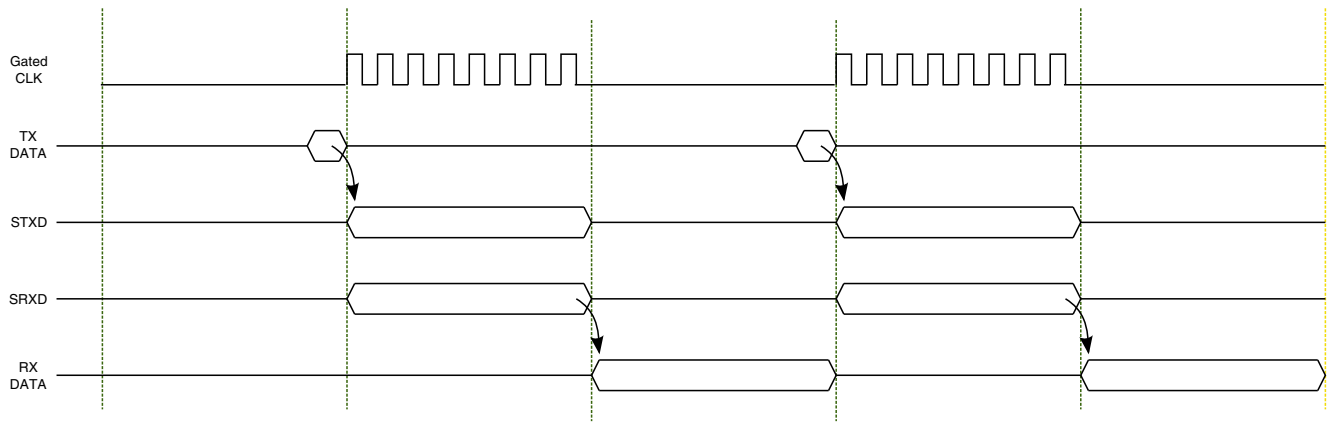


Figure 52-14. Normal Mode Timing - External Gated Clock

52.7.1.2 Network Mode

Network mode is used for creating a Time Division Multiplexed (TDM) network, such as a TDM CODEC network or a network of DSPs. In Continuous Clock mode, a frame sync occurs at the beginning of each frame.

In this mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate CODEC or DSP on the network. The DSP can be a master device that controls its own private network, or a slave device that is connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception of one data word can occur in each time slot (rather than in just the frame sync time slot as in Normal mode). The frame rate dividers, controlled by the DC[4:0] bits, select two to thirty-two time slots per frame. The length of the frame is determined by the following factors:

- The period of the serial bit clock (PSR, PM[7:0] bits for internal clock, or the frequency of the external clock on the STCK port)
- The number of bits per sample (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

In Network mode, data can be transmitted in any time slot. The distinction of the Network mode is that each time slot is identified with respect to the frame sync (data word time). This time slot identification allows the option of transmitting data during the time slot by writing to the STX registers or ignoring the time slot as determined by SSI.STMSK register bits. The receiver is treated in the same manner and received data is only transferred to the receive data register/fifo if the corresponding time slot is enabled (through SSI.SRMSK).

By utilizing the SSI.STMSK and SSI.SRMSK registers, software only has to service the SSI during valid time slots. This eliminates any overhead associated with unused time slots. Refer to SSI Transmit Time Slot Mask Register (SSI_SSI_STMSK) and SSI Receive Time Slot Mask Register (SSI_SSI_SRMSK) for more information on SSI.STMSK and SSI.SRMSK.

In the Two-Channel mode of operation, the second set of Transmit and Receive FIFOs and Data Registers are used to create two separate channels. These channels are completely independent, with a their own set of Core interrupts and DMA requests, which are identical to the ones available for the default channel. In this mode, data is transmitted/received in enabled time slots alternately from/to FIFO 0 and FIFO 1, starting from FIFO 0. The first data word is taken from FIFO 0 and transmitted in the first enabled time slot and subsequently, data is loaded from FIFO 1 and FIFO 0 alternately and transmitted. Similarly, the first received data is sent to FIFO 0 and subsequent data is sent to FIFO 1 and FIFO 0 alternately. Time slots can be selected through the Transmit and Receive Time Slot Mask registers (SSI.STMSK and SSI.SRMSK). For using this mode of operation, the TCH_EN bit (SCR[8]) needs to be set.

52.7.1.2.1 Network Mode Transmit

The transmit portion of SSI is enabled when the SSIEN and the TE bits in the SSI.SCR are both set. However, for continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new frame sync (transmission starts from the next frame boundary).

Normal start-up sequence for transmission is to perform the following:

1. Enable Network Mode.
2. Enable SSI
3. Write the data to be transmitted to the SSI.STX register. This clears the TDE flag
4. Set the TE bit to enable the transmitter on the next frame boundary (for continuous clock case).
5. Enable transmit interrupts.

(Alternatively, the programmer may decide not to transmit in a time slot by configuring the STMSK.) TDE flag is set as data is shifted from SSI.STX register to TXSR, but the STXD port remains disabled during the time slots. When the next frame sync is detected or generated (continuous clock), the data word in TXSR and is shifted out (transmitted). When the SSI.STX register is empty, the TDE bit is set, which causes a transmitter interrupt (in case the FIFO is disabled) to be sent if the TIE bit is set. Software can poll the TDE bit or use interrupts to reload the STX register with new data for the next time slot. Failing to reload the SSI.STX register before the TXSR is finished shifting (empty) causes a transmitter underrun and the TUE error bit is set. In case the FIFO is enabled, the TFE flag is set in accordance with the watermark setting and this flag causes the transmitter interrupt to occur.

The operation of clearing the TE bit disables the transmitter after completion of transmission of the current frame. Setting the TE bit enables transmission from the next frame. During that time the STXD port is disabled. The TE bit should be cleared after the TDE bit is set to ensure that all pending data is transmitted.

To summarize, the Network mode transmitter generates interrupts every enabled time slot (when FIFO is disabled) and requires the core program to respond to each enabled time slot. These responses from the core are one of the following:

- Write data in data register to enable transmission in the next time slot.
- Configure the time slot register to disable transmission in the next time slot (unless time slot is already masked by SSI.STMSK register bit).
- Do nothing-transmit underrun occurs at the beginning of the next time slot and the previous data is re-transmitted.

In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected (TDE1 is low by default).

52.7.1.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSIEN and the RE bits in the SSI.SCR are set. However, the receive enable only takes place during that time slot if RE is enabled before the second to last bit of the word. If the RE bit is cleared, the receiver is disabled at the end of the current frame.

SSI is capable of finding the start of the next frame automatically. When the word is completely received, it is transferred to the SRX register, which sets the RDR bit (Receive Data Ready). Setting the RDR bit causes a receive interrupt to occur if the receiver interrupt is enabled (the RIE bit is set) and (Receive data ready enable) RDR_EN bit is set. The second data word (second time slot in the frame), begins shifting in immediately after the transfer of the first data word to the SSI.SRX register. The core program has to read the data from the Receive Data Register (which clears RDR) before the second data word is completely received (ready to transfer to RX data register) or a receive overrun error occurs (the ROE bit is set).

An interrupt can occur after the reception of each enabled data word or the programmer can poll the RDR flag. The core program response can be one of the following:

- Read RX and use the data.
- Read RX and ignore the data.
- Do nothing-the receiver overrun exception occurs at the end of the current time slot.

NOTE

For a continuous clock, the optional frame sync output and clock output signals are not affected, even if the transmitter or receiver is disabled. TE and RE do not disable the bit clock or the frame sync generation. To disable the bit clock and the frame sync generation, the SSIEN bit in the SSI.SCR can be cleared or TFR_CLK_DIS/RFR_CLK_DIS bits can be set, or the port control logic external to the SSI (for example, in the IOMUXC) can be re configured.

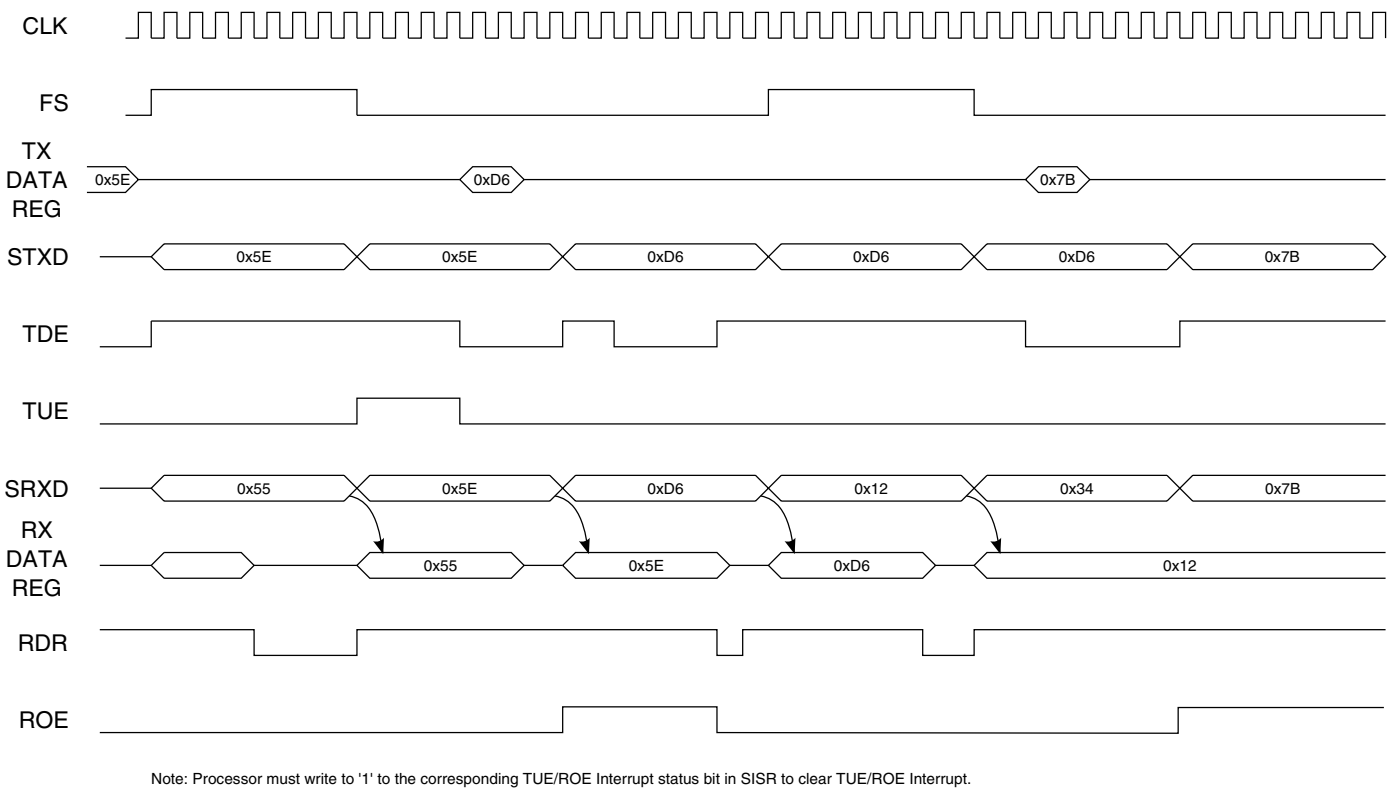
In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected.

The transmitter and receiver timing for an 8-bit word with continuous clock, FIFO disabled, three words per frame sync in Network mode is shown in the figure below).

NOTE

The transmitter repeats the value 0x5E because of an underrun condition

For the receive section, data received on the SRXD pin gets transferred to the Rx Data register at the end of each time slot. If the FIFO is disabled, the RDR flag gets set and causes a receiver interrupt if RE, RIE and RDR_EN bits are set. If the FIFO is enabled, then the RFF flag is used for interrupt generation (this flag is set in accordance with the watermark settings). Here all time slots are enabled. The receive data ready flag is set after reception of the first data (0x55). Since the flag is not cleared (Rx Data Register is not read by core), the Receive Overrun Error (ROE) flag is set on reception of the next data (0x5E). ROE flag is cleared on writing '1' to the corresponding interrupt status bit in SSI Status Register.



Note: Processor must write to '1' to the corresponding TUE/ROE Interrupt status bit in SISR to clear TUE/ROE Interrupt.

Figure 52-15. Network Mode Timing - Continuous Clock

52.7.1.3 Gated Clock Mode

Gated Clock mode is often used to hook up to SPI-type interfaces on Micro controller Units (MCUs) or external peripheral chips. In Gated Clock mode, the presence of the clock indicates that valid data is on the STXD or SRXD ports. For this reason, no frame sync is needed in this mode.

Once transmission of data has completed, the clock is pulled to the inactive state. Gated clocks are allowed for both the transmit and receive sections with either internal or external clock in Normal mode. Gated clocks are not allowed in Network mode. See Table 52-2 ("Clock Pin Configurations") for SSI configuration for gated-mode operation.

The clock runs when the TE bit and/or the RE bit are appropriately enabled. For the case of internally generated clock, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs (such as the first time slot in Normal mode), the internal bit clock is enabled onto the appropriate clock port. This allows data to be transferred out in periodic intervals in Gated Clock mode. With an external clock, the SSI waits for a clock signal to be received. Once the clock begins, valid data is shifted in. Care should be taken to clear all DC bits (0x00000) when SSI is used in Gated mode. In gated mode of operation the TFS, RFS, TLS, RLS, TFRC and RFRC bits of SSI.AISR register are not generated.

For Gated clock operated in external clock mode, a proper clock signalling must be applied to the SSI STCK in order for it to function properly. When TSCKP is 0, CLK_IST value should be 1. When TSCKP is 1, CLK_IST value should be 0. If the SSI uses rising edge transition to clock data (TSCKP=0) and the falling edge transition to latch data (RSCKP=0), the clock must be in an active low state when idle. If the SSI uses falling edge transition to clock data (TSCKP=1) and the rising edge transition to latch data (RSCKP=1), the clock must be in a active high state when idle. The figures below illustrate the different edge clocking/latching.

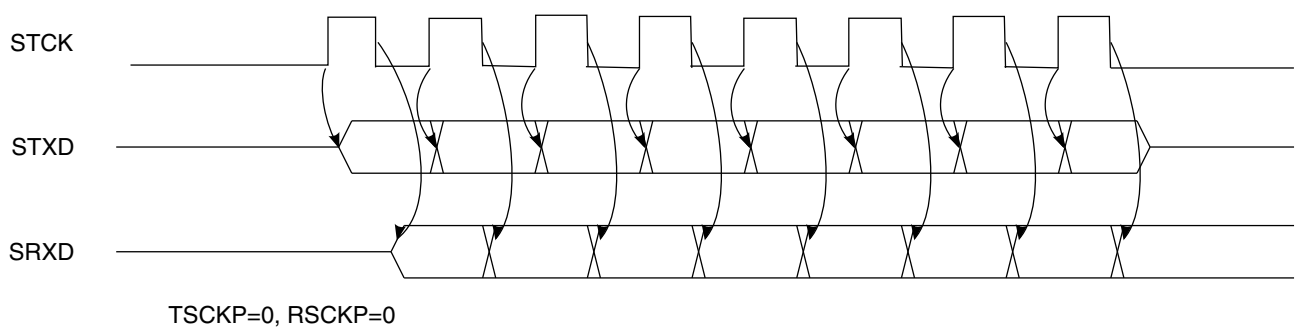


Figure 52-16. Internal Gated Mode Timing - Rising Edge Clocking / Falling Edge Latching

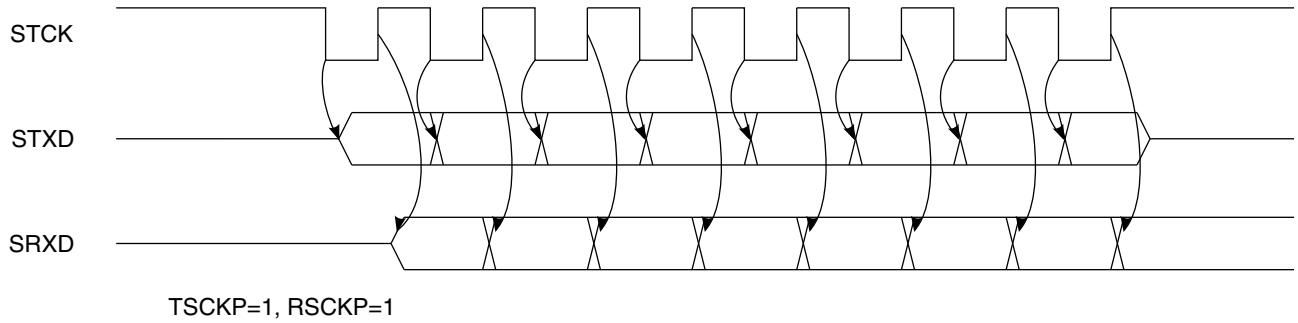


Figure 52-17. Internal Gated Mode Timing - Falling Edge Clocking / Rising Edge Latching

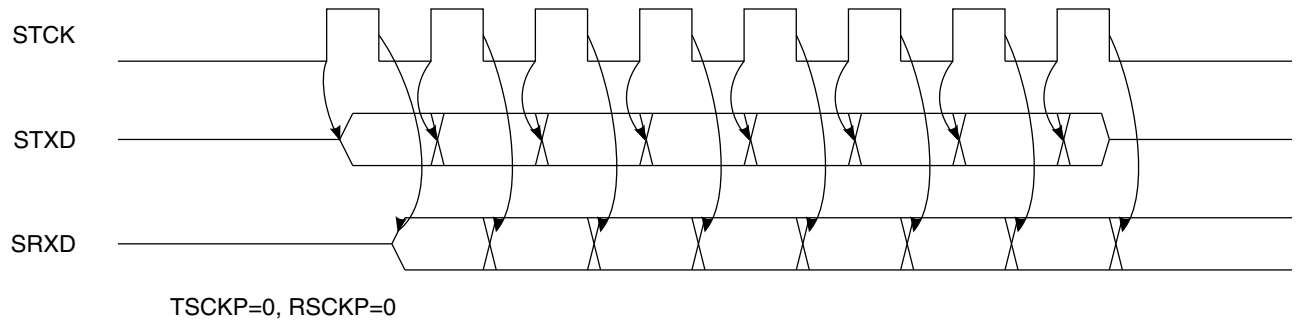


Figure 52-18. External Gated Mode Timing - Rising Edge Clocking / Falling Edge Latching

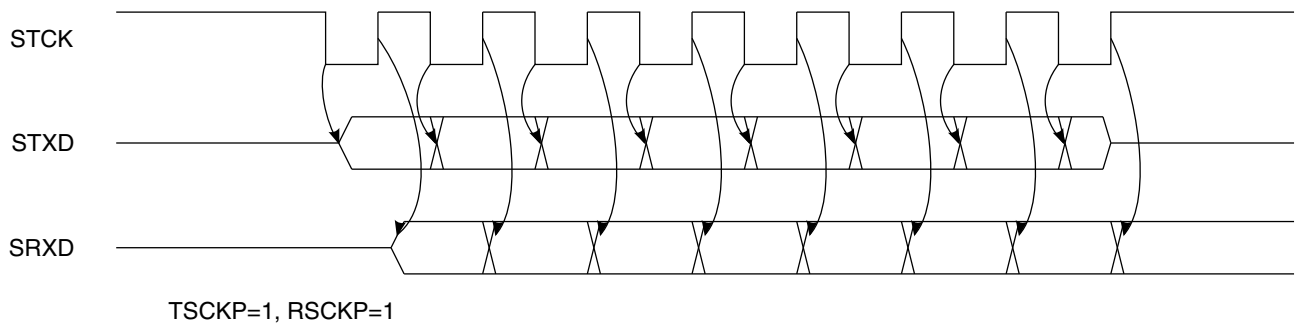


Figure 52-19. External Gated Mode Timing - Falling Edge clocking / Rising Edge Latching

The bit clock ports must be kept free of timing glitches. If a single glitch occurs, all ensuing transfers will be out of synchronization.

In case of External Gated Mode, even though the Tx Data line is put in the high-impedance state at the last non-active edge of the bit clock, the round trip delay should be sufficient to take care of hold time requirements at the external receiver.

52.7.1.4 I2S Mode

The SSI is compliant to the Inter-IC Sound (I2S) bus specification from Philips Semiconductors (February 1986, Revised June 5, 1996). For more information on I2S, refer to the latest version of NXP Semiconductor's I2S specification.

See the following figure for an illustration of the basic I2S protocol timing.

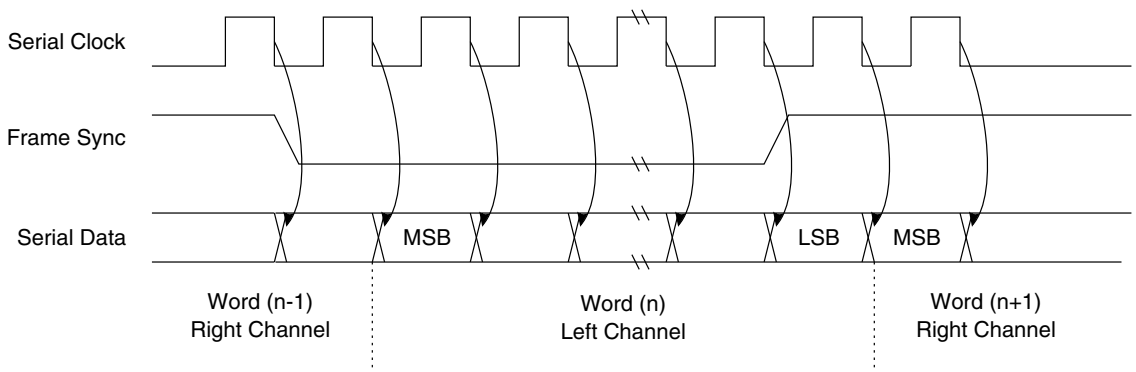


Figure 52-20. I2S Mode Timing - Serial Clock, Frame Sync and Serial Data

Select I2S mode using the options listed in the table below.

Table 52-4. I2S Mode Selection

I2S_MODE[1]	I2S_MODE[0]	Mode Type
0	0	Normal mode
0	1	I2S master mode
1	0	I2S slave mode
1	1	Normal mode

In normal mode operation, no register bits are forced to any particular state internally and the user can program the SSI to work in any operating condition.

When I2S modes are entered (I2S master (01) or I2S slave (10)), the following settings are recommended:

- Sync mode (SSI_SCR[4] =1)
- Tx shift direction: MSB transmitted first (SSI_STCR[4]=0)
- Rx shift direction: MSB received first (SSI_SRCR[4]=0)
- Tx data clocked at falling edge of the clock (SSI_STCR[3]=1)
- Rx data latched at rising edge of the clock (SSI_SRCR[3]=1)
- Tx frame sync active low (SSI_STCR[2]=1)
- Rx frame sync active low (SSI_SRCR[2]=1)

- Tx frame sync initiated one bit before data is transmitted (SSI_STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SSI_SRCR[0]=1)
- TX Frame Rate should be 2 (SSI_STCCR[12:8] = 1)
- RX Frame Rate should be 2 (SSI_SRCCR[12:8] = 1)

In I2S master mode (SSI_SCR[6:5]=01), the following additional settings are recommended:

- TXDIR bit (SSI_STCR[5]) set to 1 to select internal generated bit clock
- TFDIR bit (SSI_STCR[6]) set to 1 to select internal generated frame sync

In I2S master mode(SSI.SCR[6:5]=01), the following settings are internally overridden by the hardware:

- Network mode is selected (SSI_SCR[3]=1)
- Tx frame sync length set to one-word-long-frame (SSI_STCR[1]=0)
- Rx frame sync length set to one-word-long-frame (SSI_SRCR[1]=0)
- Tx shifting w.r.t. bit 0 of TXSR (SSI.STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SSI.SRCR[9]=1)

The user needs to set the following control bits to configure the bit clock and frame sync:

- PM (SSI_STCCR[7:0])
- PSR (SSI_STCCR[17])
- DIV2(SSI_STCCR[18]
- WL (SSI_STCCR[16:13]
- DC (SSI_STCCR[12:8])

The word length is fixed to 32 in I2S Master mode and the WL bits determine the number of bits that will contain valid data (out of the 32 transmitted/received bits in each channel).

In I2S slave mode(SSI_SCR[6:5]=10), the following additional settings are recommended:

- TXDIR bit(SSI_STCR[5]) set to 0 to select external generated bit clock
- TFDIR bit(SSI_STCR[6]) set to 0 to select external generated frame sync

In I2S slave mode(SSI_SCR[6:5]=10), the following settings are internally overridden by the hardware:

- Normal mode is selected (SSI_SCR[3]=0)
- Tx frame sync length set to one-bit-long-frame (SSI_STCR[1]=1)
- Rx frame sync length set to one-bit-long-frame (SSI_SRCR[1]=1)
- Tx shifting w.r.t. bit 0 of TXSR (SSI_STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SSI_SRCR[9]=1)

The user needs to set the following control bits to configure the data transmission:

- WL (SSI_STCCR[16:13])
- DC (SSI_STCCR[12:8])

The word length is variable in I2S slave mode and the WL bits determine the number of bits that will contain valid data. The actual word length is determined by the external CODEC. The external I2S Master still sends frame sync according to the I2S protocol (early, word wide and active low), the SSI internally operates so that each frame sync transition is the start of a new frame (the WL bits determine the number of bits to be transmitted/received). After one data word has been transferred, the SSI waits for the next frame sync transition to start operation in the next time slot. Transmit (STMSK) and receive (SRMSK) mask bits should not be used in I2S Slave mode of operation. Masking is supported only for network mode of operation.

52.7.1.5 AC97 Mode

In AC97 mode of operation, the SSI transmits a 16-bit Tag Slot at the start of a frame and the rest of the slots (in that frame) are all 20-bits wide. The same sequence is followed while receiving data. Refer to the AC97 specification for details regarding transmit and receive sequences and data formats.

NOTE

The Audio Codec specification released in 1997 [AC '97] defines the Architecture and Digital Interface, specifically designed for implementing audio and modem I/O functionality in personal computers. Companion specifications include the Modem Codec [MC '97], and the combined Audio/Modem Codec standard [AMC '97]. The current version of AC '97 was produced in 2002. The AC-97 specification defines a recommended 48-pin QFP IC package.

Note that the SSI only has one RxDATA pin so the SSI can only support one codec. Secondary codecs are not supported.

When AC97 mode is enabled, the following settings are internally overridden by the hardware. The programmed register values are not changed by entering AC97 mode but they no longer apply to the block's operation. Writing to the programmed register fields will update their values; these updates can be seen by reading back the register fields. However, these settings will not take effect until AC97 mode is turned off.

The register bits within the bracket are the equivalent settings:

- Sync mode is entered (SSI.SCR[4] =1)

- Network mode is selected (SSI.SCR[3]=1)
- Tx shift direction is MSB transmitted first (SSI.STCR[4]=0)
- Rx shift direction is MSB received first (SSI.SRCR[4]=0)
- Tx data is clocked at rising edge of the clock (SSI.STCR[3]=0)
- Rx data is latched at falling edge of the clock (SSI.SRCR[3]=0)
- Tx frame sync is active high (SSI.STCR[2]=0)
- Rx frame sync is active high (SSI.SRCR[2]=0)
- Tx frame sync length is one-word-long-frame (SSI.STCR[1]=0)
- Rx frame sync length is one-word-long-frame (SSI.SRCR[1]=0)
- Tx frame sync initiated one bit before data is transmitted (SSI.STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SSI.SRCR[0]=1)
- Tx shifting w.r.t. bit 0 of TXSR (SSI.STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SSI.SRCR[9]=1)
- Tx FIFO is enabled (SSI.STCR[7]=1)
- Rx FIFO is enabled (SSI.SRCR[7]=1)
- TFDIR bit (SSI.STCR[6]) is forced to 1 internally to select internal generated frame sync
- TXDIR bit (SSI.STCR[5]) is forced to 0 internally to select external generated bit clock

Any alteration of these bits individually will not affect the operational conditions of the SSI unless AC97 mode is deselected.

Hence, the only control bits needed to be set by the user to configure the data transmission/reception are the WL (SSI.STCCR[16:13]) and DC (SSI.STCCR[12:8]) bits. In AC97 mode, the WL bits can only legally take the values corresponding to 16-bit (truncated data) or 20-bit time slots. In case WL bits are set to select 16-bit time slots, the SSI pads the transmit data (four least significant bits) with zeros and while receiving, stores only the most significant 16 bits in the Rx FIFO.

Follow the sequence for programming the SSI to work in AC97 mode:

1. Program the WL bits to a value corresponding to either 16 or 20 bits. The WL bit setting is only for the data portion of the AC97 frame (Slots #3 through #12). The Tag slot (Slot #0) is always 16 bits wide and the Command Address and Command Data slots (Slots #1 and #2) are always 20 bits wide.
2. Select the number of time slots by programming the DC bits. For AC97 operation, DC bits should be set to a value of '0xC', resulting in 13 time slots per frame.
3. Write data to be transmitted, in Tx FIFO 0 (through Tx Data Register 0) and Tx FIFO 1 while using Two-Channel Mode (TCH_EN = 1).
4. Program the FV, TIF, RD, WR and FRDIV bits in SSI.SACNT register
5. Update the contents of SSI.SACADD, SSI.SACDAT and SSI.SATAG (for Fixed mode only) registers

6. Enable the AC97 mode of operation (AC97EN bit in SSI.SACNT register)

Once the SSI starts transmitting and receiving data (after being configured in AC97 mode), the programmer needs to service the interrupts, as and when they are raised (updates to command address/data or tag registers, reading of received data and writing more data for transmission). Further details regarding fixed and variable mode implementation are provided in the following sections.

While using AC97 in Two-Channel Mode (TCH_EN=1), it is recommended that the received tag is not stored in the Rx FIFO (TIF=0). In case the programmer needs to update the SSI.SATAG register and also issue a RD/WR command (in a single frame), it is recommended that the SSI.SATAG register be updated prior to issuing a RD/WR command.

52.7.1.5.1 AC97 Fixed Mode (SSI.SACNT[1]=0)

In fixed mode of operation, SSI transmits in accordance with the AC97 Frame Rate Divider bits (i.e. FRDIV in SACNT) which decides the number of frames for which the SSI should be idle, after operating for one frame.

In a valid frame, TAG Value (written by Core) will be transmitted in Slot #0, Command Address will be transmitted in Slot #1 in case of RD/WR Command, and Command Data will be transmitted in Slot #2 in case of a WR Command. The data from TX-FIFO is transmitted in Slot #3 - Slot #12 depending on the valid slots indicated by the TAG value.

While receiving, bit 15 of the TAG Value (Slot #0) is checked to see if the CODEC is ready. If this bit is set, the frame is received. The received TAG provides the information about Slots containing valid data. The the corresponding TAG bit is valid, the Command Address (Slot #1) and Command Data (Slot #2) values are stored in the corresponding registers. The received data (Slot #3 - Slot #12) is then stored in the Rx-FIFO (for valid slots).

52.7.1.5.2 AC97 Variable Mode (SSI.SACNT[1]=1)

In Variable Mode, the transmit slots which should contain data in the current frame are determined by SLOTREQ bits received in the previous frame. While receiving, if the CODEC is ready, the frame is received and the SLOTREQ bits (contained in Slot #1) are stored for scheduling transmission in the next frame.

The SSI.SACCST, SSI.SACCEN and SSI.SACCDIS registers helps in determining which transmit slots are active. This information is used to ensure that SSI does not transmit data for powered-down/inactive channels.

52.7.2 External Frame and Clock Operation

When applying external frame sync and clock signals to SSI, there should be at least 4-bit clock cycles between the enabling of the transmit or receive section and the rising edge of the corresponding frame sync signal. The transition of STFS or SRFS should be synchronized with the rising edge of external clock signal, STCK or SRCK.

52.7.2.1 Data Alignment Formats Supported

The SSI supports three data formats in order to provide flexibility with handling data. These formats dictate how data is written to (and read from) the data registers. Therefore, data can appear in different places in SSI.STX0/1 and SSI.SRX0/1 based on the data format and the number of bits per word. Independent data formats are supported for both the transmitter and receiver (that is, the transmitter and receiver can use different data formats).

The supported data formats are:

- MSB alignment
- LSB alignment
  - Zero-extended (receive data only)
  - Sign-extended (receive data only)

With MSB alignment, the most significant byte is bits 31 through 24 of the data register if the word length is larger than or equal to 16 bits. If the word length is less than 16 bits and MSB alignment is chosen, the most significant byte is bits 15 through 8. With LSB alignment, the least significant byte is bits 7 through 0. Data alignment is controlled by the TXBIT0 bit in the SSI.STCR and the RXBIT0 bit in the SSI.SRCR. See the table below for the bit assignment for all the data formats supported by the SSI.

Table 52-5. Data Alignment

Format	Bit Number																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8-bit LSB Aligned																									7	6	5	4	3	2	1	0
8-bit MSB Aligned																	7	6	5	4	3	2	1	0								
10-bit LSB Aligned																							9	8	7	6	5	4	3	2	1	0
10-bit MSB Aligned																	9	8	7	6	5	4	3	2	1	0						
12-bit LSB Aligned																					11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

Table 52-5. Data Alignment (continued)

12-bit MSB Aligned																	11	10	9	8	7	6	5	4	3	2	1	0				
16-bit LSB Aligned																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit MSB Aligned	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
18-bit LSB Aligned															17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
18-bit MSB Aligned	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
20-bit LSB Aligned													19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
20-bit MSB Aligned	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
22-bit LSB Aligned											21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
22-bit MSB Aligned	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
24-bit LSB Aligned									23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit MSB Aligned	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

In addition, receive data can either be zero-extended or sign-extended if LSB alignment is selected. With zero-extension, all bits above the most significant bit are 0's. This format is useful when data is stored in a pure integer format. With sign-extension, all bits above the most significant bit are equal to the most significant bit. This format is useful when data is stored in a fixed-point integer format (which implies fractional values). Receive data extension is controlled by the RXEXT bit in the SSI.SRCR. Transmit data used with LSB alignment has no concept of sign/zero-extension. Unused bits above the most significant bit are simply ignored.

When configured in I2S or AC97 mode, the SSI forces the selection of LSB alignment. However, RXEXT still permits a choice between zero-extension and sign-extension.

Refer to SSI Transmit Configuration Register (SSI_SSI_STCR) and SSI Receive Configuration Register (SSI_SSI_SRCR) for more details on the relevant bits in the SSI.STCR and SSI.SRCR registers.

52.7.3 SSI Architecture

The Synchronous Serial Interface (SSI) is connected to chip pads through the Digital Audio Mux (AUDMUX) block or directly as well. The AUDMUX can be configured to connect the SSI to the chip pads in various ways.

Refer to Figure 52-1 for a block diagram of the SSI.

52.7.4 SSI Clocking

The SSI uses the following clocks:

- Bit clock - Used to serially clock the data bits in and out of the SSI port. This clock is either generated internally (from SSI's sys clock) or taken from external clock source (through the Tx/Rx clock ports).
- Word clock - Used to count the number of data bits per word (8, 10, 12, 16, 18, 20, 22 or 24 bits). This clock is generated internally from the bit clock.
- Frame clock (Frame Sync) - Used to count the number of words in a frame. This signal can be generated internally from the bit clock, or taken from external source (from the Tx/Rx frame sync ports).
- Network clock -In master mode, this is an integer multiple of frame clock. This is oversampling clock. It is used in cases when SSI has to provide the clock.

Care should be taken to ensure that the bit clock frequency (either internally generated by dividing the SSI's sys clock or sourced from external device through Tx/Rx clock ports) is never greater than 1/5 of the ipg_clk (from CCM) frequency.

In Normal mode (SCR[6:5]=00), the bit clock, used to serially clock the data, is visible on the Serial Transmit Clock (STCK) and Serial Receive Clock (SRCK) ports. The word clock is an internal clock used to determine when transmission of an 8, 10, 12, 16, 18, 20, 22 or 24 bit word has completed. The word clock in turn then clocks the frame clock, which counts the number of words in the frame. The frame clock can be viewed on the STFS and SRFS frame sync ports, because a frame sync is generated after the correct number of words in the frame have passed. In master and synchronous mode, the unused port SRCK is used as network clock (oversampling clock) enabled by the SCR register bit 15, SYS_CLK_EN. This network clock is an oversampling clock of the frame sync clock (STFS). In this mode, the word length (WL), Prescaler Range (PSR), Prescaler Modulus (PM) and Frame rate (DC) selects the ratio of network clock to sampling clock STFS. In case of I2S mode, the network clock (oversampling clock) can be made available on this port if the SYS_CLK_EN bit is set. The relationship between the clocks and the dividers is shown in the figure below ("SSI Clocking"). The bit clock can be received from an SSI clock port or can be generated from the network clock through a divider, as shown in Figure 52-22 ("SSI Transmit Clock Generator Block Diagram").

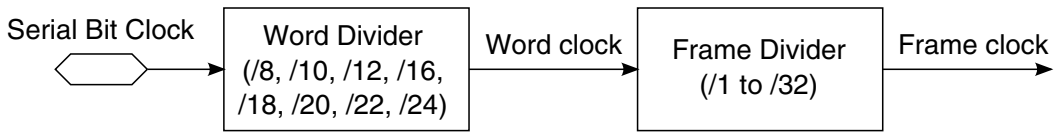


Figure 52-21. SSI Clocking

52.7.4.1 SSI Clock and Frame Sync Generation

Data clock and frame sync signals can be generated internally, or can be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the SSI's sys clock. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation.

In Gated Clock mode, the data clock is valid only when data is being transmitted. Otherwise the clock port is pulled to the inactive state. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

The following figure shows a block diagram of the clock generator for the transmit section. The serial bit clock can be internal or external, depending on the Transmit Direction (TXDIR) bit in the SSI Transmit Configuration Register (SSI.STCR). The receive section contains an equivalent clock generator circuit.

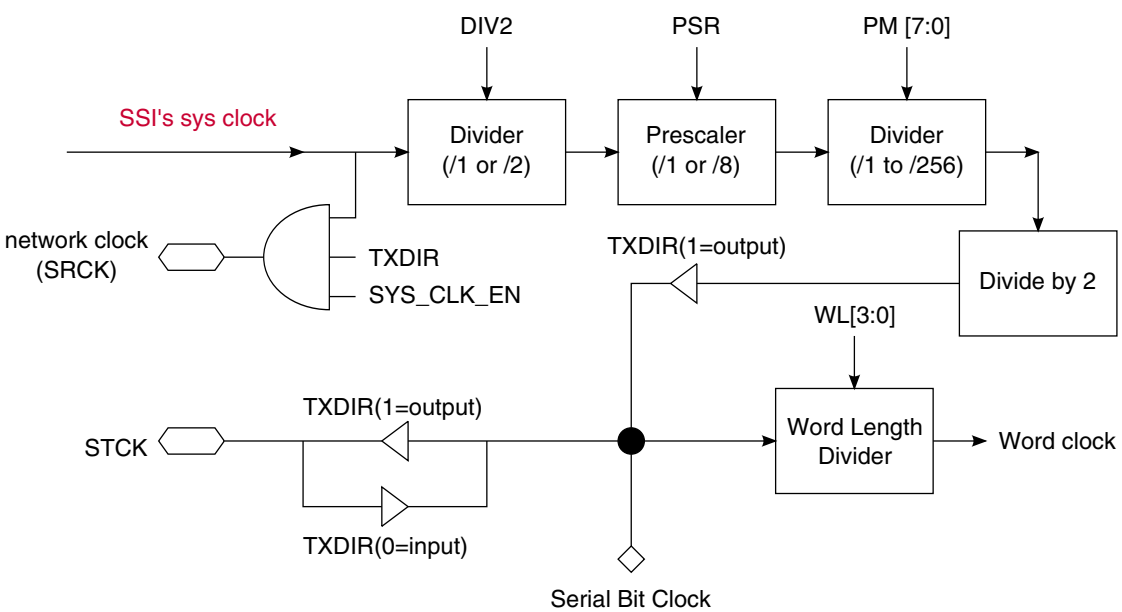


Figure 52-22. SSI Transmit Clock Generator Block Diagram

The figure below shows the Frame Sync Generator block for the transmit section. When internally generated, both receive and transmit frame sync are generated from the word clock and are defined by the Frame Rate Divider (DC[4:0]) bits and the Word Length (WL[3:0]) bits of the SSI Transmit Clock Control Register (SSI.STCCR). The receive section contains an equivalent circuit for the Frame Sync Generator.

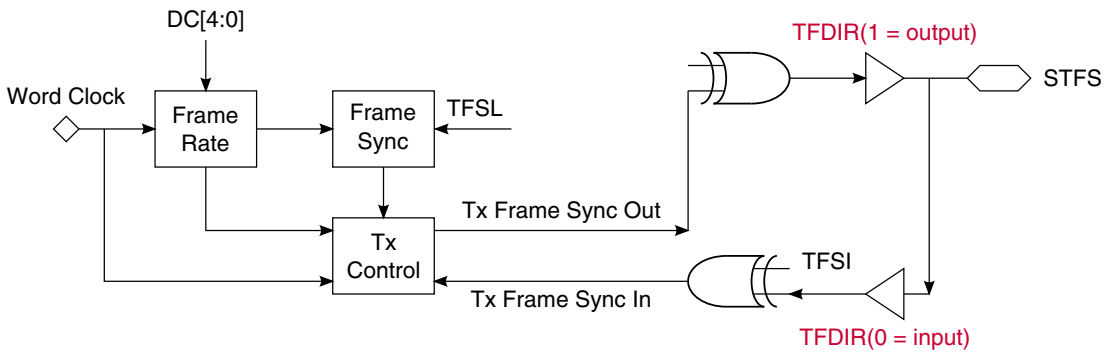


Figure 52-23. SSI Transmit Frame Sync Generator Block Diagram

52.7.4.2 DIV2, PSR and PM Bit Description

The bit clock frequency can be calculated from the SSI's sys clock using the equation in the following figure.

NOTE

You must ensure that the bit-clock frequency must be never greater than 1/5 of the peripheral clock frequency. The oversampling clock frequency can go up to peripheral clock frequency. Bits DIV2, PSR and PM should not be all set to zero at the same time.

$$f_{INT\_BIT\_CLK} = f_{SSI's\ sys\ clock} / [(DIV2 + 1) \times (7 \times PSR + 1) \times (PM + 1) \times 2]$$

where PM = PM[7:0]

$$f_{FRAME\_SYN\_CLK} = (F_{INT\_BIT\_CLK}) / [(DC + 1) \times WL]$$

where DC = DC[4:0] and WL = 8, 10, 12, 16, 18, 20, 22, 24

Figure 52-24. SSI Bit Clock Equation

For example, if the SSI's sys clock is 12.288, in 8-bit word Normal mode with DC[4:0] set to 0(00000), PM[7:0] set to 47 (0010 1111), the PSR bit cleared, DIV2 bit set to 1, a bit clock rate of 12.288 Mhz / [1 x 4 x 48] = 64 kHz is generated. Since the 8-bit word rate is equal to one (i.e. normal mode), the sampling rate (FS rate) would then be 64 kHz / [1 * 8] = 8 kHz.

In next example, SSI's sys clock is 11.2896 Mhz. A 16-bit word Network mode with DC[4:0] set to 1 (00001), PM[7:0] set to 3 (0000 0011), the PSR bit is set to 0, DIV2 bit set to 0, and a 11.2896 MHz oversampling clock, a bit clock rate of 11.2896 Mhz / [1 x 2 x 4] = 1.4112 MHz is generated. Since the 16-bit word rate is equal to two, the sampling rate (FS rate) would be 1.4112 MHz / [2 * 16] = 44.1 kHz.

The table below shows programming examples for the clock dividers in the CCM and the SSI to support various bit clock (STCK) frequencies.

Table 52-6. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2

Bits/ Word	Words / Frame	Ideal Frame Rate (kHz)	PLL Freq (Mhz)	SSIDIV (in CCM)	SSI's sys clock Freq (Mhz)	DIV 2	PS R	P M	W L	D C	Actual Serial bit clock Freq (kHz) STCK	Target Serial bit clock Freq (kHz) STCK	Erro r (Hz)
16	1	8	294.912	48	12.288	0	0	47	7	0	128	128	0
16	2	8	294.912	48	12.288	0	0	23	7	1	256	256	0
16	4	8	294.912	48	12.288	0	0	11	7	3	512	512	0
16	1	12	294.912	48	12.288	0	0	31	7	0	192	192	0
16	2	12	294.912	48	12.288	0	0	15	7	1	384	384	0
16	4	12	294.912	48	12.288	0	0	7	7	3	768	768	0
16	1	16	294.912	48	12.288	0	0	23	7	0	256	256	0
16	2	16	294.912	48	12.288	0	0	11	7	1	512	512	0
16	4	16	294.912	48	12.288	0	0	5	7	3	1024	1024	0
16	1	24	294.912	48	12.288	0	0	15	7	0	384	384	0
16	2	24	294.912	48	12.288	0	0	7	7	1	768	768	0
16	4	24	294.912	48	12.288	0	0	3	7	3	1536	1536	0
16	1	32	294.912	48	12.288	0	0	11	7	0	512	512	0
16	2	32	294.912	48	12.288	0	0	5	7	1	1024	1024	0
16	4	32	294.912	48	12.288	0	0	2	7	3	2048	2048	0
16	1	48	294.912	48	12.288	0	0	15	7	0	768	768	0
16	2	48	294.912	48	12.288	0	0	3	7	1	1536	1536	0
16	4	48	294.912	48	12.288	0	0	1	7	3	3072	3072	0

Table continues on the next page...

Table 52-6. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2 (continued)

Bits/ Word	Words / Frame	Ideal Frame Rate (kHz)	PLL Freq (Mhz)	SSIDIV (in CCM)	SSI's sys clock Freq (Mhz)	DIV 2	PS R	P M	W L	D C	Actual Serial bit clock Freq (kHz) STCK	Target Serial bit clock Freq (kHz) STCK	Erro r (Hz)
16	1	11.025	270.9504	48	11.2896	0	0	31	7	0	176.4	176.4	0
16	2	11.025	270.9504	48	11.2896	0	0	15	7	1	352.8	352.8	0
16	4	11.025	270.9504	48	11.2896	0	0	7	7	3	705.6	705.6	0
16	1	22.05	270.9504	48	11.2896	0	0	15	7	0	352.8	352.8	0
16	2	22.05	270.9504	48	11.2896	0	0	7	7	1	705.6	705.6	0
16	4	22.05	270.9504	48	11.2896	0	0	3	7	3	1411.2	1411.2	0
16	1	44.1	270.9504	48	11.2896	0	0	7	7	0	705.6	705.6	0
16	2	44.1	270.9504	48	11.2896	0	0	3	7	1	1411.2	1411.2	0
16	4	44.1	270.9504	48	11.2896	0	0	1	7	3	2822.4	2822.4	0

NOTE

The table above describes how various frame rates can be achieved with the PLLs supplying a frequency of 294.912 MHz and 270.9504 MHz (with WL and DC settings as shown). These clocks are recommended as convenient starting points but the system allows for other input clock frequencies as well.

Table 52-6 shows programming of the CCM and SSI dividers in order to generate the appropriate network clock and serial bit clock frequencies for various sampling rates. In these examples, the master mode is selected either by setting I2S master bit (SCR[6:5]=01) or individually programming the SSI in network, synchronous, transmit internal mode (the table specifically illustrates the I2S mode frequencies/sample rates). The network clock is oversampling clock.

Note that the I2S master mode requires that a word length of 32 bits be used (regardless of the actual data type). Consequently, the fixed I2S frame rate of 64 bits per frame (word length (WL) can be any value) and DC of 1 are assumed.

52.7.5 Receive Interrupt Enable Bit Description

When the RIE and RE bit are set, the processor is interrupted when either of the SSI Receive FIFO Full (RFF0/1) bits in SSI.SISR is set (if the corresponding Receive FIFO is enabled). If the Receive FIFO is not enabled, the interrupt is generated when the corresponding SSI Receive Data Ready (RDR0/1) bit in the SSI.SISR is set.

When the receive FIFO is enabled, a maximum of 15 values are available to be read (15 values per channel in Two-Channel mode). If not enabled, then one value can be read from the SRX register (one each in case of Two-Channel mode). If the RIE bit is cleared, these interrupts are disabled. However, the RFF0/1 and RDR0/1 bits still indicate the receive data register full condition. Reading the SSI.SRX registers clears the RDR bits, thus clearing the pending interrupt. Two receive data interrupts (two per channel in case of Two-Channel mode) are available: receive data with exception status and receive data without exception. The tables below show the conditions under which these interrupts are generated.

Table 52-7. SSI Receive Data 1 Interrupts

Interrupt	RIE	ROE0	RFF0/RDR0
Receive Data 1(with Exception Status)	1	1	1
Receive Data 1(without exception)	1	0	1

Table 52-8. SSI Receive Data 0 Interrupts

Interrupt	RIE	ROE1	RFF1/RDR1
Receive Data 0 (with Exception Status)	1	1	1
Receive Data 0 (without exception)	1	0	1

52.7.6 Transmit Interrupt Enable Bit Description

The SSI Transmit Interrupt Enable (TIE) control bit determines whether the processor is interrupted when the SSI transmitter needs to be serviced.

When the TIE and TE bits are set, the program controller is interrupted when either of the SSI Transmit FIFO Empty (TFE0/1) flags in SISR are set (if corresponding Transmit FIFO is enabled). If the corresponding Transmit FIFO is not enabled, an interrupt is generated when the corresponding SSI Transmit Data Register Empty (TDE0/1) flag in the SISR is set and Transmit Enable (TE) bit is set.

When Transmit FIFO 0 is enabled, a maximum of 15 values can be written to the SSI (15 per channel in case of Two-Channel mode, using Tx FIFO 1). If not enabled, then one value can be written to the SSI.STX0 register (one per channel in case of Two-Channel mode using SSI.STX1). When the TIE bit is cleared, all transmit interrupts are disabled. However, the TDE0/1 bits always indicate the corresponding SSI.STX register empty condition, even when the transmitter is disabled by the Transmit Enable (TE) bit (in the SSI.SCR). Writing data to the STX clears the corresponding TDE bit, thus clearing the interrupt. Two transmit data interrupts are available (four in case of Two-

Channel mode, two per channel): transmit data with exception status and transmit data without exceptions. The tables below show the conditions under which these interrupts are generated.

Table 52-9. SSI Transmit Data 1 Interrupts

Interrupt	TIE	TUE1	TFE1/TDE1
Transmit Data 1 (with Exception Status)	1	1	1
Transmit Data 1 (without exception)	1	0	1

Table 52-10. SSI Transmit Data 0 Interrupts

Interrupt	TIE	TUE0	TFE0/TDE0
Transmit Data 0 (with Exception Status)	1	1	1
Transmit Data 0 (without exception)	1	0	1

52.7.7 Internal Frame and Clock Shutdown

During transmit/receive operation, disabling TE/RE will ensure that data transmission/ reception stops after current frame ends following which TFRC/RFRC Status bits will get set to indicate the Frame Completion State.

If TE is disabled 4 clock cycles before the next frame, extra frame generated are invalid frames. TFR_CLK_DIS/RFR_CLK_DIS bit is set in the current or any of the previous frames, SSI will stop driving the STFS/SRFS and STCK/SRCK signals after the current frame ends.

If TFR_CLK_DIS/RFR_CLK_DIS bit is not set, SSI will continue generating STFS/ SRFS and STCK/SRCK signals (in case direction is from SSI), which then can be disabled by writing '1' to TFR_CLK_DIS/RFR_CLK_DIS bit. SSI will then stop driving these signals after end of frame is reached following which TFRC/RFRC status bits will get set to indicate the Frame Completion State.

The following figure is an illustration of transmission case where TXDIR and TFDIR are both set to '1'. In this case TE is disabled with TFR_CLK_DIS bit set in current or any of the previous frames.

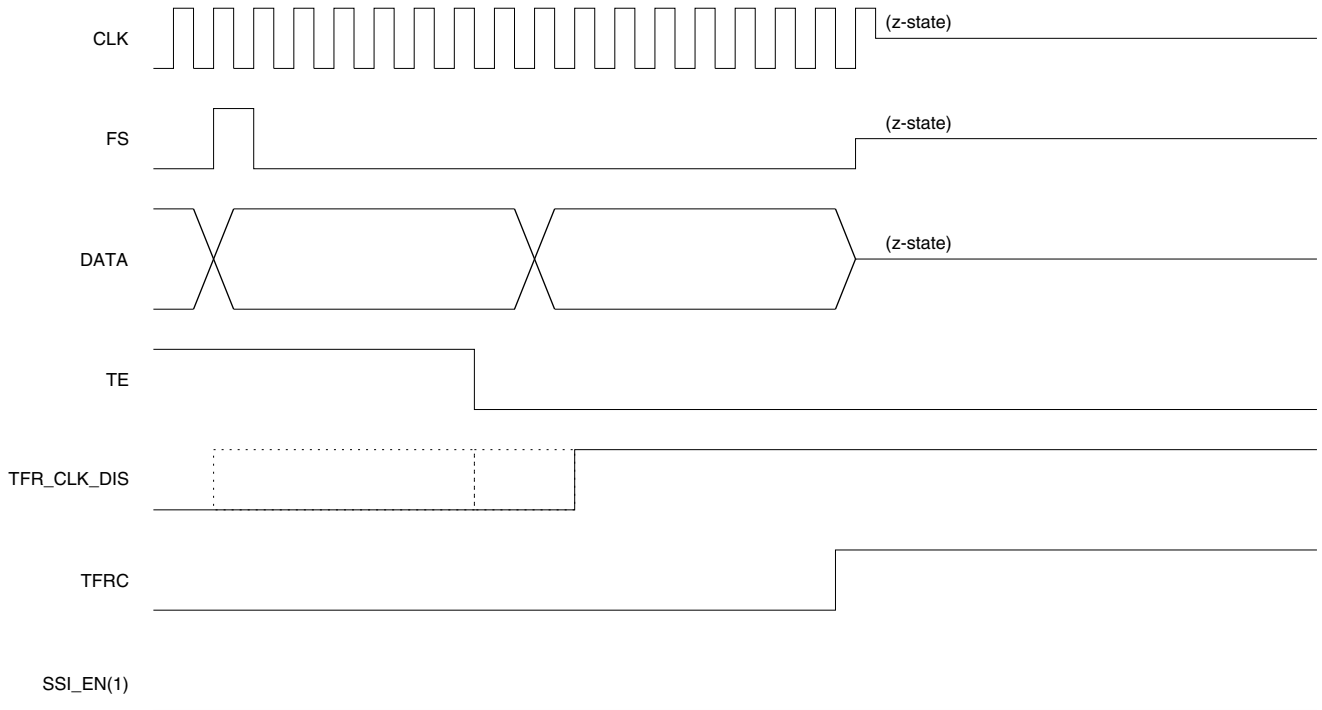


Figure 52-25. TFR_CLK_DIS assertion in current or previous frame as TE disable

The figure below is an illustration of transmission case where TXDIR and TFDIR are both set to '1'. In this case TFR_CLK_DIS bit is set after few frames of disabling TE. TFRC (Transmit Frame Complete) is set at frame boundary after TE is cleared. Once software services this interrupt and sets TFR_CLK_DIS bit later, TFRC bit is again set at next frame boundary.

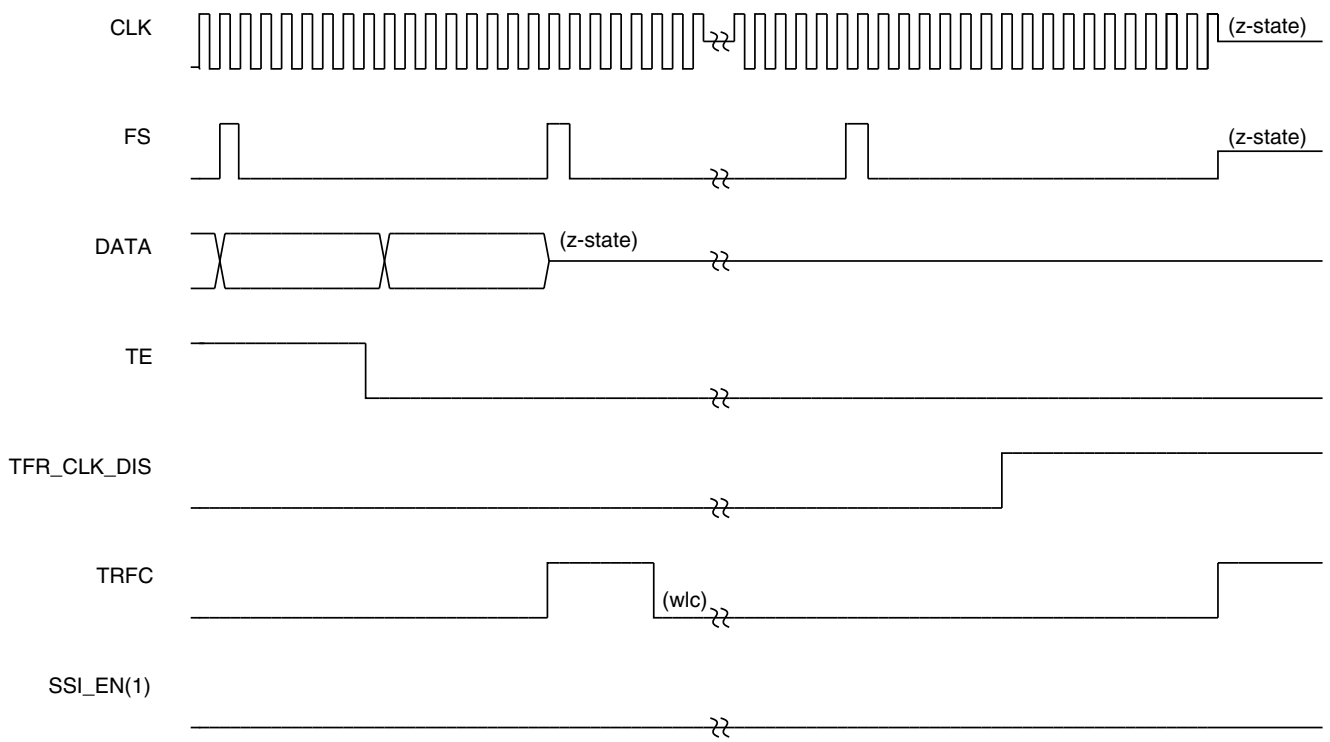


Figure 52-26. TFR_CLK_DIS assertion in subsequent frame after disabling TE

52.7.8 Peripheral Bus Interface

The SSI has a Peripheral Bus interface to provide a control and data interface. This interface is used by both the processor and DMA controller.

52.7.8.1 Transfer Lengths Supported

The Peripheral Bus interface of the SSI only supports 32-bit transfers with all SSI registers other than SSI.STX0, SSI.STX1, SSI.SRX0, and SSI.SRX1 (that is, the data registers).

With the exception of the data registers, using 8-bit and 16-bit transactions could result in undesired behavior but will not result in a transfer bus error. The data registers (SSI.STX0, SSI.STX1, SSI.SRX0, and SSI.SRX1) support 8-bit, 16-bit, and 32-bit transfer lengths without restrictions.

52.7.8.2 Transfer Bus Errors

Transfer bus errors are generated upon response to the following:

- Write transfer to a read-only register.
- Read or write access to a register space beyond the last populated register of the SSI in its memory map (up until the end of the allocated memory address range of the SSI).

52.7.8.3 Clock Rate

The Peripheral Bus clock frequency must be at least five times the serial bit clock frequency.

52.7.9 Reset

The SSI is affected by the following types of reset:

- Power-on Reset-The Power-on reset clears the SSIEN bit in SSI.SCR, which disables the SSI. All other status and control bits in the SSI are affected as described in SSI Programming Model in the "Memory Map and Register Definition section".
- SSI Reset-The SSI reset is generated when the SSIEN bit in the SSI.SCR is cleared. The SSI status bits are preset to the same state produced by the Power-on reset. The SSI control bits are unaffected. The control bits in the SSI.SCR are also unaffected. The SSI reset is useful for selective reset of the SSI without changing the present SSI control bits and without affecting the other peripherals.

52.8 Programmable Registers

SSI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5001_4000	SSI Transmit Data Register n (SSI-2_SSI_STX0)	32	R/W	0000_0000h	52.8.1/ 2955
5001_4004	SSI Transmit Data Register n (SSI-2_SSI_STX1)	32	R/W	0000_0000h	52.8.1/ 2955
5001_4008	SSI Receive Data Register n (SSI-2_SSI_SRX0)	32	R	0000_0000h	52.8.2/ 2955
5001_400C	SSI Receive Data Register n (SSI-2_SSI_SRX1)	32	R	0000_0000h	52.8.2/ 2955

Table continues on the next page...

SSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5001_4010	SSI Control Register (SSI-2_SSI_SCR)	32	R/W	0000_0000h	52.8.3/ 2956
5001_4014	SSI Interrupt Status Register (SSI-2_SSI_SISR)	32	w1c	0000_3003h	52.8.4/ 2958
5001_4018	SSI Interrupt Enable Register (SSI-2_SIER)	32	R/W	0000_3003h	52.8.5/ 2963
5001_401C	SSI Transmit Configuration Register (SSI-2_SSI_STCR)	32	R/W	0000_0200h	52.8.6/ 2965
5001_4020	SSI Receive Configuration Register (SSI-2_SSI_SRCR)	32	R/W	0000_0200h	52.8.7/ 2967
5001_4024	SSI Transmit Clock Control Register (SSI-2_SSI_STCCR)	32	R/W	0004_0000h	52.8.8/ 2969
5001_4028	SSI Receive Clock Control Register (SSI-2_SRCCR)	32	R/W	0004_0000h	52.8.9/ 2971
5001_402C	SSI FIFO Control/Status Register (SSI-2_SSI_SFCSR)	32	R/W	0081_0081h	52.8.10/ 2972
5001_4038	SSI AC97 Control Register (SSI-2_SSI_SACNT)	32	R/W	0000_0000h	52.8.11/ 2976
5001_403C	SSI AC97 Command Address Register (SSI-2_SSI_SACADD)	32	R/W	0000_0000h	52.8.12/ 2977
5001_4040	SSI AC97 Command Data Register (SSI-2_SSI_SACDAT)	32	R/W	0000_0000h	52.8.13/ 2978
5001_4044	SSI AC97 Tag Register (SSI-2_SATAG)	32	R/W	0000_0000h	52.8.14/ 2978
5001_4048	SSI Transmit Time Slot Mask Register (SSI-2_SSI_STMSK)	32	R/W	0000_0000h	52.8.15/ 2979
5001_404C	SSI Receive Time Slot Mask Register (SSI-2_SSI_SRMSK)	32	R/W	0000_0000h	52.8.16/ 2979
5001_4050	SSI AC97 Channel Status Register (SSI-2_SSI_SACCST)	32	R	0000_0000h	52.8.17/ 2980
5001_4054	SSI AC97 Channel Enable Register (SSI-2_SSI_SACCEN)	32	W	0000_0000h	52.8.18/ 2980
5001_4058	SSI AC97 Channel Disable Register (SSI-2_SSI_SACCDIS)	32	W	0000_0000h	52.8.19/ 2981
63FC_C000	SSI Transmit Data Register n (SSI-1_SSI_STX0)	32	R/W	0000_0000h	52.8.1/ 2955
63FC_C004	SSI Transmit Data Register n (SSI-1_SSI_STX1)	32	R/W	0000_0000h	52.8.1/ 2955
63FC_C008	SSI Receive Data Register n (SSI-1_SSI_SRX0)	32	R	0000_0000h	52.8.2/ 2955

Table continues on the next page...

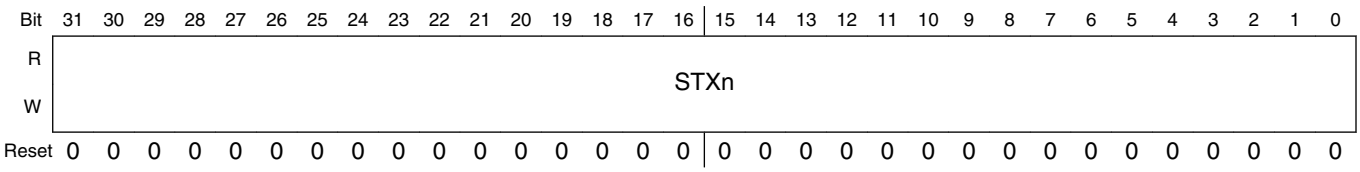
SSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63FC_C00C	SSI Receive Data Register n (SSI-1_SSI_SRX1)	32	R	0000_0000h	52.8.2/ 2955
63FC_C010	SSI Control Register (SSI-1_SSI_SCR)	32	R/W	0000_0000h	52.8.3/ 2956
63FC_C014	SSI Interrupt Status Register (SSI-1_SSI_SISR)	32	w1c	0000_3003h	52.8.4/ 2958
63FC_C018	SSI Interrupt Enable Register (SSI-1_SIER)	32	R/W	0000_3003h	52.8.5/ 2963
63FC_C01C	SSI Transmit Configuration Register (SSI-1_SSI_STCR)	32	R/W	0000_0200h	52.8.6/ 2965
63FC_C020	SSI Receive Configuration Register (SSI-1_SSI_SRCR)	32	R/W	0000_0200h	52.8.7/ 2967
63FC_C024	SSI Transmit Clock Control Register (SSI-1_SSI_STCCR)	32	R/W	0004_0000h	52.8.8/ 2969
63FC_C028	SSI Receive Clock Control Register (SSI-1_SRCCR)	32	R/W	0004_0000h	52.8.9/ 2971
63FC_C02C	SSI FIFO Control/Status Register (SSI-1_SSI_SFCSR)	32	R/W	0081_0081h	52.8.10/ 2972
63FC_C038	SSI AC97 Control Register (SSI-1_SSI_SACNT)	32	R/W	0000_0000h	52.8.11/ 2976
63FC_C03C	SSI AC97 Command Address Register (SSI-1_SSI_SACADD)	32	R/W	0000_0000h	52.8.12/ 2977
63FC_C040	SSI AC97 Command Data Register (SSI-1_SSI_SACDAT)	32	R/W	0000_0000h	52.8.13/ 2978
63FC_C044	SSI AC97 Tag Register (SSI-1_SATAG)	32	R/W	0000_0000h	52.8.14/ 2978
63FC_C048	SSI Transmit Time Slot Mask Register (SSI-1_SSI_STMSK)	32	R/W	0000_0000h	52.8.15/ 2979
63FC_C04C	SSI Receive Time Slot Mask Register (SSI-1_SSI_SRMSK)	32	R/W	0000_0000h	52.8.16/ 2979
63FC_C050	SSI AC97 Channel Status Register (SSI-1_SSI_SACCST)	32	R	0000_0000h	52.8.17/ 2980
63FC_C054	SSI AC97 Channel Enable Register (SSI-1_SSI_SACCEN)	32	W	0000_0000h	52.8.18/ 2980
63FC_C058	SSI AC97 Channel Disable Register (SSI-1_SSI_SACCDIS)	32	W	0000_0000h	52.8.19/ 2981

52.8.1 SSI Transmit Data Register n (SSIx_SSI_STX*n*)

Enable SSI (SSIEN=1) before writing to SSI Transmit Data Registers.

Addresses: SSI-2_SSI_STX0 is 5001_4000h base + 0h offset = 5001_4000h
SSI-2_SSI_STX1 is 5001_4000h base + 4h offset = 5001_4004h
SSI-1_SSI_STX0 is 63FC_C000h base + 0h offset = 63FC_C000h
SSI-1_SSI_STX1 is 63FC_C000h base + 4h offset = 63FC_C004h

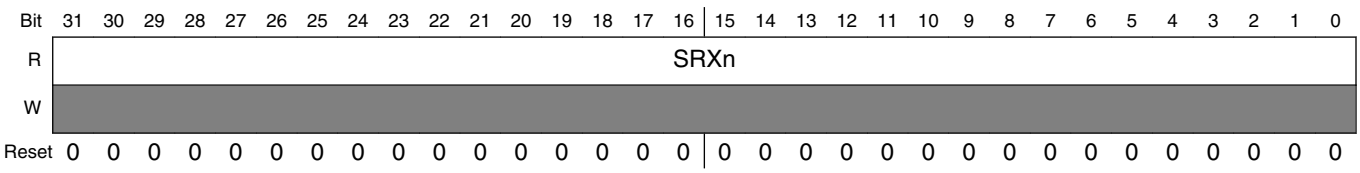


SSIx_SSI_STX*n* field descriptions

Field	Description
31–0 STXn	SSI Transmit Data. These bits store the data to be transmitted by the These are implemented as the first word of their respective Tx FIFOs. Data written to these registers is transferred to the Transmit Shift Register (TXSR), when shifting of the previous data is complete. If both FIFOs are in use, data is alternately transferred from STX0 and STX1, to TXSR. Multiple writes to the STX registers will not result in the previous data being over-written by the subsequent data. STX1 can only be used in Two-Channel mode of operation. Protection from over-writing is present irrespective of whether the transmitter is enabled or not. Example 1: If Tx FIFO0 is in use and user writes Data1...Data16 to STX0,Data16 will not over-write Data1. Data1...Data15 are stored in the FIFO whileData16 is discarded. Example 2: If Tx FIFO0 is not in use and user writes Data1, Data2 to STX0, then Data2 will not over-write Data1 and will be discarded.

52.8.2 SSI Receive Data Register n (SSIx_SSI_SRX*n*)

Addresses: SSI-2_SSI_SRX0 is 5001_4000h base + 8h offset = 5001_4008h
SSI-2_SSI_SRX1 is 5001_4000h base + Ch offset = 5001_400Ch
SSI-1_SSI_SRX0 is 63FC_C000h base + 8h offset = 63FC_C008h
SSI-1_SSI_SRX1 is 63FC_C000h base + Ch offset = 63FC_C00Ch



SSIx_SSI_SRX*n* field descriptions

Field	Description
31–0 SRXn	SSI Receive Data. These bits store the data received by the These are implemented as the first word of their respective Rx FIFOs. These bits receive data from the RXSR depending on the mode of operation.

SSIx_SSI_SRX*n* field descriptions (continued)

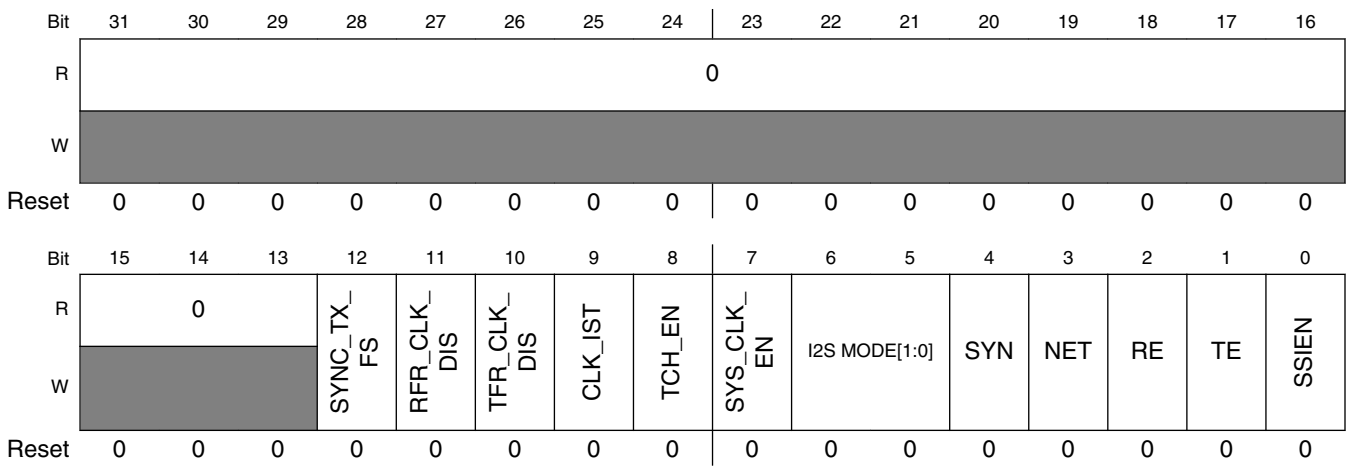
Field	Description
	In case both FIFOs are in use, data is transferred to each data register alternately. SRX1 can only be used in Two-Channel mode of operation.

52.8.3 SSI Control Register (SSIx_SSI_SCR)

The SSI Control Register (SSI_SCR) sets up the SSI reset is controlled by bit 0 in the SSI_SCR. SSI operating modes are also selected in this register (except AC97 mode which is selected in the SSI_SACNT register).

Addresses: SSI-2_SSI_SCR is 5001_4000h base + 10h offset = 5001_4010h

SSI-1_SSI_SCR is 63FC_C000h base + 10h offset = 63FC_C010h



SSIx_SSI_SCR field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value zero. Reserved
12 SYNC_TX_FS	SYNC_FS_TX bit provides a safe window for TE to be visible to the internal circuit which is just after FS occurrence. When SYNC_TX_FS is set, TE(SCR[1]) gets latched on FS occurrence & latched TE is used to enable/disable SSI transmitter. TE needs setup of 2 bit-clock cycles before occurrence of FS. If TE is changed within 2 bit-clock cycles of FS occurrence, there is high probability that TE will be latched on next FS. Note: With TFR_CLK_DIS feature on, TE is used directly to enable transmitter in following cases (i) Sync mode & Rx disabled (ii) Async Mode. Latched-TE is used to disable the transmitter. This bit has no relevance in gated mode and AC97 mode. 0 - TE not latched with FS occurrence & used directly for transmitter enable/disable. 1 - TE latched with FS occurrence & latched-TE used for transmitter enable/disable.
11 RFR_CLK_DIS	Receive Frame Clock Disable.

Table continues on the next page...

SSIx_SSI_SCR field descriptions (continued)

Field	Description
	This bit provides the option to keep the Frame-sync and Clock enabled or to disable them after the receive frame in which the receiver is disabled. Writing to this bit has effect only when RE is disabled.The receiver is disabled by clearing the RE bit. 0 Continue Frame-sync/Clock generation after current frame during which RE is cleared. This may be required when Frame-sync and Clocks are required from SSI, even when no data is to be received. 1 Stop Frame-sync/Clock generation at next frame boundary. This will be effective also in case where receiver is already disabled in current or previous frames.
10 TFR_CLK_DIS	Transmit Frame Clock Disable. This bit provide option to keep the Frame-sync and Clock enabled or disabled after current transmit frame, in which transmitter is disabled by clearing TE bit. Writing to this bit has effect only when SSI is enabled TE is disabled. 0 Continue Frame-sync/Clock generation after current frame during which TE is cleared. This may be required when Frame-sync and Clocks are required from SSI, even when no data is to be received. 1 Stop Frame-sync/Clock generation at next frame boundary. This will be effective also in case where transmitter is already disabled in current or previous frames.
9 CLK_IST	Clock Idle State. This bit controls the idle state of the transmit clock port during SSI internal gated mode. Note: When Clock idle state is '1' the clock polarity should always be negedge triggered and when Clock idle = '0' the clock polarity should always be positive edge triggered. 0 Clock idle state is '0'. 1 Clock idle state is '1'.
8 TCH_EN	Two-Channel Operation Enable. This bit allows SSI to operate in the two-channel mode.In this mode while receiving, the RXSR transfers data to SRX0 and SRX1 alternately and while transmitting, data is alternately transferred from STX0 and STX1 to TXSR. For an even number of slots, Two-Channel Operation can be enabled to optimize usage of both FIFOs or disabled as in the case of odd number of active slots. This feature is especially useful in I2S mode, where data for Left Speaker can be placed in Tx-FIFO0 and for Right speaker in Tx-FIFO1. 0 Two-channel mode disabled. 1 Two-channel mode enabled.
7 SYS_CLK_EN	Network Clock (Oversampling Clock) Enable. When set, this bit allows the SSI to output the network clock at the SRCK port, provided that synchronous mode, and transmit internal clock mode are set. The relationship between bit clock and network clock is determined by DIV2, PSR, and PM bits. This feature is especially useful in I2S Master mode to output network clock (oversampling clock) on SRCK port. 0 network clock not output on SRCK port. 1 network clock output on SRCK port.
6–5 I2S MODE[1:0]	I2S Mode Select. These bits allow the SSI to operate in Normal, I2S Master or I2S Slave mode. Refer to I2S Mode for a detailed description of I2S Mode of operation. Refer to Table 52-4 for details regarding settings.
4 SYN	Synchronous Mode. This bit controls whether SSI is in synchronous mode or not. In synchronous mode, the transmit and receive sections of SSI share a common clock port (STCK) and frame sync port (STFS). 0 Asynchronous mode selected. 1 Synchronous mode selected.
3 NET	Network Mode. This bit controls whether SSI is in network mode or not.

Table continues on the next page...

SSIx_SSI_SCR field descriptions (continued)

Field	Description
	0 Network mode not selected. 1 Network mode selected.
2 RE	Receive Enable. This control bit enables the receive section of the When this bit is enabled, data reception starts with the arrival of the next frame sync. If data is being received when this bit is cleared, data reception continues until the end of the current frame and then stops. If this bit is set again before the second to last bit of the last time slot in the current frame, then reception continues without interruption. RE should not be toggled in the same frame. 0 Receive section disabled. 1 Receive section enabled.
1 TE	Transmit Enable. This control bit enables the transmit section of the It enables the transfer of the contents of the STX registers to the TXSR and also enables the internal transmit clock. The transmit section is enabled when this bit is set and a frame boundary is detected. When this bit is cleared, the transmitter continues to send data until the end of the current frame and then stops. Data can be written to the STX registers with the TE bit cleared (the corresponding TDE bit will be cleared). If the TE bit is cleared and then set again before the second to last bit of the last time slot in the current frame, data transmission continues without interruption. The normal transmit enable sequence is to write data to the STX register(s) and then set the TE bit. The normal disable sequence is to clear the TE and TIE bits after the TDE bit is set. In gated clock mode, clearing the TE bit results in the clock stopping after the data currently in TXSR has shifted out. When the TE bit is set, the clock starts immediately (for internal gated clock mode). TE should not be toggled in the same frame. After enabling/disabling transmission, SSI expects 4 setup clock cycles before arrival of frame-sync for frame-sync to be accepted/rejected by In case of fewer clock cycles, there is high probability of the frame-sync to get missed. Note: If continuos clock is not provided, SSI expects 6 clock cycles before arrival of frame-sync for frame-sync to be accepted by 0 Transmit section disabled. 1 Transmit section enabled.
0 SSIEN	SSIEN - SSI Enable This bit is used to enable/disable the When disabled, all SSI status bits are preset to the same state produced by the power-on reset, all control bits are unaffected, the contents of Tx and Rx FIFOs are cleared. When SSI is disabled, all internal clocks are disabled (except register access clock). 0 SSI is disabled. 1 SSI is enabled.

52.8.4 SSI Interrupt Status Register (SSIx_SSI_SISR)

The SSI Interrupt Status Register (SSI_SISR) is used to monitor the This register is used by the core to interrogate the status of the In gated mode of operation the TFS, RFS, TLS, RLS, TFRC and RFRC bits of AISR register are not generated. The status bits are described in the following table.

- SSI Status flags are valid when SSI is enabled.

- See Receive Interrupt Enable Bit Description and Transmit Interrupt Enable Bit Description for interrupt source mapping.
- All the flags in the SSI_SISR are updated after the first bit of the next SSI word has completed transmission or reception. Certain status bits (ROE0/1 and TUE0/1) are cleared by writing 1 to the corresponding interrupt status bit in SSI_SISR.

Addresses: SSI-2_SSI_SISR is 5001_4000h base + 14h offset = 5001_4014h

SSI-1_SSI_SISR is 63FC_C000h base + 14h offset = 63FC_C014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							RFRC	TFRC	0				CMDAU	CMDDU	RXT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE0
W					w1c	w1c	w1c	w1c								
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

SSIx_SSI_SISR field descriptions

Field	Description
31–25 Reserved	This read-only field is reserved and always has the value zero. Reserved
24 RFRC	Receive Frame Complete. This flag is set at the end of the frame during which Receiver is disabled. If Receive Frame & Clock are not disabled in the same frame, this flag is also set at the end of the frame in which Receive Frame & Clock are disabled. See description of RFR_CLK_DIS (SSI Control Register (SSI_SSI_SCR)) bit for more details on how to disable Receiver Frame & Clock or keep them enabled after receiver is disabled. 0 End of Frame not reached 1 End of frame reached after disabling RE or disabling RFR_CLK_DIS, when receiver is already disabled.
23 TFRC	Transmit Frame Complete. This flag is set at the end of the frame during which Transmitter is disabled. If Transmit Frame & Clock are not disabled in the same frame, this flag is also set at the end of the frame in which Transmit Frame & Clock are disabled. See description of TFR_CLK_DIS (SSI Control Register (SSI_SSI_SCR)) bit for more details on how to disable Transmit Frame & Clock or keep them enabled after transmitter is disabled.

Table continues on the next page...

SSIx_SSI_SISR field descriptions (continued)

Field	Description
	0 End of Frame not reached 1 End of frame reached after disabling TE or disabling TFR_CLK_DIS, when transmitter is already disabled.
22–19 Reserved	This read-only field is reserved and always has the value zero. Reserved
18 CMDAU	Command Address Register Updated. This bit causes the Command Address Updated interrupt (when CMDAU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Address. This bit is cleared on reading the SACADD register. 0 No change in SACADD register. 1 SACADD register updated with different value.
17 CMDDU	Command Data Register Updated. This bit causes the Command Data Updated interrupt (when CMDDU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Data. This bit is cleared on reading the SACDAT register. 0 No change in SACDAT register. 1 SACDAT register updated with different value.
16 RXT	Receive Tag Updated. This status bit is set each time there is a difference in the previous and current value of the received tag. It causes the Receive Tag Interrupt (if RXT_EN bit is set). This bit is cleared on reading the SATAG register. 0 No change in SATAG register. 1 SATAG register updated with different value.
15 RDR1	Receive Data Ready 1. This flag bit is set when SRX1 or Rx FIFO 1 is loaded with a new value and Two-Channel mode is selected. RDR1 is cleared when the Core reads the SRX1 register. If Rx FIFO 1 is enabled, RDR1 is cleared when the FIFO is empty. If RIE and RDR1_EN are set, a Receive Data 1 interrupt request is issued on setting of RDR1 bit in case Rx FIFO1 is disabled, if the FIFO is enabled, the interrupt is issued on RFF1 assertion. The RDR1 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
14 RDR0	Receive Data Ready 0. This flag bit is set when SRX0 or Rx FIFO 0 is loaded with a new value. RDR0 is cleared when the Core reads the SRX0 register. If Rx FIFO 0 is enabled, RDR0 is cleared when the FIFO is empty. If RIE and RDR0_EN are set, a Receive Data 0 interrupt request is issued on setting of RDR0 bit in case Rx FIFO0 is disabled, if the FIFO is enabled, the interrupt is issued on RFF0 assertion. The RDR0 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
13 TDE1	Transmit Data Register Empty 1. This flag is set whenever data is transferred to TXSR from STX1 register and Two-Channel mode is selected. If Tx FIFO1 is enabled, this occurs when there is at least one empty slot in STX1 or Tx FIFO1. If Tx FIFO1 is not enabled, this occurs when the contents of STX1 are transferred to TXSR. The TDE1 bit is cleared when the Core writes to STX1. If TIE and TDE1_EN are set, an SSI Transmit Data 1 interrupt request is issued on setting of TDE1 bit. The TDE1 bit is cleared by POR and SSI reset.

Table continues on the next page...

SSIx_SSI_SISR field descriptions (continued)

Field	Description
	0 Data available for transmission. 1 Data needs to be written by the Core for transmission.
12 TDE0	Transmit Data Register Empty 0. This flag is set whenever data is transferred to TXSR from STX0 register. If Tx FIFO 0 is enabled, this occurs when there is at least one empty slot in STX0 or Tx FIFO 0. If Tx FIFO 0 is not enabled, this occurs when the contents of STX0 are transferred to TXSR. The TDE0 bit is cleared when the Core writes to STX0. If TIE and TDE0_EN are set, an SSI Transmit Data 0 interrupt request is issued on setting of TDE0 bit. The TDE0 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the Core for transmission.
11 ROE1	Receiver Overrun Error 1. This flag is set when the RXSR is filled and ready to transfer to SRX1 register or to Rx FIFO 1 (when enabled) and these are already full and Two-Channel mode is selected. If Rx FIFO 1 is enabled, this is indicated by RFF1 flag, else this is indicated by the RDR1 flag. The RXSR is not transferred in this case. The ROE1 flag causes an interrupt if RIE and ROE1_EN are set. The ROE1 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. Clearing the RE bit does not affect the ROE1 bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
10 ROE0	Receiver Overrun Error 0. This flag is set when the RXSR is filled and ready to transfer to SRX0 register or to Rx FIFO 0 (when enabled) and these are already full. If Rx FIFO 0 is enabled, this is indicated by RFF0 flag, else this is indicated by the RDR0 flag. The RXSR is not transferred in this case. The ROE0 flag causes an interrupt if RIE and ROE0_EN are set. The ROE0 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. Clearing the RE bit does not affect the ROE0 bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
9 TUE1	Transmitter Underrun Error 1. This flag is set when the TXSR is empty (no data to be transmitted), the TDE1 flag is set, a transmit time slot occurs and the SSI is in Two-Channel mode. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE1 flag causes an interrupt if TIE and TUE1_EN are set. The TUE1 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
8 TUE0	Transmitter Underrun Error 0. This flag is set when the TXSR is empty (no data to be transmitted), the TDE0 flag is set and a transmit time slot occurs. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE0 flag causes an interrupt if TIE and TUE0_EN are set. The TUE0 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit.

Table continues on the next page...

SSIx_SSI_SISR field descriptions (continued)

Field	Description
	0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
7 TFS	Transmit Frame Sync. This flag indicates the occurrence of transmit frame sync. Data written to the STX registers during the time slot when the TFS flag is set, is sent during the second time slot (in Network mode) or in the next first time slot (in Normal mode). In Network mode, the TFS bit is set during transmission of the first time slot of the frame and is then cleared when starting transmission of the next time slot. In Normal mode, this bit is high for the first time slot. This flag causes an interrupt if TIE and TFS_EN are set. The TFS bit is cleared by POR and SSI reset. 0 No Occurrence of Transmit frame sync. 1 Transmit frame sync occurred during transmission of last word written to STX registers.
6 RFS	Receive Frame Sync. This flag indicates the occurrence of receive frame sync. In Network mode, the RFS bit is set when the first slot of the frame is being received. It is cleared when the next slot begins to be received. In Normal mode, this bit is always high (When DC = 0). This flag causes an interrupt if RIE and RFS_EN are set. The RFS bit is cleared by POR and SSI reset. 0 No Occurrence of Receive frame sync. 1 Receive frame sync occurred during reception of next word in SRX registers.
5 TLS	Transmit Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last time slot of the frame. TLS is set at the start of the last transmit time slot and causes the SSI to issue an interrupt (if TIE and TLS_EN are set). TLS is not generated when frame rate is 1 in normal mode of operation. TLS is cleared when the SISR is read with this bit set. The TLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last transmit time slot of frame.
4 RLS	Receive Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last receive time slot of the frame. RLS is set at the end of the last time slot and causes the SSI to issue an interrupt (if RIE and RLS_EN are set). RLS is cleared when the SISR is read with this bit set. The RLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last receive time slot of frame.
3 RFF1	Receive FIFO Full 1. This flag is set when Rx FIFO1 is enabled, the data level in Rx FIFO1 reaches the selected Rx FIFO WaterMark 1 (RFWM1) threshold and the SSI is in Two-Channel mode. The setting of RFF1 only causes an interrupt when RIE and RFF1_EN are set, Rx FIFO1 is enabled and the Two-Channel mode is selected. RFF1 is automatically cleared when the amount of data in Rx FIFO1 falls below the threshold. The RFF1 bit is cleared by POR and SSI reset. When Rx FIFO1 contains15 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 Space available in Receive FIFO1. 1 Receive FIFO1 is full.
2 RFF0	Receive FIFO Full 0. This flag is set when Rx FIFO0 is enabled and the data level in Rx FIFO0 reaches the selected Rx FIFO WaterMark 0 (RFWM0) threshold. The setting of RFF0 only causes an interrupt when RIE and RFF0_EN are set and Rx FIFO0 is enabled. RFF0 is automatically cleared when the amount of data in Rx FIFO0 falls below the threshold. The RFF0 bit is cleared by POR and SSI reset. When Rx FIFO0 contains15 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read.

Table continues on the next page...

SSIx_SSI_SISR field descriptions (continued)

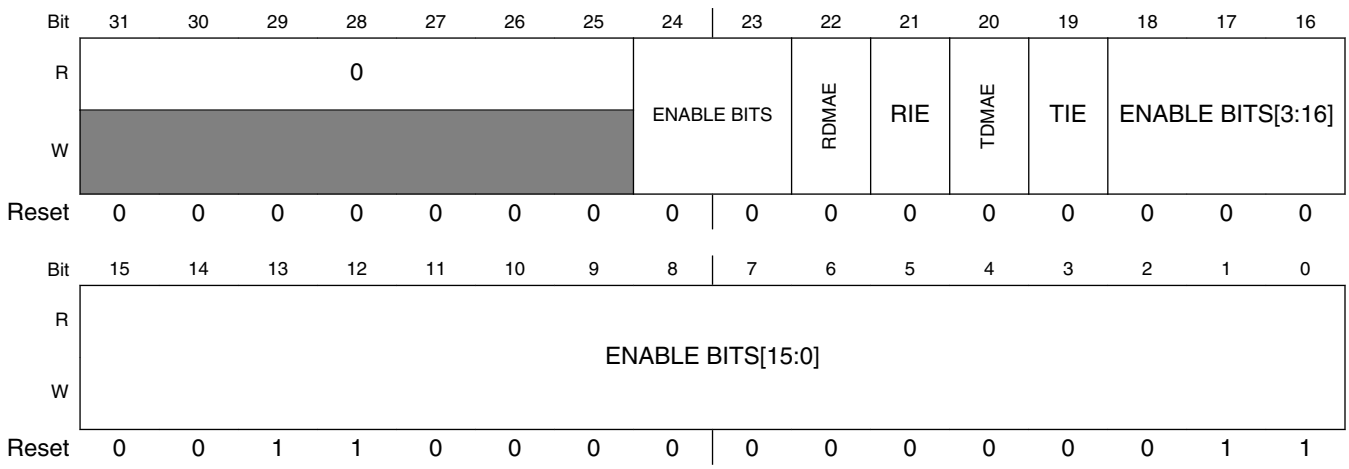
Field	Description
	0 Space available in Receive FIFO0. 1 Receive FIFO0 is full.
1 TFE1	Transmit FIFO Empty 1. This flag is set when the empty slots in Tx FIFO exceed or are equal to the selected Tx FIFO WaterMark 1 (TFWM1) threshold and the Two-Channel mode is selected. The setting of TFE1 only causes an interrupt when TIE and TFE1_EN are set, Tx FIFO1 is enabled and Two-Channel mode is selected. The TFE1 bit is automatically cleared when the data level in Tx FIFO1 becomes more than the amount specified by the watermark bits. The TFE1 bit is set by POR and SSI reset. 0 Transmit FIFO1 has data for transmission. 1 Transmit FIFO1 is empty.
0 TFE0	Transmit FIFO Empty 0. This flag is set when the empty slots in Tx FIFO exceed or are equal to the selected Tx FIFO WaterMark 0 (TFWM0) threshold. The setting of TFE0 only causes an interrupt when TIE and TFE0_EN are set and Tx FIFO0 is enabled. The TFE0 bit is automatically cleared when the data level in Tx FIFO0 becomes more than the amount specified by the watermark bits. The TFE0 bit is set by POR and SSI reset. 0 Transmit FIFO0 has data for transmission. 1 Transmit FIFO0 is empty.

52.8.5 SSI Interrupt Enable Register (SSIx_SIER)

The SSI Interrupt Enable Register (SIER) is a 25-bit register used to set up the SSI interrupts and DMA requests.

Addresses: SSI-2_SIER is 5001_4000h base + 18h offset = 5001_4018h

SSI-1_SIER is 63FC_C000h base + 18h offset = 63FC_C018h



SSIx_SIER field descriptions

Field	Description
31–25 Reserved	This read-only field is reserved and always has the value zero. Reserved

Table continues on the next page...

SSIx_SIER field descriptions (continued)

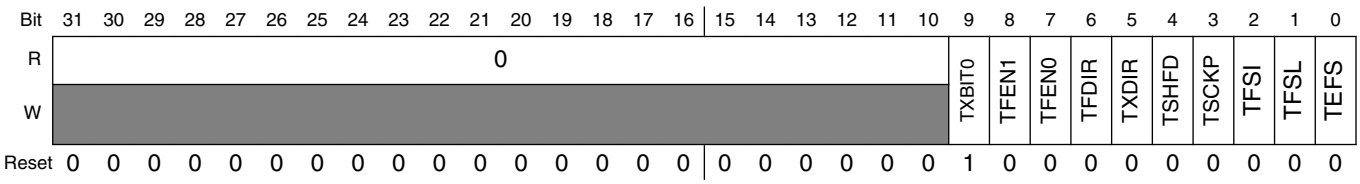
Field	Description
24–23 ENABLE BITS	Enable Bit. Each bit controls whether the corresponding status bit in SISR can issue an interrupt to the Core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
22 RDMAE	Receive DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the RFF0/1 bits in the SISR are set and if the corresponding RFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding RDR bit is set. 0 SSI Receiver DMA requests disabled. 1 SSI Receiver DMA requests enabled.
21 RIE	Receive Interrupt Enable. This control bit allows the SSI to issue receiver related interrupts to the Core. Refer to Receive Interrupt Enable Bit Description for a detailed description of this bit. 0 SSI Receiver Interrupt requests disabled. 1 SSI Receiver Interrupt requests enabled.
20 TDMAE	Transmit DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the TFE0/1 bits in the SISR are set and if the corresponding TFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding TDE bit is set. 0 SSI Transmitter DMA requests disabled. 1 SSI Transmitter DMA requests enabled.
19 TIE	Transmit Interrupt Enable. This control bit allows the SSI to issue transmitter data related interrupts to the Core. Refer to Transmit Interrupt Enable Bit Description for a detailed description of this bit. 0 SSI Transmitter Interrupt requests disabled. 1 SSI Transmitter Interrupt requests enabled.
18–0 ENABLE BITS	Enable Bit. Each bit controls whether the corresponding status bit in SISR can issue an interrupt to the Core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.

52.8.6 SSI Transmit Configuration Register (SSIx_SSI_STCR)

The SSI Transmit Configuration Register (SSI_STCR) is a read/write control registers used to direct the transmit operation of the STCR controls the direction of the bit clock and frame sync ports, STCK and STFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power-on reset clears all SSI_STCR bits. However, SSI reset does not affect the SSI_STCR bits. The SSI_STCR bits are described in the following paragraphs. See the Programmable Registers section for the programming model of the SSI. The SSI Control Register (SSI_SCR) must first be set to enable interrupts. Next, the SSI interrupt bit in the Interrupt Enable Register (SSI_SIER) must be set to enable the interrupt. Finally, the interrupt can be enabled from within the

Addresses: SSI-2_SSI_STCR is 5001_4000h base + 1Ch offset = 5001_401Ch

SSI-1_SSI_STCR is 63FC_C000h base + 1Ch offset = 63FC_C01Ch



SSIx_SSI_STCR field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9 TXBIT0	Transmit Bit 0. This control bit allows SSI to transmit the data word from bit position 0 or 15/31 in the transmit shift register. The shifting data direction can be MSB or LSB first, controlled by the TSHFD bit. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of transmit shift register (MSB aligned). 1 Shifting with respect to bit 0 of transmit shift register (LSB aligned).
8 TFEN1	Transmit FIFO Enable 1. This bit enables transmit FIFO 1. When enabled, the FIFO allows15 samples to be transmitted by the SSI (per channel) (a 9th sample can be shifting out) before TDE1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 1 disabled. 1 Transmit FIFO 1 enabled.
7 TFEN0	Transmit FIFO Enable 0. This bit enables transmit FIFO 0. When enabled, the FIFO allows15 samples to be transmitted by the SSI per channel (a 9th sample can be shifting out) before TDE0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 0 disabled. 1 Transmit FIFO 0 enabled.

Table continues on the next page...

SSIx_SSI_STCR field descriptions (continued)

Field	Description
6 TFDIR	Transmit Frame Direction. This bit controls the direction and source of the transmit frame sync signal. Internally generated frame sync signal is sent out through the STFS port and external frame sync is taken from the same port. 0 Frame Sync is external. 1 Frame Sync generated internally.
5 TXDIR	Transmit Clock Direction. This bit controls the direction and source of the clock signal used to clock the TXSR. Internally generated clock is output through the STCK port. External clock is taken from this port. Refer to Table 52-2 for details of clock pin configurations. 0 Transmit Clock is external. 1 Transmit Clock generated internally.
4 TSHFD	Transmit Shift Direction. This bit controls whether the MSB or LSB will be transmitted first in a sample. **NOTE:** The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (TSHFD cleared). 0 Data transmitted MSB first. 1 Data transmitted LSB first.
3 TSCKP	Transmit Clock Polarity. This bit controls which bit clock edge is used to clock out data for the transmit section. Note: TSCKP is 0 CLK_IST = 0; TSCKP is 1CLK_IST = 1 0 Data clocked out on rising edge of bit clock. 1 Data clocked out on falling edge of bit clock.
2 TFSI	Transmit Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the transmit section of SSI. 0 Transmit frame sync is active high. 1 Transmit frame sync is active low.
1 TFSL	Transmit Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the transmit section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Transmit frame sync is one-word long. 1 Transmit frame sync is one-clock-bit long.
0 TEFS	Transmit Early Frame Sync. This bit controls when the frame sync is initiated for the transmit section. The frame sync signal is deasserted after one bit-for-bit length frame sync and after one word-for-word length frame sync. In case of synchronous operation, the frame sync can also be initiated on receiving the first bit of data. 0 Transmit frame sync initiated as the first bit of data is transmitted. 1 Transmit frame sync is initiated one bit before the data is transmitted.

52.8.7 SSI Receive Configuration Register (SSIx_SSI_SRCR)

The SSI Receive Configuration Register (SSI_SRCR) is a read/write control registers used to direct the receive operation of the SSI. SSI.SRCR controls the direction of the bit clock and frame sync ports, SRCK and SRFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power-on reset clears all SSI_SRCR bits. However, SSI reset does not affect the SSI_SRCR bits.

Addresses: SSI-2_SSI_SRCR is 5001_4000h base + 20h offset = 5001_4020h

SSI-1_SSI_SRCR is 63FC_C000h base + 20h offset = 63FC_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																					RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

SSIx_SSI_SRCR field descriptions

Field	Description
31–11 Reserved	This read-only field is reserved and always has the value zero. Reserved
10 RXEXT	Receive Data Extension. This control bit allows SSI to store the received data word in sign extended form. This bit affects data storage only in case received data is LSB aligned (SRCR[9]=1) 0 Sign extension turned off. 1 Sign extension turned on.
9 RXBIT0	Receive Bit 0. This control bit allows SSI to receive the data word at bit position 0 or 15/31 in the receive shift register. The shifting data direction can be MSB or LSB first, controlled by the RSHFD bit. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of receive shift register (MSB aligned). 1 Shifting with respect to bit 0 of receive shift register (LSB aligned).
8 RFEN1	Receive FIFO Enable 1. This bit enables receive FIFO 1. When enabled, the FIFO allows15 samples to be received by the SSI per channel (a 16th sample can be shifting in) before RDR1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 1 disabled. 1 Receive FIFO 1 enabled.
7 RFEN0	Receive FIFO Enable 0. This bit enables receive FIFO 0. When enabled, the FIFO allows15 samples to be received by the SSI (per channel) (a 16th sample can be shifting in) before RDR0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 0 disabled. 1 Receive FIFO 0 enabled.

Table continues on the next page...

SSIx_SSI_SRCR field descriptions (continued)

Field	Description
6 RFDIR	Receive Frame Direction. This bit controls the direction and source of the receive frame sync signal. Internally generated frame sync signal is sent out through the SRFS port and external frame sync is taken from the same port. 0 Frame Sync is external. 1 Frame Sync generated internally.
5 RXDIR	Receive Clock Direction. This bit controls the direction and source of the clock signal used to clock the RXSR. Internally generated clock is output through the SRCK port. External clock is taken from this port. Refer to Table 52-2 for details on clock pin configurations. 0 Receive Clock is external. 1 Receive Clock generated internally.
4 RSHFD	Receive Shift Direction. This bit controls whether the MSB or LSB will be received first in a sample. **NOTE:** The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (RSHFD cleared). 0 Data received MSB first. 1 Data received LSB first.
3 RSCKP	Receive Clock Polarity. This bit controls which bit clock edge is used to latch in data for the receive section. 0 Data latched on falling edge of bit clock. 1 Data latched on rising edge of bit clock.
2 RFSI	Receive Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the receive section of SSI. 0 Receive frame sync is active high. 1 Receive frame sync is active low.
1 RFSL	Receive Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the receive section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Receive frame sync is one-word long. 1 Receive frame sync is one-clock-bit long.
0 REFS	Receive Early Frame Sync. This bit controls when the frame sync is initiated for the receive section. The frame sync is disabled after one bit-for-bit length frame sync and after one word-for-word length frame sync. 0 Receive frame sync initiated as the first bit of data is received. 1 Receive frame sync is initiated one bit before the data is received.

52.8.8 SSI Transmit Clock Control Register (SSIx_SSI_STCCR)

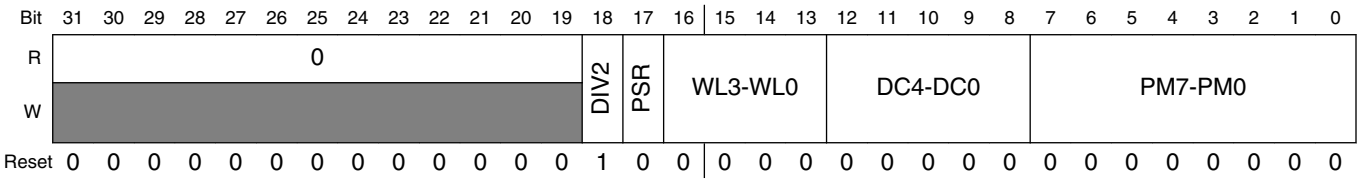
The SSI Transmit and Receive Control (SSI_STCCR and SSI_SRCCR) registers are 19-bit, read/write control registers used to direct the operation of the SSI. The Clock Controller Module (CCM) can source the SSI clock (SSI's sys clock-from CCM's ssi_clk_root) from multiple sources and perform fractional division to support commonly used audio bit rates. The CCM can maintain the SSI's sys clock frequency at a constant rate even in cases where the ipg_clk (from CCM) frequency changes. These registers control the SSI clock generator, bit and frame sync rates, word length, and number of words per frame for the serial data. The SSI.STCCR register is dedicated to the transmit section, and the SSI_SRCCR register is dedicated to the receive section except in Synchronous mode, in which the SSI_STCCR register controls both the receive and transmit sections. Power-on reset clears all SSI_STCCR and SSI_SRCCR bits. SSI reset does not affect the SSI_STCCR and SSI_SRCCR bits. The control bits are described in the following paragraphs. Although the bit patterns of the SSI_STCCR and SSI_SRCCR registers are the same, the contents of these two registers can be programmed differently.

Table 52-51. SSI Data Length

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
0	0	0	0	2	No
0	0	0	1	4	No
0	0	1	0	6	No
0	0	1	1	8	Yes
0	1	0	0	10	Yes
0	1	0	1	12	Yes
0	1	1	0	14	No
0	1	1	1	16	Yes
1	0	0	0	18	Yes
1	0	0	1	20	Yes
1	0	1	0	22	Yes
1	0	1	1	24	Yes
1	1	0	0	26	No
1	1	0	1	28	No
1	1	1	0	30	No
1	1	1	1	32	No

Addresses: SSI-2_SSI_STCCR is 5001_4000h base + 24h offset = 5001_4024h

SSI-1_SSI_STCCR is 63FC_C000h base + 24h offset = 63FC_C024h



SSIx_SSI_STCCR field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value zero. Reserved
18 DIV2	Divide By 2. This bit controls a divide-by-two divider in series with the rest of the prescalers. 0 Divider bypassed. 1 Divider used to divide clock by 2.
17 PSR	Prescaler Range. This bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is required. 0 Prescaler bypassed. 1 Prescaler used to divide clock by 8.
16–13 WL3-WL0	Word Length Control. These bits are used to control the length of the data words being transferred by the SSI. These bits control the Word Length Divider in the Clock Generator. They also control the frame sync pulse length when the FSL bit is cleared. In I2S Master mode, the SSI works with a fixed word length of 32, and the WL bits are used to control the amount of valid data in those 32 bits. In AC97 Mode of operation, if word length is set to any value other than 16 bits, it will result in a word length of 20 bits.
12–8 DC4-DC0	Frame Rate Divider Control. These bits are used to control the divide ratio for the programmable frame rate dividers. The divide ratio works on the word clock. In Normal mode, this ratio determines the word transfer rate. In Network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 in Normal mode and from 2 to 32 in Network mode. In Normal mode, a divide ratio of 1 (DC=00000) provides continuous periodic data word transfer. A bit-length frame sync must be used in this case. These bits can be programmed with values ranging from "00000" to "11111" to control the number of words in a frame.
7–0 PM7-PM0	Prescaler Modulus Select. These bits control the prescale divider in the clock generator. This prescaler is used only in Internal Clock mode to divide the internal clock. The bit clock output is available at the clock port. A divide ratio from 1 to 256 (PM[7:0] = 0x00 to 0xFF) can be selected. Refer to DIV2, PSR and PM Bit Description for details regarding settings.

52.8.9 SSI Receive Clock Control Register (SSIx_SRCCR)

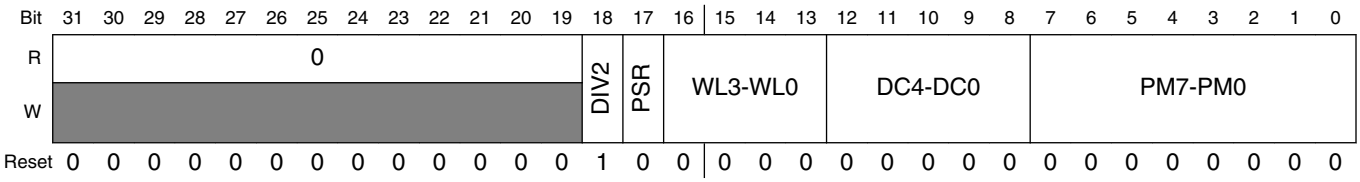
The SSI Transmit and Receive Control (SSI_STCCR and SSI_SRCCR) registers are 19-bit, read/write control registers used to direct the operation of the SSI. The Clock Controller Module (CCM) can source the SSI clock (SSI's sys clock-from CCM's ssi_clk_root) from multiple sources and perform fractional division to support commonly used audio bit rates. The CCM can maintain the SSI's sys clock frequency at a constant rate even in cases where the ipg_clk from CCM frequency changes. These registers control the SSI clock generator, bit and frame sync rates, word length, and number of words per frame for the serial data. The SSI_STCCR register is dedicated to the transmit section, and the SSI_SRCCR register is dedicated to the receive section except in Synchronous mode, in which the SSI_STCCR register controls both the receive and transmit sections. Power-on reset clears all SSI_STCCR and SSI_SRCCR bits. SSI reset does not affect the SSI_STCCR and SSI_SRCCR bits. The control bits are described in the following paragraphs. Although the bit patterns of the SSI_STCCR and SSI_SRCCR registers are the same, the contents of these two registers can be programmed differently.

Table 52-53. SSI Data Length

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
0	0	0	0	2	No
0	0	0	1	4	No
0	0	1	0	6	No
0	0	1	1	8	Yes
0	1	0	0	10	Yes
0	1	0	1	12	Yes
0	1	1	0	14	No
0	1	1	1	16	Yes
1	0	0	0	18	Yes
1	0	0	1	20	Yes
1	0	1	0	22	Yes
1	0	1	1	24	Yes
1	1	0	0	26	No
1	1	0	1	28	No
1	1	1	0	30	No
1	1	1	1	32	No

Addresses: SSI-2_SRCCR is 5001_4000h base + 28h offset = 5001_4028h

SSI-1_SRCCR is 63FC_C000h base + 28h offset = 63FC_C028h



SSIx_SRCCR field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value zero. Reserved
18 DIV2	Divide By 2. This bit controls a divide-by-two divider in series with the rest of the prescalers. 0 Divider bypassed. 1 Divider used to divide clock by 2.
17 PSR	Prescaler Range. This bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is required. 0 Prescaler bypassed. 1 Prescaler used to divide clock by 8.
16–13 WL3-WL0	Word Length Control. These bits are used to control the length of the data words being transferred by the SSI. These bits control the Word Length Divider in the Clock Generator. They also control the frame sync pulse length when the FSL bit is cleared. In I2S Master mode, the SSI works with a fixed word length of 32, and the WL bits are used to control the amount of valid data in those 32 bits. In AC97 Mode of operation, if word length is set to any value other than 16 bits, it will result in a word length of 20 bits.
12–8 DC4-DC0	Frame Rate Divider Control. These bits are used to control the divide ratio for the programmable frame rate dividers. The divide ratio works on the word clock. In Normal mode, this ratio determines the word transfer rate. In Network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 in Normal mode and from 2 to 32 in Network mode. In Normal mode, a divide ratio of 1 (DC=00000) provides continuous periodic data word transfer. A bit-length frame sync must be used in this case. These bits can be programmed with values ranging from "00000" to "11111" to control the number of words in a frame.
7–0 PM7-PM0	Prescaler Modulus Select. These bits control the prescale divider in the clock generator. This prescaler is used only in Internal Clock mode to divide the internal clock. The bit clock output is available at the clock port. A divide ratio from 1 to 256 (PM[7:0] = 0x00 to 0xFF) can be selected. Refer to DIV2, PSR and PM Bit Description for details regarding settings.

52.8.10 SSI FIFO Control/Status Register (SSIx_SSI_SFCSR)

The SSI FIFO Control / Status Register indicates the status of the Transmit FIFO Empty flag, with different settings of the Transmit FIFO WaterMark bits and varying amounts of data in the Tx FIFO

.

Table 52-55. Status of Transmit FIFO Empty Flag

Transmit FIFO Watermark (TFWM)	Number of data in Tx-Fifo														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
4	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
6	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
7	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
10	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
12	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Addresses: SSI-2_SSI_SFCSR is 5001_4000h base + 2Ch offset = 5001_402Ch

SSI-1_SSI_SFCSR is 63FC_C000h base + 2Ch offset = 63FC_C02Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]				RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

SSIx_SSI_SFCSR field descriptions

Field	Description
31–28 RFCNT1[3:0]	Receive FIFO Counter1. These bits indicate the number of data words in Receive FIFO 1. 0000 0 data word in receive FIFO 0001 1 data word in receive FIFO 0010 2 data word in receive FIFO 0011 3 data word in receive FIFO 0100 4 data word in receive FIFO 0101 5 data word in receive FIFO 0110 6 data word in receive FIFO 0111 7 data word in receive FIFO 1000 8 data word in receive FIFO 1001 9 data word in receive FIFO 1010 10 data word in receive FIFO

Table continues on the next page...

SSIx_SSI_SFCSR field descriptions (continued)

Field	Description
	1011 11 data word in receive FIFO 1100 12 data word in receive FIFO 1101 13 data word in receive FIFO 1110 14 data word in receive FIFO 1111 15 data word in receive FIFO
27–24 TFCNT1[3:0]	Transmit FIFO Counter1. These bits indicate the number of data words in Transmit FIFO. 0000 0 data word in transmit FIFO 0001 1 data word in transmit FIFO 0010 2 data word in transmit FIFO 0011 3 data word in transmit FIFO 0100 4 data word in transmit FIFO 0101 5 data word in transmit FIFO 0110 6 data word in transmit FIFO 0111 7 data word in transmit FIFO 1000 8 data word in transmit FIFO 1001 9 data word in transmit FIFO 1010 10 data word in transmit FIFO 1011 11 data word in transmit FIFO 1100 12 data word in transmit FIFO 1101 13 data word in transmit FIFO 1110 14 data word in transmit FIFO 1111 15 data word in transmit FIFO
23–20 RFWM1[3:0]	Receive FIFO Full WaterMark 1. These bits control the threshold at which the RFF1 flag will be set. The RFF1 flag is set whenever the data level in Rx FIFO 1 reaches the selected threshold. 0000 Reserved 0001 RFF set when at least one data word has been written to the Receive FIFO. Set when RxFIFO = 1,2.....15 data words 0010 RFF set when 2 or more data words have been written to the Receive FIFO. Set when RxFIFO = 2,3.....15 data words 0011 RFF set when 3 or more data words have been written to the Receive FIFO. Set when RxFIFO = 3,4.....15 data words 0100 RFF set when 4 or more data words have been written to the Receive FIFO. Set when RxFIFO = 4,5.....15 data words 0101 RFF set when 5 or more data words have been written to the Receive FIFO. Set when RxFIFO = 5,6.....15 data words 0110 RFF set when 6 or more data words have been written to the Receive.. Set when RxFIFO = 6,7.....15 data words 0111 RFF set when 7 or more data words have been written to the Receive FIFO. Set when RxFIFO = 7,8.....15 data words 1000 RFF set when 8 or more data words have been written to the Receive FIFO. Set when RxFIFO =8,9.....15 data words 1001 RFF set when 9 or more data words have been written to the Receive FIFO. Set when RxFIFO = 9,10.....15 data words 1010 RFF set when 10 or more data words have been written to the Receive FIFO. Set when RxFIFO = 10,11.....15 data words 1011 RFF set when 11 or more data words have been written to the Receive FIFO. Set when RxFIFO = 11,12.....15 data words

Table continues on the next page...

SSIx_SSI_SFCSR field descriptions (continued)

Field	Description
	1100 RFF set when 12 or more data words have been written to the Receive FIFO. Set when RxFIFO = 12,13.....15 data words 1101 RFF set when 13 or more data words have been written to the Receive FIFO. Set when RxFIFO = 13,14,15data words 1110 RFF set when 14 or more data words have been written to the Receive FIFO. Set when RxFIFO = 14,15 data words 1111 RFF set when 15 data words have been written to the Receive FIFO (default). Set when RxFIFO = 15 data words
19–16 TFWM1[3:0]	Transmit FIFO Empty WaterMark 1. These bits control the threshold at which the TFE1 flag will be set. The TFE1 flag is set whenever the empty slots in Tx FIFO exceed or are equal to the selected threshold. 0000 Reserved 0001 TFE set when there are more than or equal to 1 empty slots in Transmit FIFO (default). Transmit FIFO empty is set when TxFIFO <= 14 data. 0010 TFE set when there are more than or equal to 2 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=13 data. 0011 TFE set when there are more than or equal to 3 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=12 data. 0100 TFE set when there are more than or equal to 4 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=11 data. 0101 TFE set when there are more than or equal to 5 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=10 data. 0110 TFE set when there are more than or equal to 6 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=9 data. 0111 TFE set when there are more than or equal to 7 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=8 data. 1000 TFE set when there are more than or equal to 8 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <=7 data. 1001 TFE set when there are more than or equal to 9 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 6 data. 1010 TFE set when there are more than or equal to 10 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 5 data. 1011 TFE set when there are more than or equal to 11 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 4 data. 1100 TFE set when there are more than or equal to 12 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 3 data. 1101 TFE set when there are more than or equal to 13 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 2 data. 1110 TFE set when there are more than or equal to 14 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 1 data. 1111 TFE set when there are 15 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO = 0 data.
15–12 RFCNT0[3:0]	Receive FIFO Counter 0. These bits indicate the number of data words in Receive FIFO 0. Refer to RFCNT1[3:0] for details regarding settings for receive FIFO counter bits.
11–8 TFCNT0[3:0]	Transmit FIFO Counter 0. These bits indicate the number of data words in Transmit FIFO 0. Refer to TFCNT1[3:0] for details regarding settings for transmit FIFO counter bits.
7–4 RFWM0[3:0]	Receive FIFO Full WaterMark 0. These bits control the threshold at which the RFF0 flag will be set. The RFF0 flag is set whenever the data level in Rx FIFO 0 reaches the selected threshold. Refer to RFWM1[3:0] for details regarding settings for receive FIFO watermark bits.

Table continues on the next page...

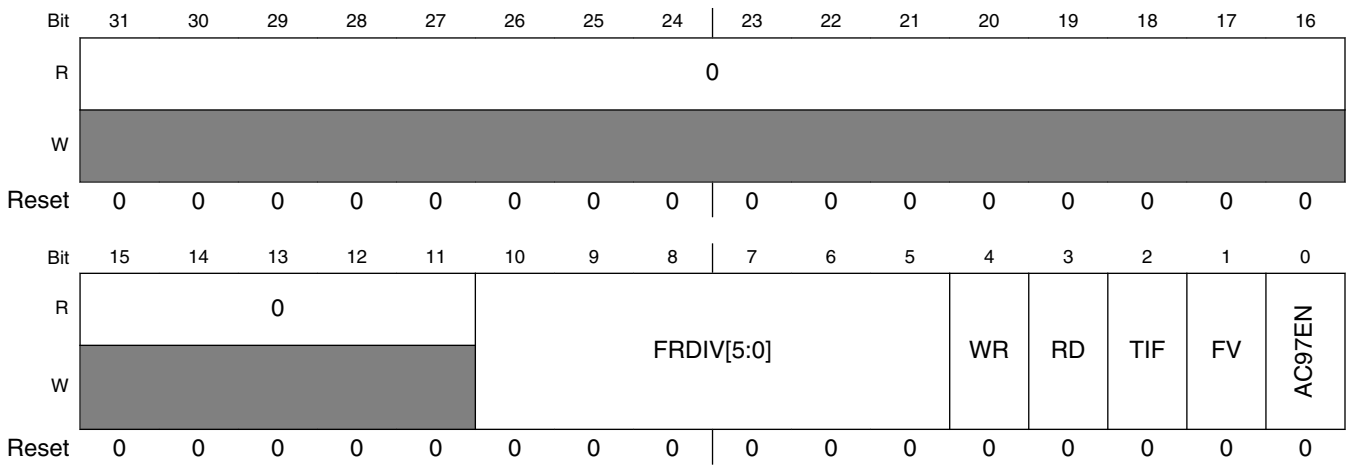
SSIx_SSI_SFCSR field descriptions (continued)

Field	Description
3–0 TFWM0[3:0]	Transmit FIFO Empty WaterMark 0. These bits control the threshold at which the TFE0 flag will be set. The TFE0 flag is set whenever the empty slots in Tx FIFO exceed or are equal to the selected threshold. Refer to TFWM1[3:0] for details regarding settings for transmit FIFO watermark bits.

52.8.11 SSI AC97 Control Register (SSIx_SSI_SACNT)

Addresses: SSI-2_SSI_SACNT is 5001_4000h base + 38h offset = 5001_4038h

SSI-1_SSI_SACNT is 63FC_C000h base + 38h offset = 63FC_C038h



SSIx_SSI_SACNT field descriptions

Field	Description
31–11 Reserved	This read-only field is reserved and always has the value zero. Reserved
10–5 FRDIV[5:0]	Frame Rate Divider. These bits control the frequency of AC97 data transmission/reception. They are programmed with the number of frames for which the SSI should be idle, after operating in one frame. Through these bits, AC97 frequency of operation, from 48 KHz (000000) to 1 KHz (101111) can be achieved. Sample Value: 001010 (10 Decimal) = SSI will operate once every 11 frames.
4 WR	Write Command. This bit specifies whether the next frame will carry an AC97 Write Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bits (corresponding to Command Address and Command Data slots of the next Tx frame) are automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a Write Command. 1 Next frame will have a Write Command.
3 RD	Read Command. This bit specifies whether the next frame will carry an AC97 Read Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bit (corresponding to Command Address slot of the next Tx frame) is automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame.

Table continues on the next page...

SSIx_SSI_SACNT field descriptions (continued)

Field	Description
	0 Next frame will not have a Read Command. 1 Next frame will have a Read Command.
2 TIF	Tag in FIFO. This bit controls the destination of the information received in AC97 tag slot (Slot #0). 0 Tag info stored in SATAG register. 1 Tag info stored in Rx FIFO 0.
1 FV	Fixed/Variable Operation. This bit selects whether the SSI is in AC97 Fixed mode or AC97 Variable mode. 0 AC97 Fixed Mode. 1 AC97 Variable Mode.
0 AC97EN	AC97 Mode Enable. This bit is used to enable SSI AC97 operation. Refer to AC97 Mode for details of AC97 operation. 0 AC97 mode disabled. 1 SSI in AC97 mode.

52.8.12 SSI AC97 Command Address Register (SSIx_SSI_SACADD)

Addresses: SSI-2_SSI_SACADD is 5001_4000h base + 3Ch offset = 5001_403Ch

SSI-1_SSI_SACADD is 63FC_C000h base + 3Ch offset = 63FC_C03Ch

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

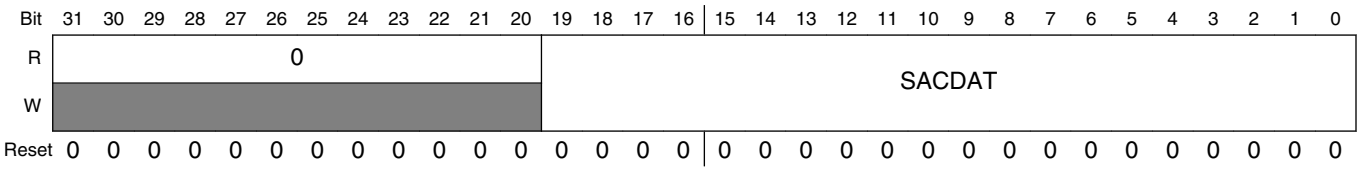
SSIx_SSI_SACADD field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value zero. Reserved
18–0 SACADD	AC97 Command Address. These bits store the Command Address Slot information (bit 19 of the slot is sent in accordance with the Read and Write Command bits in SSI_SACNT register). These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Address Slot. If the contents of these bits change due to an update, the CMDAU bit in SISR is set.

52.8.13 SSI AC97 Command Data Register (SSIx_SSI_SACDAT)

Addresses: SSI-2_SSI_SACDAT is 5001_4000h base + 40h offset = 5001_4040h

SSI-1_SSI_SACDAT is 63FC_C000h base + 40h offset = 63FC_C040h



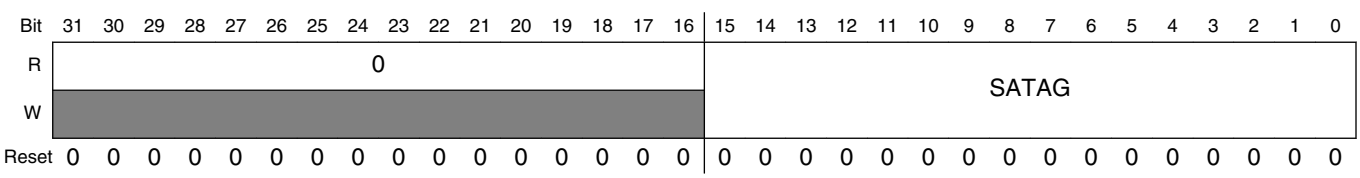
SSIx_SSI_SACDAT field descriptions

Field	Description
31–20 Reserved	This read-only field is reserved and always has the value zero. Reserved
19–0 SACDAT	AC97 Command Data. The outgoing Command Data Slot carries the information contained in these bits. These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Data Slot. If the contents of these bits change due to an update, the CMDDU bit in SISR is set. These bits are transmitted only during AC97 Write Command. During AC97 Read Command, 0x00000 is transmitted in time slot #2.

52.8.14 SSI AC97 Tag Register (SSIx_SATAG)

Addresses: SSI-2_SATAG is 5001_4000h base + 44h offset = 5001_4044h

SSI-1_SATAG is 63FC_C000h base + 44h offset = 63FC_C044h



SSIx_SATAG field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 SATAG	AC97 Tag Value. Writing to this register (by the Core) sets the value of the Tx-Tag in AC97 fixed mode of operation. On a read, the Core gets the Rx-Tag Value received (in the last frame) from the Codec. If TIF bit in SSI_SACNT register is set, the TAG value is also stored in Rx-FIFO in addition to SATAG register. When the received Tag value changes, the RXT bit in SISR register is set. Bits SATAG[1:0] convey the Codec -ID. In current implementation only Primary Codecs are supported. Thus writing value 2'b00 to this field is mandatory.

52.8.15 SSI Transmit Time Slot Mask Register (SSIx_SSI_STMSK)

Addresses: SSI-2_SSI_STMSK is 5001_4000h base + 48h offset = 5001_4048h

SSI-1_SSI_STMSK is 63FC_C000h base + 48h offset = 63FC_C048h

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

SSIx_SSI_STMSK field descriptions

Field	Description
31–0 STMSK	Transmit Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI transmits data. Each bit has info corresponding to the respective time slot in the frame. Transmit mask bits should not be used in I2S Slave mode of operation. SSI_STMSK register value must be set before enabling Transmission. 0 Valid Time Slot. 1 Time Slot masked (no data transmitted in this time slot).

52.8.16 SSI Receive Time Slot Mask Register (SSIx_SSI_SRMSK)

Addresses: SSI-2_SSI_SRMSK is 5001_4000h base + 4Ch offset = 5001_404Ch

SSI-1_SSI_SRMSK is 63FC_C000h base + 4Ch offset = 63FC_C04Ch

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

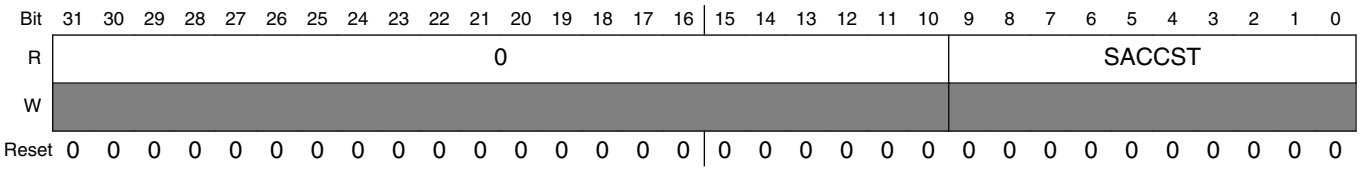
SSIx_SSI_SRMSK field descriptions

Field	Description
31–0 SRMSK	Receive Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI receives data. Each bit has info corresponding to the respective time slot in the frame. SSI_SRMSK register value must be set before enabling Receiver. Receive mask bits should not be used in I2S Slave mode of operation. 0 Valid Time Slot. 1 Time Slot masked (no data received in this time slot).

52.8.17 SSI AC97 Channel Status Register (SSIx_SSI_SACCST)

Addresses: SSI-2_SSI_SACCST is 5001_4000h base + 50h offset = 5001_4050h

SSI-1_SSI_SACCST is 63FC_C000h base + 50h offset = 63FC_C050h



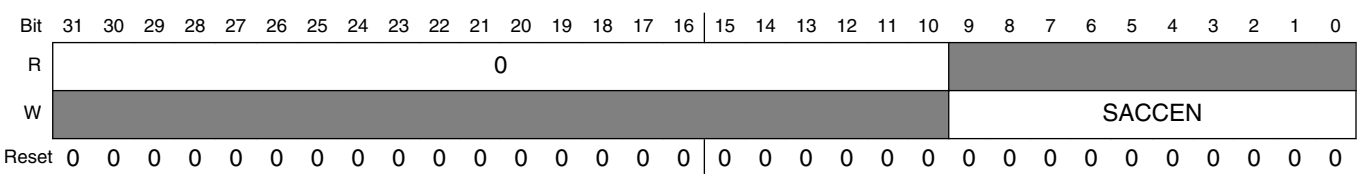
SSIx_SSI_SACCST field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9–0 SACCST	AC97 Channel Status. These bits indicate which data slot has been enabled in AC97 variable mode operation. This register is updated in case the core enables/disables a channel through a write to SSI_SACCEN/SSI_SACCDIS register or the external codec enables a channel by sending a '1' in the corresponding SLOTREQ bit. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). The contents of this register only have relevance while the SSI is operating in AC97 variable mode. Writes to this register result in an error response on the block interface. 0 Data channel disabled. 1 Data channel enabled.

52.8.18 SSI AC97 Channel Enable Register (SSIx_SSI_SACCEN)

Addresses: SSI-2_SSI_SACCEN is 5001_4000h base + 54h offset = 5001_4054h

SSI-1_SSI_SACCEN is 63FC_C000h base + 54h offset = 63FC_C054h



SSIx_SSI_SACCEN field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9–0 SACCEN	AC97 Channel Enable. The Core writes a '1' to these bits to enable an AC97 data channel. Writing a '0' has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as '0' by the Core.

Table continues on the next page...

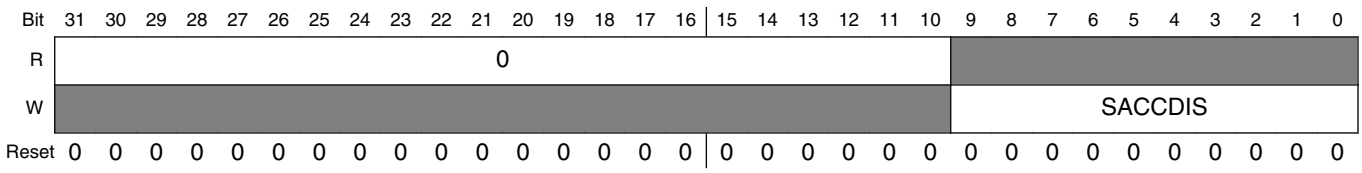
SSIx_SSI_SACCEN field descriptions (continued)

Field	Description
	0 Write Has no effect. 1 Write Enables the corresponding data channel.

52.8.19 SSI AC97 Channel Disable Register (SSIx_SSI_SACCDIS)

Addresses: SSI-2_SSI_SACCDIS is 5001_4000h base + 58h offset = 5001_4058h

SSI-1_SSI_SACCDIS is 63FC_C000h base + 58h offset = 63FC_C058h



SSIx_SSI_SACCDIS field descriptions

Field	Description
31–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9–0 SACCDIS	AC97 Channel Disable. The Core writes a '1' to these bits to disable an AC97 data channel. Writing a '0' has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as '0' by the Core. 0 Write Has no effect. 1 Write Disables the corresponding data channel.

Chapter 53
Temperature Sensor (TempSensor)

53.1 Introduction

The temperature sensor on i.MX50 implements a simple temperature to time conversion which based on the known clock frequency of the digital-counters used in the algorithm, the programming temperature of the TempSensor fuses and the known design parameters can then be converted by software to determine the actual junction temperature of the device.

53.2 Overview

The temperature sensor architecture uses the dependence of a bipolar VBE voltage to generate a temperature dependent reference voltage. When a temperature reading is requested, the digital counters driven by a known clock source are started at the same time as a temperature independent ramp voltage. The digital counters are stopped when the ramp voltage crosses the temperature dependent reference voltages. The resulting counter values can then be converted to a junction temperature value based on a mathematical equation presented further in this chapter. Figure 53-1 below shows a graphical representation of the TempSensor operation.

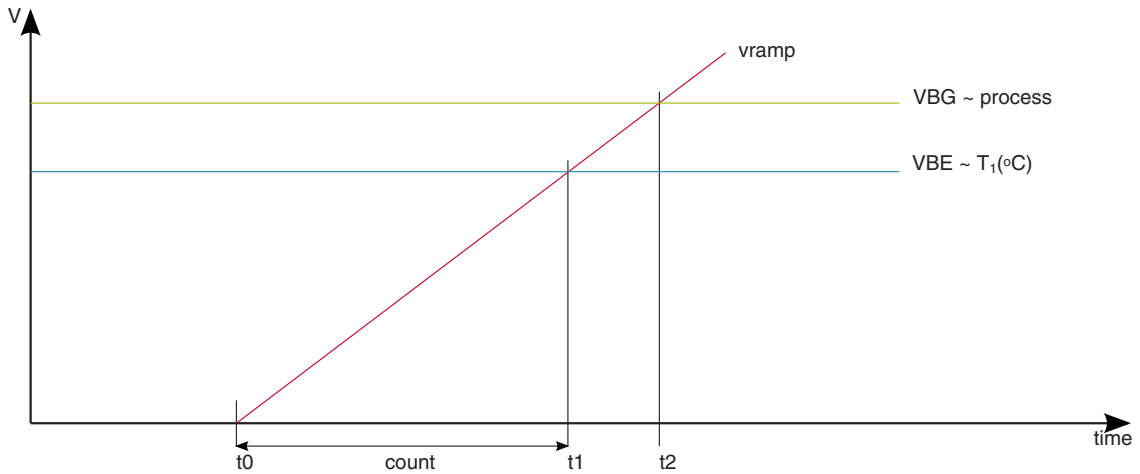


Figure 53-1. Temp-Sensor Operation

53.3 Features

The main features of the TempSensor architecture in i.MX50 are:

- Low-power operation, self-power up and power-down feature when a conversion cycle is requested
- Simple design

53.4 Digital-Count-to-Temperature Conversion Algorithm

The required information needed to extract the local junction temperature from the digital-count when a temperature-sense cycle is executed is the following:

- K_T: Designed slope: number-of-clocks/°C
- T_p: Temperature of fuse-programming
- N_p: Digital counter value at programmed temperature
- N_T: Digital counter value of temp-sense cycle(TVAL1)
- N_R: Typical counter value from simulation at programmed temperature

With the above information known, the temperature read-out can be obtained using the following equation: $T_J = T_P + (N_P - N_T) \times (N_R/N_T)/K_T$

53.5 TempSensor Memory Map/Register Definition

TEMPSENSOR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4101_8080	TempSensor Control Register (TEMPSENSOR_ANADIG_CTRL)	32	R/W	0000_0000h	53.5.1/ 2985
4101_8084	TempSensor Control Register (TEMPSENSOR_ANADIG_CTRL_SET)	32	R/W	0000_0000h	53.5.1/ 2985
4101_8088	TempSensor Control Register (TEMPSENSOR_ANADIG_CTRL_CLR)	32	R/W	0000_0000h	53.5.1/ 2985
4101_808C	TempSensor Control Register (TEMPSENSOR_ANADIG_CTRL_TOG)	32	R/W	0000_0000h	53.5.1/ 2985

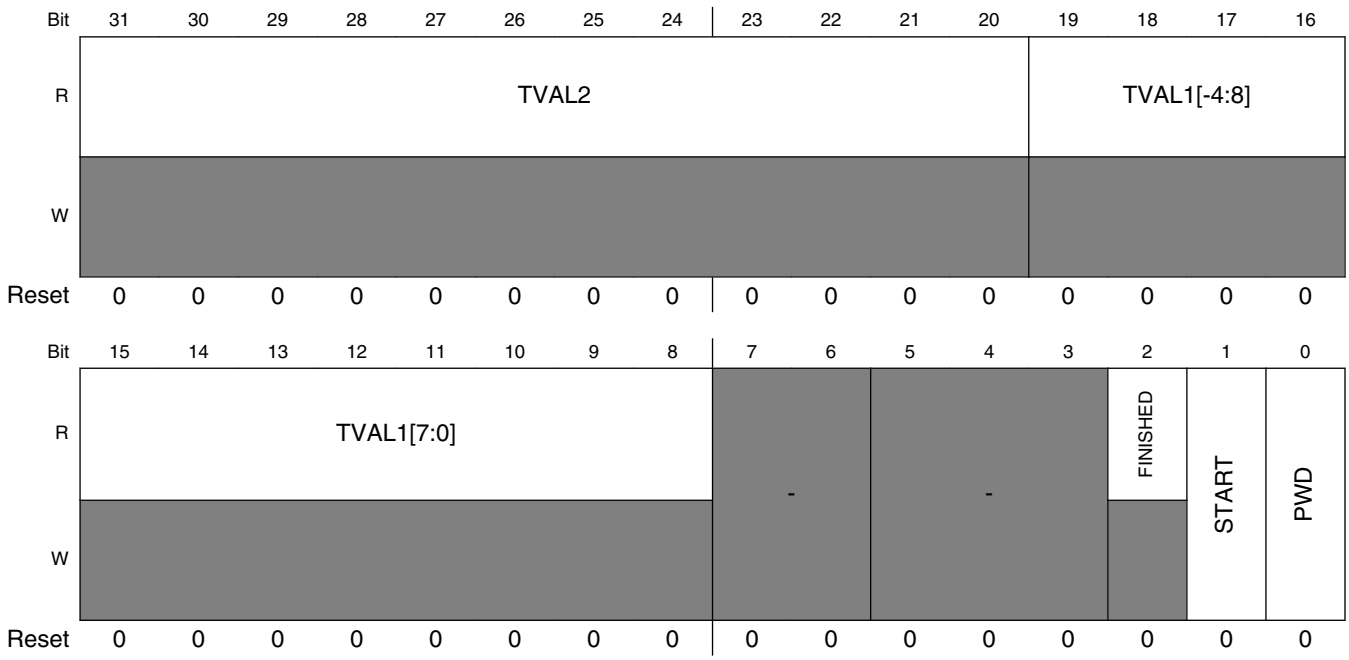
53.5.1 TempSensor Control Register (TEMPSENSOR_ANADIG_CTRL*n*)

Addresses: TEMPSENSOR_ANADIG_CTRL is 4101_8000h base + 80h offset = 4101_8080h

TEMPSENSOR_ANADIG_CTRL_SET is 4101_8000h base + 84h offset = 4101_8084h

TEMPSENSOR_ANADIG_CTRL_CLR is 4101_8000h base + 88h offset = 4101_8088h

TEMPSENSOR_ANADIG_CTRL_TOG is 4101_8000h base + 8Ch offset = 4101_808Ch



TEMPSENSOR_ANADIG_CTRL*n* field descriptions

Field	Description
31–20 TVAL2	Output code of the TempSensor VBG based counter. This code can be used to account for process variations that can affect the accuracy of the TempSensor as described in the next section.

Table continues on the next page...

TEMPSENSOR_ANADIG_CTRL*n* field descriptions (continued)

Field	Description
19–8 TVAL1	Output code of the TempSensor VBE based counter. This code is used in determining the junction temperature as described in the next section.
7–6 -	Reserved
5–3 -	Reserved
2 FINISHED	Output bit that signals that the TempSensor has finished and that the values in the TVAL1 and TVAL2 fields are valid.
1 START	Control bits to start a TempSensor conversion cycle.
0 PWD	Control bit which sets the self-power-down behavior of the analog portion of the TempSensor. 0 **The analog portion of the TempSensor will always be powered-up.** — 1 **The TempSensor digital controller will power-up the analog portion only after the START signal is set and power-down after the temperature conversion cycle has completed.** —

Chapter 54
Interrupt Controller (TZIC)

54.1 Overview

The interrupt controller allows complete and independent control over every interrupt connected to the controller, including enable and priority functionality. It supports priority masking and access to raw interrupt state and pending transactions.

NOTE

Although we use the mnemonic "TZIC," the Interrupt Controller in this chip does not support TrustZone.

NOTE

Chip-specific system interrupts are defined and described in the Interrupts and SDMA Events chapter.

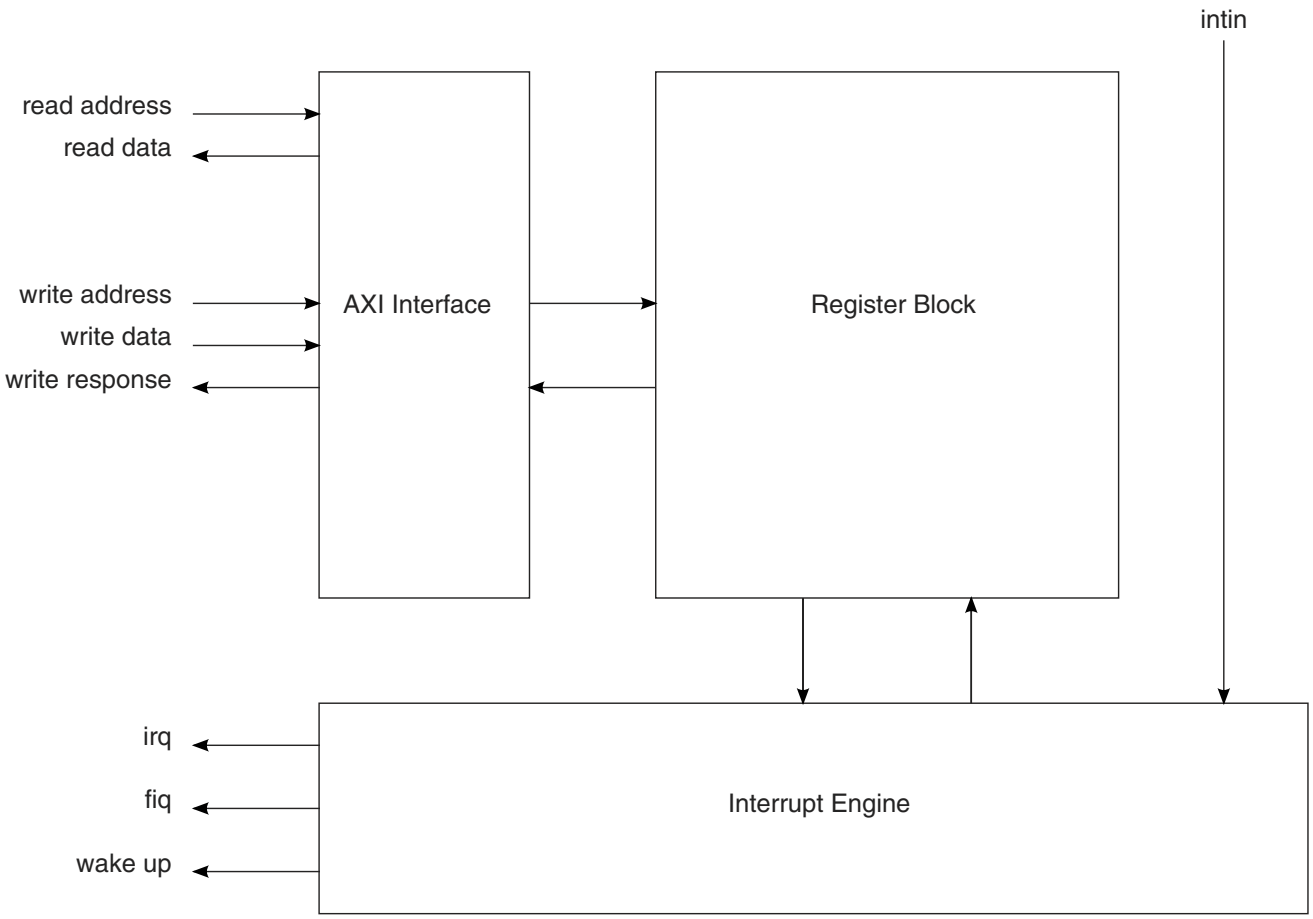


Figure 54-1. Block Diagram

54.2 Features

The following features are included in the interrupt controller

- AXI interface
- Support for up to 128 interrupts
- Secure configuration available on all interrupts
- Software triggered interrupts supported
- High priority pending registers for autovectored interrupt service routines
- Set/Clear registers for interrupt enable, force, and secure registers
- 16/32 programmable priority levels
- Wakeup configuration available on all interrupts

54.3 External Signal Description

The interrupt controller has no external signals.

54.4 Functional Description

The TZIC is an interrupt controller . This implementation allows up to 128 interrupts. It supports autovectoring solutions and provides a high priority pending register for software to directly access the interrupts currently pending at the highest active priority level .

The interrupt controller is made up of three primary components:

1. The AXI Interface
2. The Register Block
3. The Interrupt Engine

54.4.1 Security Configurability

In this chip, the interrupt controller is configured to support a single domain.

54.4.1.1 Interrupt Priority

The priority scheme implemented in the interrupt controller allows for flexible prioritization of interrupts, and ease of use with legacy software.

Each interrupt can be assigned a priority value up to 8 bits (depending on the PRIORITY_MASK parameter). The highest priority value is 0, the next is 1, and so on. For designs where not all 8 bits are used, the lower bits are masked off, such that an implementation with only 5 bits would use values 0 (8'h00), 8 (8'h08), 16 (8'h10), 24 (8'h18), and so on. The number of bits available can be discovered by software through writing 8'hFF to a priority register and reading back the resulting value actually stored in the register.

54.4.2 AXI Interface

The AXI interface allows access to all functionality of the interrupt controller. It is fully compatible with the AMBA 3 AXI specification from ARM.

Read transactions are generally returned with one initial wait state and two clock cycles per read data. All registers can be read with 32-bit transactions, the priority registers allow 8-bit transactions. 16-bit transactions are not supported and will result in an error. Exclusive reads are not supported. The interrupt controller will accept one read address at a time. If a second read transaction is valid before completion of the previous read, it will be held off until the interrupt controller is finished returning data from the first read.

Write transactions are generally accepted with one initial wait state, the one clock cycle per write data. Registers are word writeable only with the exception of the priority and priority mask registers which support byte writes. Exclusive writes are not supported. The interrupt controller will accept one write address at a time. If a second write transaction is valid before completion of the previous write, it will be held off until the interrupt controller is finished returning the write response from the first write. The interrupt controller will also hold off the write data bus until it receives a write address.

54.4.3 Interrupt Engine

The interrupt engine performs all processing on the raw interrupt information. It receives configuration information from the Register Block, interrupts from the system, and based on that information determines when to assert interrupts to the core.

An interrupt is asserted to the core (IRQ or FIQ) when all the following criteria are met:

- The EN bit in the TZIC_INTCTRL register is enabled, FIQ or normal IRQ.
- An interrupt is asserted, either on an interrupt input (intin) or by writing the TZIC_SRCSET or TZIC_SWINT registers.
- The interrupt is configured appropriately for the desired interrupt in the TZIC_INTSEC register, set as FIQ or IRQ.
- The asserted interrupt is enabled in the TZIC_ENSET/TZIC_ENCLEAR register.
- The priority of the asserted interrupt is greater than the TZIC_PRIOMASK register value. Note: this requires the actual priority value to be greater than the TZIC_PRIOMASK register value.

This interrupt controller can handle up to 128 interrupts (intin). Each interrupt has an associated interrupt ID number 0 - 127.

The interrupt synchronizers are part of the interrupt engine. A two-flop synchronizer is implemented for each interrupt input. Clock gating is available for the synchronizers and is configured through the Synchronizer Control (TZIC_SYNCCTRL) Register. For more information see Synchronizer Control (TZIC_SYNCCTRL).

The interrupt engine handles the wakeup functionality of the interrupt controller, processing the dsm_int_holdoff signal input from the ARM platform and asserting tzic_wakeup_request to the clock controller module as necessary.

tzic_wakeup_request is an asynchronous, combinational output determined by the asynchronous interrupt sources (intin) and the TZIC_WAKEUP registers. It asserts when any interrupt is asserted on the intin input while configured for wakeup. It negates when there are no more interrupts asserted that are configured for wakeup. A software interrupt will not assert this output, because a software interrupt would require a clock to be triggered and the tzic_wakeup_request is intended for enabling clocks and/or power during low power states.

dsm_int_holdoff causes the interrupt synchronizers to stop updating. This is for shutting down clocks in the system. While preparing for low power modes, the ARM platform may be in a position where it expects no more interrupts to be occurring, but has not yet executed a standby-wait-for-interrupt. The ARM platform must write to the dsm_int_holdoff bit, then read it back to verify that it was successfully set. If not set, an interrupt occurred during the process before the synchronizers could be stopped and the ARM platform must process the interrupt and write to the dsm_int_holdoff bit again.

NOTE

Above internal decription is for information only, because the mechanisms are automated through the clock controller module when going into a low power mode.

54.4.4 Auto-Vectored Interrupt Handling

The TZIC is intended to be used with auto-vectored interrupt service routines. Once interrupts have been enabled using the TZIC_ENSET/TZIC_ENCLEAR registers and the TZIC_INTCTRL register, the interrupt engine is live and will signal an interrupt to the ARM platform when an enabled interrupt is asserted. Once the ARM platform interrupt is triggered, a read to the TZIC_PND or TZIC_HIPND registers will determine which interrupts are currently pending.

The TZIC_PND registers will contain information on every interrupt currently pending, whether asserted by the interrupt input or forced by software. This includes interrupts of priority lower than the TZIC_PRIOMASK value. The TZIC_PRIOMASK register limits the assertion of the IRQ/FIQ signals based on priority, it does not affect which interrupts are visible in the pending registers.

The TZIC_HIPND registers will contain information on which interrupts are currently pending at the highest pending priority, allowing autovectoring software to choose between the highest priority interrupts without having to process such information on its own. As with the TZIC_PNDPND register, the TZI.HIPND register will return pending values even if the highest pending priority is of equal or lower priority than the TZIC_PRIOMASK value, although in such a case the interrupt signals will not be asserted.

After choosing a pending interrupt to service, writing the priority of the interrupt being serviced to the TZIC_PRIOMASK register will allow for interrupt pre-emption by a higher priority interrupt being asserted later.

54.4.5 Reset

Coming out of reset, all interrupts are configured as disabled, and set to priority level 0. Also, the TZIC_INTCTRL register is set to disable all interrupts.

54.5 Initialization Information

Note: When writing to any registers affecting the interrupt outputs to the ARM platform it is recommended the ARM platform interrupt inputs be masked to prevent interrupts asserting and negating due to register writes.

When the interrupt controller comes out of reset, all interrupts are disabled and set as FIQ. It is advised that the TZIC_INTCTRL register enables are not activated until all other configuration options have been set up. To set up the interrupt controller for autovectoring functionality:

1. Software sets TZIC_INTSEC registers appropriately - This configures all interrupts as either FIQ or normal IRQ interrupts.
2. Configure TZIC_ENSET/TZIC_ENCLEAR - Software must enable all interrupts. At this point interrupt functionality has yet to be enabled in the TZIC_INTCTRL register so this will not cause interrupts to occur during initialization.

3. Configure TZIC_PRIORITY - Software must set TZIC_PRIORITY registers to appropriate levels, otherwise all interrupts are set to the lowest priority level as determined by the PRIORITY_MASK parameter.
4. Set TZIC_PRIOMASK - The TZIC_PRIOMASK register must be set to allow interrupts to occur, out of reset it is set to the lowest priority level which masks all priority levels.
5. Set TZIC_INTCTRL - Software must enable interrupt functionality in the TZIC_INTCTRL register. This will enable the interrupt controller to send interrupts to the ARM platform.

At this point the interrupt controller will generate interrupt signals to the ARM platform. The ARM platform itself must enable interrupts as well. In this configuration the interrupt controller is enabled for autovectoring software to access the TZIC_PND or TZIC_HIPND registers and determine which interrupt to service.

54.6 Programmable Registers

This is the memory map for all possible registers in the interrupt controller. The presence of some registers will be dependent upon the implementation configuration. Address space not specified is reserved.

This section contains the detailed register descriptions for the interrupt controler registers.

Write bursts are taken with one clock per write data after an initial one clock delay. Read bursts are returned with two clocks per read data.

The specific behavior for each register set is described in the field descriptions. In some cases where the register described spans multiple 32-bit addressible locations, the variable X is used to denote which particular register corresponds to which interrupts. For instance, a register with 1 bit associated with each interrupt will have one 32-bit register for each set of 32 interrupts. The registers can be numbered such that REGISTER0 corresponds to interrupts 0-31, REGISTER1 corresponds to interrupts 32-63, and so on. This relationship can be expressed using the following equation:

REGISTER(X)[Y] corresponds to interrupt (32 x (X))+Y

Such that if X is 2 and Y is 17:

REGISTER(2)[17] -> (32 x (2))+17 = 81

The interrupt would be number 81.

This terminology is used to express the location and relationship for several registers in this section.

All registers are accessible with 32-bit accesses. The PRIOMASK register and PRIORITY registers support 8-bit transactions, all others are not supported and will yield unpredictable results. 16-bit transactions are not supported to any registers and will result in a slave error. Any access larger than 32-bit will also result in an error.

All registers are bufferable, non-cacheable, data, supervisor only. Transactions marked as cacheable, allocate, user, or instruction will result in error responses.

TZIC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0FFF_C000	Control Register (TZIC_INTCTRL)	32	R/W	0000_0000h	54.6.1/ 2997
0FFF_C004	Interrupt Controller Type Register (TZIC_INTTYPE)	32	R	0000_0403h	54.6.2/ 2998
0FFF_C00C	Priority Mask Register (TZIC_PRIOMASK)	32	R/W	0000_0000h	54.6.3/ 2999
0FFF_C010	Synchronizer Control (TZIC_SYNCCTRL)	32	R/W	0000_0000h	54.6.4/ 3000
0FFF_C014	DSM Interrupt Holdoff (TZIC_DSMINT)	32	R/W	0000_0000h	54.6.5/ 3001
0FFF_C080	Interrupt Security (FIQ) Register: Irq 0 to 31 (TZIC_INTSEC0)	32	R/W	0000_0000h	54.6.6/ 3002
0FFF_C084	Interrupt Security (FIQ) Register: Irq 0 to 31 (TZIC_INTSEC1)	32	R/W	0000_0000h	54.6.6/ 3002
0FFF_C088	Interrupt Security (FIQ) Register: Irq 0 to 31 (TZIC_INTSEC2)	32	R/W	0000_0000h	54.6.6/ 3002
0FFF_C08C	Interrupt Security (FIQ) Register: Irq 0 to 31 (TZIC_INTSEC3)	32	R/W	0000_0000h	54.6.6/ 3002
0FFF_C100	Enable Set Register: Irq 0 to 31 (TZIC_ENSET0)	32	R/W	0000_0000h	54.6.7/ 3003
0FFF_C104	Enable Set Register: Irq 0 to 31 (TZIC_ENSET1)	32	R/W	0000_0000h	54.6.7/ 3003
0FFF_C108	Enable Set Register: Irq 0 to 31 (TZIC_ENSET2)	32	R/W	0000_0000h	54.6.7/ 3003
0FFF_C10C	Enable Set Register: Irq 0 to 31 (TZIC_ENSET3)	32	R/W	0000_0000h	54.6.7/ 3003
0FFF_C180	Enable Clear Register: Irq 0 to 31 (TZIC_ENCLEAR0)	32	R/W	0000_0000h	54.6.8/ 3004
0FFF_C184	Enable Clear Register: Irq 0 to 31 (TZIC_ENCLEAR1)	32	R/W	0000_0000h	54.6.8/ 3004

Table continues on the next page...

TZIC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0FFF_C188	Enable Clear Register: Irq 0 to 31 (TZIC_ENCLEAR2)	32	R/W	0000_0000h	54.6.8/ 3004
0FFF_C18C	Enable Clear Register: Irq 0 to 31 (TZIC_ENCLEAR3)	32	R/W	0000_0000h	54.6.8/ 3004
0FFF_C200	Source Set Register: Irq 0 to 31 (TZIC_SRCSET0)	32	R/W	0000_0000h	54.6.9/ 3005
0FFF_C204	Source Set Register: Irq 0 to 31 (TZIC_SRCSET1)	32	R/W	0000_0000h	54.6.9/ 3005
0FFF_C208	Source Set Register: Irq 0 to 31 (TZIC_SRCSET2)	32	R/W	0000_0000h	54.6.9/ 3005
0FFF_C20C	Source Set Register: Irq 0 to 31 (TZIC_SRCSET3)	32	R/W	0000_0000h	54.6.9/ 3005
0FFF_C280	Source Clear Register: Irq 0 to 31 (TZIC_SRCCLEAR0)	32	R/W	0000_0000h	54.6.10/ 3006
0FFF_C284	Source Clear Register: Irq 0 to 31 (TZIC_SRCCLEAR1)	32	R/W	0000_0000h	54.6.10/ 3006
0FFF_C288	Source Clear Register: Irq 0 to 31 (TZIC_SRCCLEAR2)	32	R/W	0000_0000h	54.6.10/ 3006
0FFF_C28C	Source Clear Register: Irq 0 to 31 (TZIC_SRCCLEAR3)	32	R/W	0000_0000h	54.6.10/ 3006
0FFF_C400	Priority Register: Irq 0 to 3 (TZIC_PRIORITY0)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C404	Priority Register: Irq 0 to 3 (TZIC_PRIORITY1)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C408	Priority Register: Irq 0 to 3 (TZIC_PRIORITY2)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C40C	Priority Register: Irq 0 to 3 (TZIC_PRIORITY3)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C410	Priority Register: Irq 0 to 3 (TZIC_PRIORITY4)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C414	Priority Register: Irq 0 to 3 (TZIC_PRIORITY5)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C418	Priority Register: Irq 0 to 3 (TZIC_PRIORITY6)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C41C	Priority Register: Irq 0 to 3 (TZIC_PRIORITY7)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C420	Priority Register: Irq 0 to 3 (TZIC_PRIORITY8)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C424	Priority Register: Irq 0 to 3 (TZIC_PRIORITY9)	32	R/W	0000_0000h	54.6.11/ 3006

Table continues on the next page...

TZIC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0FFF_C428	Priority Register: Irq 0 to 3 (TZIC_PRIORITY10)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C42C	Priority Register: Irq 0 to 3 (TZIC_PRIORITY11)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C430	Priority Register: Irq 0 to 3 (TZIC_PRIORITY12)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C434	Priority Register: Irq 0 to 3 (TZIC_PRIORITY13)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C438	Priority Register: Irq 0 to 3 (TZIC_PRIORITY14)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C43C	Priority Register: Irq 0 to 3 (TZIC_PRIORITY15)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C440	Priority Register: Irq 0 to 3 (TZIC_PRIORITY16)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C444	Priority Register: Irq 0 to 3 (TZIC_PRIORITY17)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C448	Priority Register: Irq 0 to 3 (TZIC_PRIORITY18)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C44C	Priority Register: Irq 0 to 3 (TZIC_PRIORITY19)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C450	Priority Register: Irq 0 to 3 (TZIC_PRIORITY20)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C454	Priority Register: Irq 0 to 3 (TZIC_PRIORITY21)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C458	Priority Register: Irq 0 to 3 (TZIC_PRIORITY22)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C45C	Priority Register: Irq 0 to 3 (TZIC_PRIORITY23)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C460	Priority Register: Irq 0 to 3 (TZIC_PRIORITY24)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C464	Priority Register: Irq 0 to 3 (TZIC_PRIORITY25)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C468	Priority Register: Irq 0 to 3 (TZIC_PRIORITY26)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C46C	Priority Register: Irq 0 to 3 (TZIC_PRIORITY27)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C470	Priority Register: Irq 0 to 3 (TZIC_PRIORITY28)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C474	Priority Register: Irq 0 to 3 (TZIC_PRIORITY29)	32	R/W	0000_0000h	54.6.11/ 3006

Table continues on the next page...

TZIC memory map (continued)

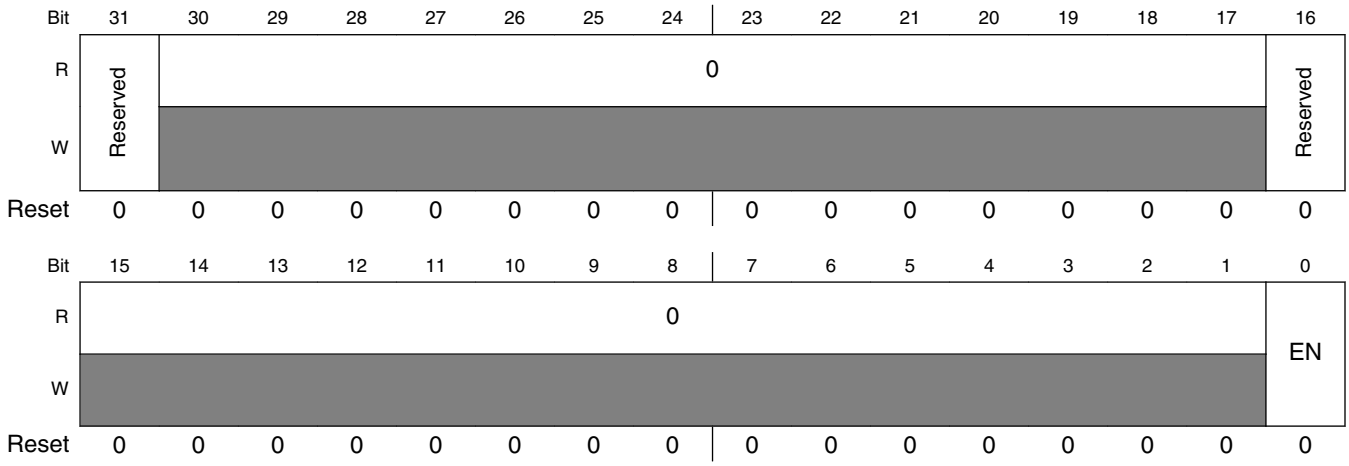
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0FFF_C478	Priority Register: Irq 0 to 3 (TZIC_PRIORITY30)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_C47C	Priority Register: Irq 0 to 3 (TZIC_PRIORITY31)	32	R/W	0000_0000h	54.6.11/ 3006
0FFF_CD00	Pending Register: Irq 0 to 31 (TZIC_PND0)	32	R	0000_0000h	54.6.12/ 3008
0FFF_CD04	Pending Register: Irq 0 to 31 (TZIC_PND1)	32	R	0000_0000h	54.6.12/ 3008
0FFF_CD08	Pending Register: Irq 0 to 31 (TZIC_PND2)	32	R	0000_0000h	54.6.12/ 3008
0FFF_CD0C	Pending Register: Irq 0 to 31 (TZIC_PND3)	32	R	0000_0000h	54.6.12/ 3008
0FFF_CD80	High Priority Pending Register: Irq 0 to 31 (TZIC_HIPND0)	32	R	0000_0000h	54.6.13/ 3008
0FFF_CD84	High Priority Pending Register: Irq 0 to 31 (TZIC_HIPND1)	32	R	0000_0000h	54.6.13/ 3008
0FFF_CD88	High Priority Pending Register: Irq 0 to 31 (TZIC_HIPND2)	32	R	0000_0000h	54.6.13/ 3008
0FFF_CD8C	High Priority Pending Register: Irq 0 to 31 (TZIC_HIPND3)	32	R	0000_0000h	54.6.13/ 3008
0FFF_CE00	Wakeup Config Register: Irq 0 to 31 (TZIC_WAKEUP0)	32	R	0000_0000h	54.6.14/ 3009
0FFF_CE04	Wakeup Config Register: Irq 0 to 31 (TZIC_WAKEUP1)	32	R	0000_0000h	54.6.14/ 3009
0FFF_CE08	Wakeup Config Register: Irq 0 to 31 (TZIC_WAKEUP2)	32	R	0000_0000h	54.6.14/ 3009
0FFF_CE0C	Wakeup Config Register: Irq 0 to 31 (TZIC_WAKEUP3)	32	R	0000_0000h	54.6.14/ 3009
0FFF_CF00	Software Interrupt Trigger Register (TZIC_SWINT)	32	W	0000_0000h	54.6.15/ 3010

54.6.1 Control Register (TZIC_INTCTRL)

The interrupt control register (INTCTRL) can enable or disable all interrupts to the ARM platform.

This register can only be accessed by a 32-bit supervisor transaction.

Address: TZIC_INTCTRL is FFFC000h base + 0h offset = 0FFF_C000h



TZIC_INTCTRL field descriptions

Field	Description
31 Reserved	This field is reserved. 0
30–17 Reserved	This read-only field is reserved and always has the value zero. Reserved
16 Reserved	This field is reserved. 0
15–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 EN	Interrupt Enable. This bit, when set, enables interrupts. 0 Interrupts disabled 1 Interrupts enabled 0 1

54.6.2 Interrupt Controller Type Register (TZIC_INTTYPE)

The interrupt controller type register (INTTYPE) contains information about the type of interrupt controller that has been implemented. It can be used to determine the number of interrupts, ARM cores, and if security is enabled for a given instantiation.

The current version only supports configurations with one ARM core and one security domain(s).

This register can only be accessed by a 32-bit supervisor read transaction.

Address: TZIC_INTTYPE is FFFC000h base + 4h offset = 0FFF_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					DOM	0		CPUS			ITLINES				
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1

TZIC_INTTYPE field descriptions

Field	Description
31–11 Reserved	This read-only field is reserved and always has the value zero. Reserved
10 DOM	Security Domains. This bit indicates the number of domains supported by the current configuration. 0 Supports 1 Domain 1 Supports 2 Domains: Secure and Non-Secure
9–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–5 CPUS	ARM core Count. This field indicates the number of ARM cores supported by the current configuration. Currently only 1 ARM core is supported, this field is specified for future compatability. 000 Supports 1 ARM core 001 Supports 2 ARM cores 010 Supports 3 ARM cores 111 Supports 8 ARM cores
4–0 ITLINES	Interrupt Lines. This field indicates the total number of interrupt lines supported by the current configuration. 00000 up to 32 interrupt lines supported 00001 up to 64 interrupt lines supported 00010 up to 96 interrupt lines supported 11110 up to 992 interrupt lines supported 11111 up to 1020 interrupt lines supported

54.6.3 Priority Mask Register (TZIC_PRIOMASK)

The Priority Mask Register (PRIOMASK) is a single register used to mask interrupts based on their priority configuration. The ARM platform will only receive an interrupt if there is a pending interrupt with a priority higher than the value in this register. If an interrupt source asserts and its priority is equal to or lower than the priority mask value, it will not cause an interrupt to the ARM platform.

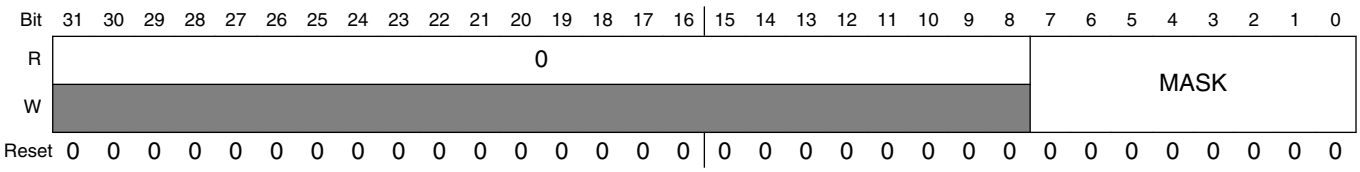
The lowest priority the priority mask value can be set to is the lowest priority level in the controller. Therefore any interrupt set to the lowest allowed priority will never cause an interrupt since it is not possible for its priority to be greater than the mask value.

This register is affected by an integration parameter which varies the available resolution of priority values. The number of bits available can vary from four to eight, with the lowest resolution only implementing bits [7:4], and the highest resolution implementing bits [7:0]. Which bits are implemented can be determined by writing 8'hFF to bits [7:0] and reading back the result. The bits set high will indicate which register bits are implemented in this register as well as the PRIORITY registers.

For more information regarding priorities, see Interrupt Priority.

This register can only be accessed by 8-bit or 32-bit supervisor transactions.

Address: TZIC_PRIOMASK is FFFC000h base + Ch offset = 0FFF_C00Ch



TZIC_PRIOMASK field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–0 MASK	Priority Mask - This field stores the priority mask value. Interrupts with a priority level set equal to or below this value are masked. For READS: 00000000 The priority mask is set to 0 00000001 The priority mask is set to 1 11111111 The priority mask is set to 255.

54.6.4 Synchronizer Control (TZIC_SYNCCTRL)

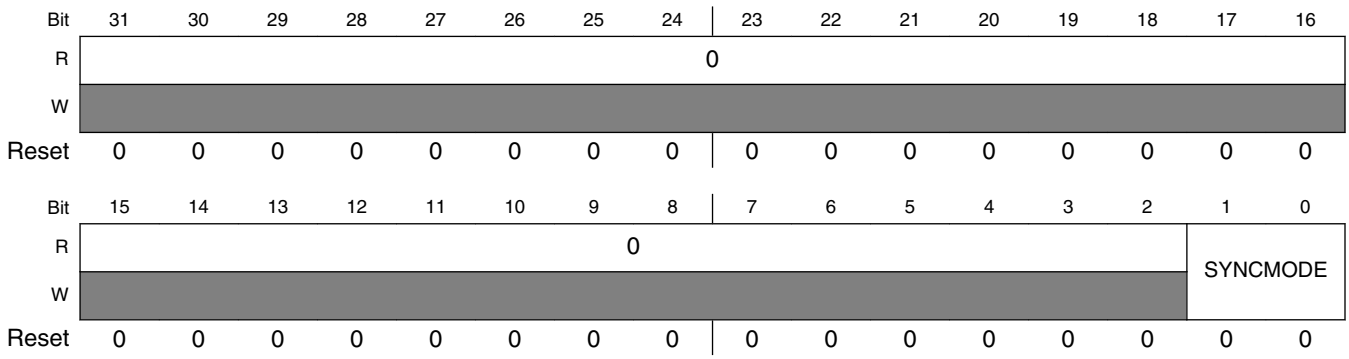
The Synchronizer Control Register (SYNCCTRL) is used to control latency and power usage in the interrupt synchronizers. There are three configurations for synchronization:

- Low Latency-The synchronizer is clocked continuously and interrupts are synchronized as they arrive.
- Low Power-The synchronizer is not clocked until a difference is detected between the inputs and outputs of the synchronizer. When a difference is detected, the detection result is synchronized and used as a clock enable for the synchronizer,

which introduces additional latency. Total latency is four clocks, two for the enable and two for the interrupt.

This register can only be accessed by 32-bit supervisor transactions.

Address: TZIC_SYNCCTRL is FFFC000h base + 10h offset = 0FFF_C010h



TZIC_SYNCCTRL field descriptions

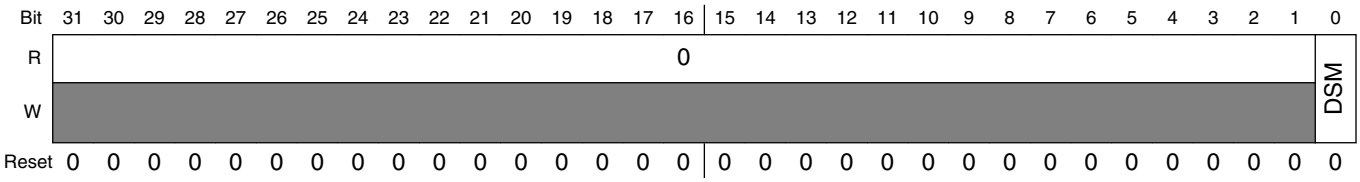
Field	Description
31–2 Reserved	This read-only field is reserved and always has the value zero. Reserved
1–0 SYNCMODE	Synchronizer Mode. 00 Low Latency 01 Low Power 10 Reserved 11 Reserved

54.6.5 DSM Interrupt Holdoff (TZIC_DSMINT)

The DSM Interrupt Holdoff Register (DSMINT) is used to stop interrupts to the ARM platform from occuring until a wakeup signal occurs. This is used for entering Deep Sleep Mode. When written, the DSM bit will cause the interrupt synchronizer to stop updating. If an interrupt occurs while this register is being written the bit will not be set and the ARM platform will have to try to write again. If no interrupt to the ARM platform occurs while the register is being written the write will succeed and the interrupt synchronizers will not update until the DSM bit clears. The DSM bit is cleared by either a write to the DSMINT register or assertion of the dsm_wakeup_request output of the interrupt controller. When a wakeup request is asserted, the interrupt controller will remove the DSM interrupt holdoff bit and interrupts will resume.

This register can only be accessed by 32-bit supervisor transactions.

Address: TZIC_DSMINT is FFFC000h base + 14h offset = 0FFF_C014h



TZIC_DSMINT field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 DSM	DSM Interrupt Holdoff 0 Interrupt synchronizer updates 1 Interrupt synchronizer does not update

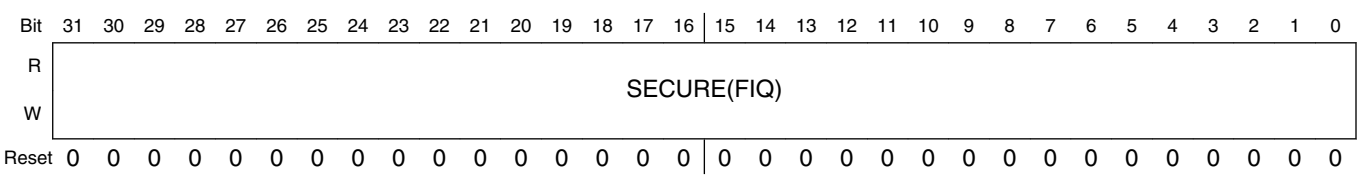
54.6.6 Interrupt Security (FIQ) Register: Irq 0 to 31 (TZIC_INTSEC*n*)

In a single security domain device, the INTSEC registers are used to set interrupts as FIQ or normal IRQ. Each bit in the register corresponds to an interrupt source available in the system. For each interrupt, if its bit is set to 1'b0, the interrupt is FIQ and will cause a high priority interrupt to the ARM platform. If its bit is set to 1'b1, the interrupt is normal and will cause a lower priority interrupt to the ARM platform.

Out of reset, all interrupts are configured as FIQ. If normal interrupts are required then these must be changed to 1'b1 after reset.

These registers can only be accessed by 32-bit supervisor transactions.

Addresses: TZIC_INTSEC0 is FFFC000h base + 80h offset = 0FFF_C080h
TZIC_INTSEC1 is FFFC000h base + 84h offset = 0FFF_C084h
TZIC_INTSEC2 is FFFC000h base + 88h offset = 0FFF_C088h
TZIC_INTSEC3 is FFFC000h base + 8Ch offset = 0FFF_C08Ch



TZIC_INTSEC*n* field descriptions

Field	Description
31–0 SECURE(FIQ)	Interrupt Status. When read, this register returns the status bit values of the associated interrupts. When written, this register sets the configuration for the associated interrupts. The corresponding interrupts for an INTSEC(X) are determined through the following formula: FIQ(X)[31] -> (32*X)+31 FIQ(X)[0] -> (32*X)+0 0 Interrupt is fast 1 Interrupt is normal

54.6.7 Enable Set Register: Irq 0 to 31 (TZIC_ENSET*n*)

The Enable Set Registers (ENSET) are used to enable interrupt requests to the core. Each bit in the register corresponds to an interrupt source available in the system. For each interrupt, when read, if its enable bit is set to 1'b1, the interrupt is enabled and will be transmitted to the ARM platform. If its bit is set to 1'b0, any activity on that interrupt will have no effect.

Writing a bit to 1'b1 in this register will enable the interrupt associated with that bit. Writing a bit to 1'b0 will have no effect. To clear an enable bit, the Enable Clear Registers must be used.

The values read in the ENSET register represent the currently enabled interrupts.

These registers can only be accessed by 32-bit supervisor transactions.

Addresses: TZIC_ENSET0 is FFFC000h base + 100h offset = 0FFF_C100h
TZIC_ENSET1 is FFFC000h base + 104h offset = 0FFF_C104h
TZIC_ENSET2 is FFFC000h base + 108h offset = 0FFF_C108h
TZIC_ENSET3 is FFFC000h base + 10Ch offset = 0FFF_C10Ch

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

TZIC_ENSET*n* field descriptions

Field	Description
31–0 INTENSET	Interrupt Enable Set. When written, this register sets the enable bits of the associated interrupts. When read, this register returns the enable bit values of the associated interrupts. The corresponding interrupts for register ENSETX are determined through the following formula: INTENSET[31] -> (32*X)+31 INTENSET[0] -> (32*X)+0

TZIC_ENSET*n* field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled

54.6.8 Enable Clear Register: Irq 0 to 31 (TZIC_ENCLEAR*n*)

The Enable Clear Registers (ENCLEAR) are used to disable interrupt requests to the core. Each bit in the register corresponds to an interrupt source available in the system. For each interrupt, when read, if its enable bit is set to 1'b1, the interrupt is enabled and will be transmitted to the ARM platform. If its bit is set to 1'b0, any activity on that interrupt will have no effect.

Writing a bit to 1'b1 in this register will disable the interrupt associated with that bit. Writing a bit to 1'b0 will have no effect. To set an enable bit, the Enable Set Registers must be used.

The values read in the ENCLEAR register represent the currently enabled interrupts.

These registers can only be accessed by 32-bit supervisor accesses.

Addresses: TZIC_ENCLEAR0 is FFFC000h base + 180h offset = 0FFF_C180h
TZIC_ENCLEAR1 is FFFC000h base + 184h offset = 0FFF_C184h
TZIC_ENCLEAR2 is FFFC000h base + 188h offset = 0FFF_C188h
TZIC_ENCLEAR3 is FFFC000h base + 18Ch offset = 0FFF_C18Ch

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

TZIC_ENCLEAR*n* field descriptions

Field	Description
31–0 INTENCLEAR	Interrupt Enable Cleared. When written, this register clears the enable bits of the associated interrupts. When read, this register returns the enable bit values of the associated interrupts. The corresponding interrupts for register ENCLEARX are determined through the following formula: INTENCLEAR[31] -> (32*X)+31 INTENCLEAR[0] -> (32*X)+0 0 Interrupt is disabled 1 Interrupt is enabled

54.6.9 Source Set Register: Irq 0 to 31 (TZIC_SRCSET*n*)

The Source Set Registers (SRCSET) are used to determine which interrupts are currently asserted and to force interrupts to an asserted state regardless of activity on the interrupt lines. Each bit in the register corresponds to an interrupt source available in the system. For each interrupt, when read, if its source bit is set to 1'b1, the interrupt is asserted and pending action from the ARM platform. If its bit is set to 1'b0, the interrupt is not asserted.

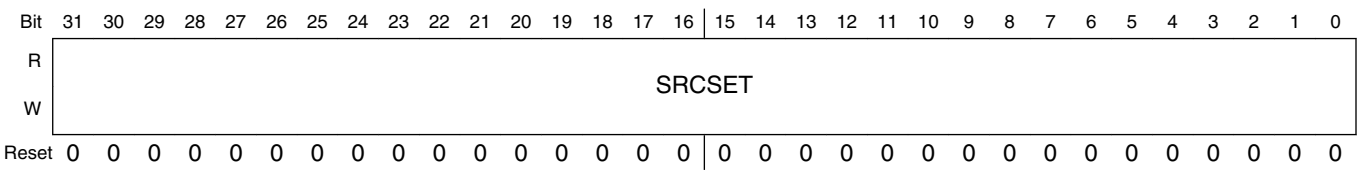
Writing a bit to 1'b1 in this register will override the interrupt line and force the corresponding interrupt to assert. Writing a bit to 1'b0 will have no effect. To clear a source bit, the Source Clear Registers or SWINT must be used.

If a source bit is set which corresponds to an interrupt not enabled in the ENSET/ ENCLEAR registers, the source bit will be 1'b1 but will not be considered pending and will not cause an interrupt to the core.

The values read in the SRCSET register represent the currently asserted interrupts.

These registers can only be accessed by 32-bit supervisor accesses.

Addresses: TZIC_SRCSET0 is FFFC000h base + 200h offset = 0FFF_C200h
TZIC_SRCSET1 is FFFC000h base + 204h offset = 0FFF_C204h
TZIC_SRCSET2 is FFFC000h base + 208h offset = 0FFF_C208h
TZIC_SRCSET3 is FFFC000h base + 20Ch offset = 0FFF_C20Ch



TZIC_SRCSET*n* field descriptions

Field	Description
31–0 SRCSET	Interrupt Source Set. When written, this register forces the source bits of the associated interrupts. When read, this register returns the source bit values of the associated interrupts. The corresponding interrupts for register SRCSETX are determined through the following formula: SRCSET[31] -> (32*X)+31 SRCSET[0] -> (32*X)+0 0 Interrupt is not asserted 1 Interrupt is asserted

54.6.10 Source Clear Register: Irq 0 to 31 (TZIC_SRCCLEAR*n*)

The Source Clear Registers (SRCCLEAR) are used to determine which interrupts are currently asserted and to clear interrupts from the asserted state that were triggered by software, either through the SRCSET or SWINT registers. Each bit in the register corresponds to an interrupt source available in the system. For each interrupt, when read, if its source bit is set to 1'b1, the interrupt is asserted and is pending action from the ARM platform. If its bit is set to 1'b0, the interrupt is not asserted.

Writing a bit to 1'b1 in this register will clear an asserted interrupt that was triggered by software. Writing a bit to 1'b0 will have no effect. To set a source bit, the Source Set Registers or Software Triggered Interrupt Register must be used.

The values read in the SRCCLEAR register represent the currently asserted interrupts.

These registers can only be accessed by 32-bit supervisor accesses.

Addresses: TZIC_SRCCLEAR0 is FFFC000h base + 280h offset = 0FFF_C280h
TZIC_SRCCLEAR1 is FFFC000h base + 284h offset = 0FFF_C284h
TZIC_SRCCLEAR2 is FFFC000h base + 288h offset = 0FFF_C288h
TZIC_SRCCLEAR3 is FFFC000h base + 28Ch offset = 0FFF_C28Ch

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

TZIC_SRCCLEAR*n* field descriptions

Field	Description
31–0 SRCCLEAR	Interrupt Source Clear. When written, this register forces the source bits of the associated interrupts. When read, this register returns the source bit values of the associated interrupts. The corresponding interrupts for register SRCCLEARX are determined through the following formula: SRCCLEAR[31] -> (32*X)+31 SRCCLEAR[0] -> (32*X)+0 0 Interrupt is not asserted 1 Interrupt is asserted

54.6.11 Priority Register: Irq 0 to 3 (TZIC_PRIORITY*n*)

The Priority Registers (PRIORITY) are used to configure the priority of each interrupt for masking purposes.

Each eight bit priority field is affected by an integration parameter which determines the available resolution of priority levels. The number of bits available can vary from four to eight, with the lowest resolution only implementing bits [7:4], and the highest resolution implementing bits [7:0]. The implemented number of bits can be found in a manner similar to that described in Priority Mask Register (TZIC_PRIOMASK).

The priority scheme used is such that an interrupt assigned to 0 has the highest priority.

These registers can only be accessed by 32-bit or 8-bit supervisor transactions.

Addresses: FFFC000h base + 400h offset + (4d × *n*), where *n* = 0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRIO3								PRIO2								PRIO1								PRIO0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TZIC_PRIORITY*n* field descriptions

Field	Description
31–24 PRIO3	Interrupt Priority. This field stores the priority value for its corresponding interrupt. The interrupts for register PRIORITYX are determined through the following formula: PRIO3 -> (4*X)+3 PRIO2 -> (4*X)+2 PRIO1 -> (4*X)+1 PRIO0 -> (4*X)+0 00000000 Interrupt priority is set to 0 00000001 Interrupt priority is set to 1 11111111 Interrupt priority is set to 255 If AWPROT[1] == 1'b1 (nonsecure)
23–16 PRIO2	Description and settings as for PRIO3.
15–8 PRIO1	Description and settings as for PRIO3.
7–0 PRIO0	Description and settings as for PRIO3.

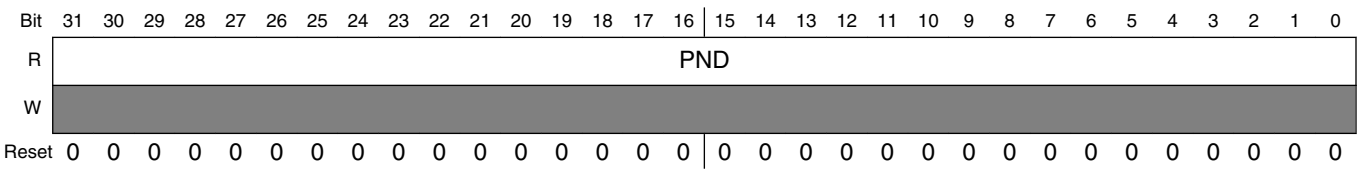
54.6.12 Pending Register: Irq 0 to 31 (TZIC_PND*n*)

The Pending Registers (PND) are used to determine which interrupts are currently pending action from the ARM platform. Each bit in the register corresponds to an interrupt source available in the system. For each interrupt, if its pending bit is set to 1'b1, the interrupt is enabled in the ENSET/ENCLEAR registers and asserted either on the interrupt inputs or in the SRCSET/SRCCLEAR registers, and therefore is pending action from the ARM platform. Priority values have no effect on whether an interrupt is pending.

This is a read-only register, writes will be ignored.

These registers can only be accessed by 32-bit supervisor transactions.

Addresses: TZIC_PND0 is FFFC000h base + D00h offset = 0FFF_CD00h
TZIC_PND1 is FFFC000h base + D04h offset = 0FFF_CD04h
TZIC_PND2 is FFFC000h base + D08h offset = 0FFF_CD08h
TZIC_PND3 is FFFC000h base + D0Ch offset = 0FFF_CD0Ch



TZIC_PND*n* field descriptions

Field	Description
31–0 PND	Interrupt Pending Status. When read, this register returns the pending bit values of the associated interrupts, indicating which interrupts are enabled and asserted. The corresponding interrupts for register PNDX are determined through the following formula: PND[31] -> (32*X)+31 PND[0] -> (32*X)+0 0 Interrupt is not pending 1 Interrupt is pending

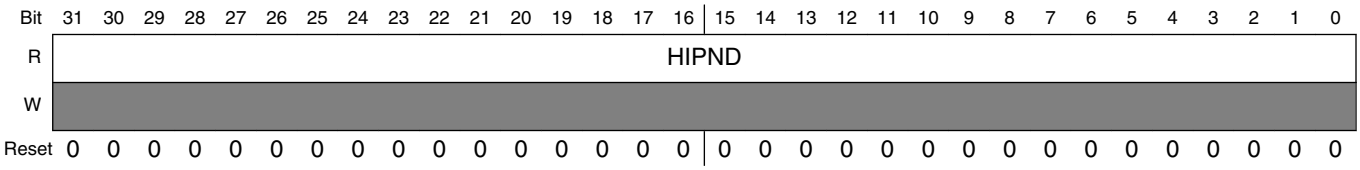
54.6.13 High Priority Pending Register: Irq 0 to 31 (TZIC_HIPND*n*)

The High Priority Pending Registers (HIPND) are used to determine which interrupts are currently pending at the highest active priority. Each bit in the register corresponds to an interrupt source available in the system. For each interrupt, if its pending bit is set to 1'b1, the interrupt is pending action from the ARM platform and its priority is the highest level amongst all pending enabled interrupts. See Pending Register: Irq 0 to 31 (TZIC_PND*n*) for more information about pending interrupts.

This is a read-only register, writes will be ignored.

These registers can only be accessed by 32-bit supervisor transactions.

Addresses: TZIC_HIPND0 is FFFC000h base + D80h offset = 0FFF_CD80h
TZIC_HIPND1 is FFFC000h base + D84h offset = 0FFF_CD84h
TZIC_HIPND2 is FFFC000h base + D88h offset = 0FFF_CD88h
TZIC_HIPND3 is FFFC000h base + D8Ch offset = 0FFF_CD8Ch



TZIC_HIPND*n* field descriptions

Field	Description
31–0 HIPND	High Priority Interrupt Pending Status. When read, this register returns the pending bit values of the associated interrupts which are set to the highest currently active priority level. The corresponding interrupts for register HIPNDX are determined through the following formula: HIPND[31] -> (32*X)+31 HIPND[0] -> (32*X)+0 0 Interrupt is not pending at the higest priority level 1 Interrupt is pending at the higest priority level

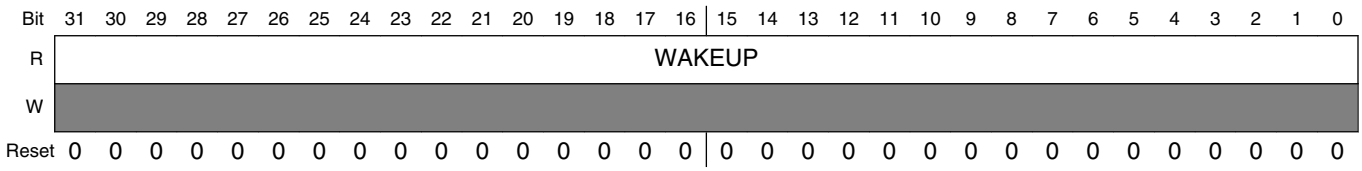
54.6.14 Wakeup Config Register: Irq 0 to 31 (TZIC_WAKEUP*n*)

The Wakeup Conguration Registers (WAKEUP) are used to set which interrupts will assert the tzic_wakeup_request signal. Each bit in the register corresponds to an interrupt source available in the system. For each interrupt, if its wakeup bit is set to 1'b1 assertion of the interrupt will cause the wakeup request signal to assert combinationally. When its wakeup bit is set to 1'b0, the interrupt will not assert the wakeup request signal.

The values read in the WAKEUP register represent interrupts currently set to wake up the system. Out of reset, all interrupts are configured not to wake up the system. Non-secure transactions can only enable or disable wakeup configuration of interrupts configured as non-secure. Secure transactions can enable or disable wakeup configuration of any interrupts.

These registers can only be accessed by 32-bit supervisor transactions.

Addresses: TZIC_WAKEUP0 is FFFC000h base + E00h offset = 0FFF_CE00h
TZIC_WAKEUP1 is FFFC000h base + E04h offset = 0FFF_CE04h
TZIC_WAKEUP2 is FFFC000h base + E08h offset = 0FFF_CE08h
TZIC_WAKEUP3 is FFFC000h base + E0Ch offset = 0FFF_CE0Ch



TZIC_WAKEUP*n* field descriptions

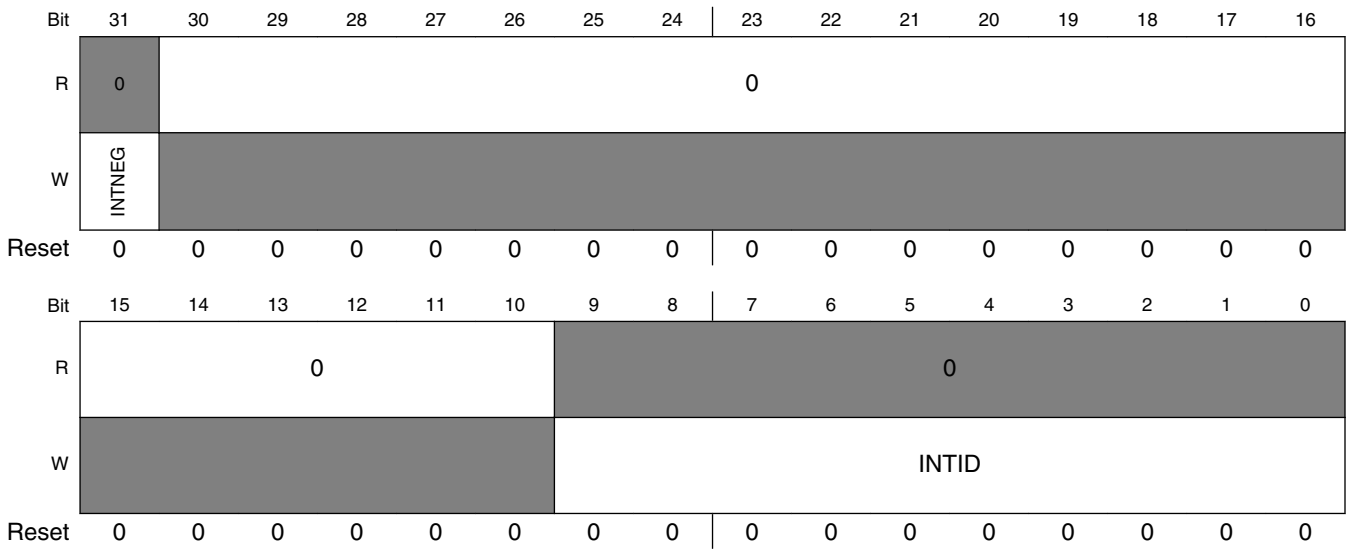
Field	Description
31–0 WAKEUP	Wakeup Configuration. When read, this register returns the wakeup bit values of the associated interrupts. When written, this register sets the configuration for the associated interrupts. The corresponding interrupts for register WAKEUPX are determined through the following formula: WAKEUP[31] -> (32*X)+31 WAKEUP[0] -> (32*X)+0 0 Interrupt will not assert tzic_wakeup_request 1 Interrupt will assert tzic_wakeup_request

54.6.15 Software Interrupt Trigger Register (TZIC_SWINT)

The interrupt controller software interrupt register (SWINT) can be used by software to assert or negate an interrupt at a specific interrupt number. Writing the number of the desired interrupt will either trigger an interrupt to assert if not already asserted by hardware, or negate an interrupt that was originally asserted by software. An interrupt asserted by hardware cannot be negated by this register. Triggering an interrupt with the SWINT is equivalent to asserting the associated interrupt input pin, except regarding the tzic_wakeup_request output. It will be asserted in the SRCSET/SRCCLEAR regitsers, but will not be pending unless the interrupt is enabled. A software interrupt will not cause a wakeup request.

This register can only be accessed by 32-bit supervisor transactions.

Address: TZIC_SWINT is FFFC000h base + F00h offset = 0FFF_CF00h



TZIC_SWINT field descriptions

Field	Description
31 INTNEG	Interrupt Negate. This bit indicates whether the interrupt ID in the INTID field is being asserted or negated. Assuming the interrupt is not being asserted by hardware, when 1'b0, an interrupt is asserted, when 1'b1, an interrupt is negated.
30–10 Reserved	This read-only field is reserved and always has the value zero. Reserved
9–0 INTID	Interrupt ID. This field indicates the number of the interrupt to be triggered (asserted or negated). Triggering an interrupt outside of the maximum number of interrupts will result in UPREDICTABLE behavior.

Chapter 55
Universal Asynchronous Receiver/Transmitter (UART)

55.1 Overview

Universal Asynchronous Receiver/Transmitter (UART) provides serial communication capability with external devices through an RS-232 cable or through use of external circuitry that converts infrared signals to electrical signals (for reception) or transforms electrical signals to signals that drive an infrared LED (for transmission) to provide low speed IrDA compatibility. UART supports NRZ encoding format and IrDA-compatible infrared slow data rate (SIR) format.

The following figure is the UART block diagram.

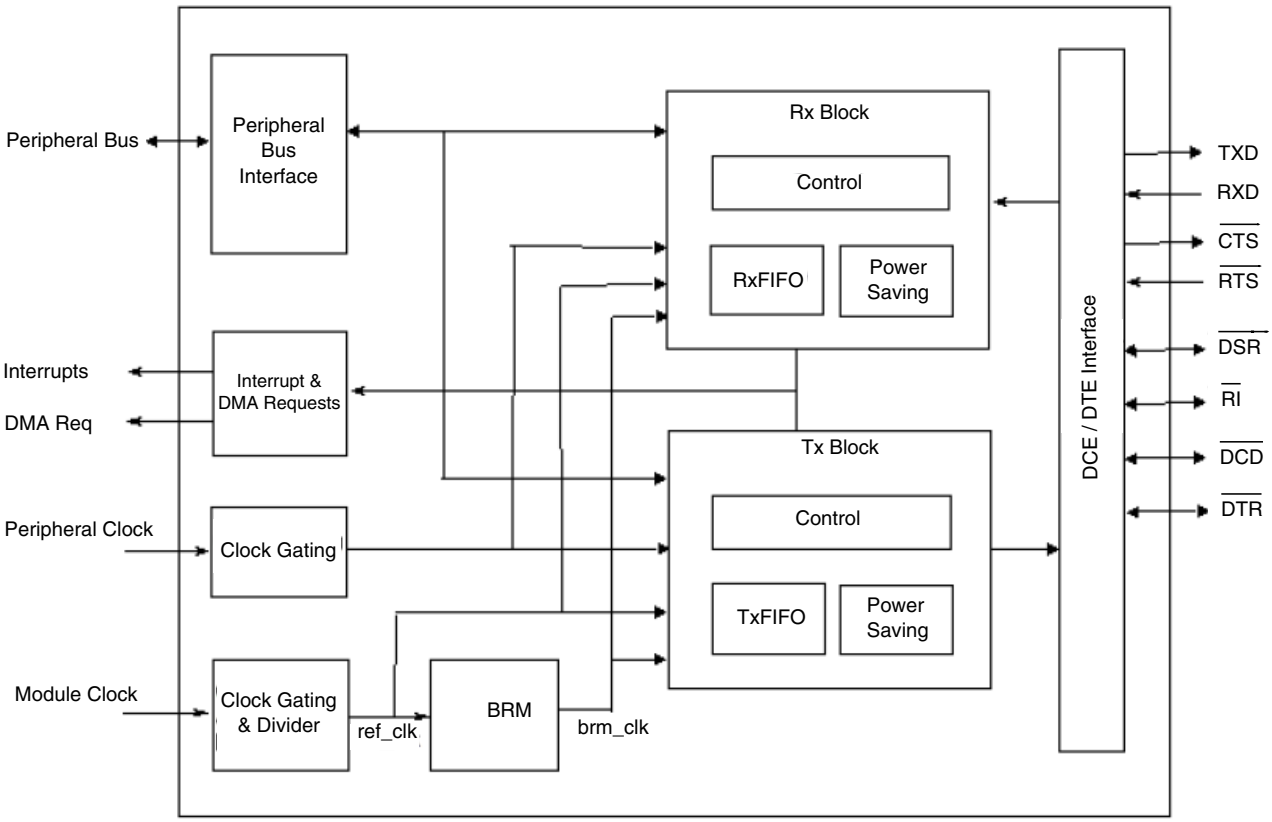


Figure 55-1. UART Block Diagram

55.1.1 Features

The UART includes the following features:

- High-speed TIA/EIA-232-F compatible, up to 4.0 Mbit/s
- Serial IR interface low-speed, IrDA-compatible (up to 115.2 Kbit/s)
- 7 or 8 data bits
- 1 or 2 stop bits
- Programmable parity (even, odd, and no parity)
- Hardware flow control support for request to send ($\overline{RTS}$) and clear to send ($\overline{CTS}$) signals
- Edge-selectable $\overline{RTS}$ and edge-detect interrupts
- Status flags for various flow control and FIFO states
- Voting logic for improved noise immunity (16x oversampling)
- Transmitter FIFO empty interrupt suppression
- UART internal clocks enable/disable
- Auto baud rate detection (up to 115.2 Kbit/s)
- Receiver and transmitter enable/disable for power saving

- DCE/DTE capability
- $\overline{\text{RTS}}$, IrDA asynchronous wake (AIRINT), receive asynchronous wake (AWAKE), $\overline{\text{RI}}$ (DTE only), $\overline{\text{DCD}}$ (DTE only), $\overline{\text{DTR}}$ (DCE only) and $\overline{\text{DSR}}$ (DTE only) interrupts wake the processor from STOP mode
- Maskable interrupts
- Two DMA Requests (TxFIFO DMA Request and RxFIFO DMA Request)
- Escape character sequence detection
- Software reset ($\overline{\text{SRST}}$)
- Two independent, 32-entry FIFOs for transmit and receive
- The peripheral clock can be totally asynchronous with the module clock. The module clock determines baud rate. This allows frequency scaling on peripheral clock (such as during DVFS mode) while remaining the module clock frequency and baud rate.

55.1.2 Modes of Operation

- Serial RS-232NRZ mode
- IrDA mode

55.2 External Signals

The following table lists conventions for representing signals.

Table 55-1. Module Signal Conventions

Category	Convention	Example(s
Off-chip signal	Uppercase (all capital letters)	TXD
Internal signal[1]	Lowercase italics	*core_int*
Active low signal	_B (_b) suffix or overbar	RESET_EN_B or $\overline{\text{RESET\_EN}}$
Range of bussed or commonly named signals	Beginning and end points of the range are: • Separated by a colon. • Surrounded by square brackets.	ADDR[31:0] CSE_B[7:0] or $\overline{\text{CSE}}$[7:0]
Individual signal in a range of bussed or commonly named signals	Individual number in the range appears without a colon or square brackets	ADDR31 CSE0_B or $\overline{\text{CSE0}}$

1. Internal signals are for reference only in descriptions of internal module or SoC functionality.

The table below describes all UART signals that connect off-chip.

Table 55-2. Off-Chip Module Signals

Signal name	I/O	Active state	Description	Reset state
Serial / IrDA Signals				

Table continues on the next page...

Table 55-2. Off-Chip Module Signals (continued)

Signal name	I/O	Active state	Description	Reset state
RXD	I		Serial / infrared data receive	
TXD	O		Serial/infrared data transmit	High
Modem Control Signals				
$\overline{\text{CTS}}$	O	Low	Clear to send	High
$\overline{\text{RTS}}$	I	Low	Request to send	
$\overline{\text{DSR}}$	I/O	Low	Data set ready	High
$\overline{\text{DCD}}$	I/O	Low	Data carrier detected	High
$\overline{\text{DTR}}$	I/O	Low	Data terminal ready	
$\overline{\text{RI}}$	I/O	Low	Ring indicator	High
Interrupts				
interrupt_uart	O	Low	UART interrupt	High
DMA Requests				
dma_req_rx	O	Low	Receiver DMA request	High
dma_req_tx	O	Low	Transmitter DMA request	High
Clocks				
peripheral_clock	I		Peripheral clock	
module_clock	I		Clock source for the module's logic	
Special Signals				
stop_req	I	High	Module stop mode	
doze_req	I	High	Module doze mode	
debug_req	I	High	Module debug	

55.2.1 Detailed Signal Descriptions

55.2.1.1 Serial/IrDA Signals

55.2.1.1.1 RXD - Data Receive

Input asynchronous data receive in Serial and IrDA modes.

55.2.1.1.2 TXD - Data Transmit

Output asynchronous data transmit in Serial and IrDA modes.

55.2.1.2 Modem Control Signals

55.2.1.2.1 $\overline{\text{CTS}}$ - Clear To Send

Output in DCE and DTE mode. This signal informs the remote modem that UART is ready to receive data.

55.2.1.2.2 $\overline{\text{RTS}}$ - Request To Send

Input in DCE and DTE mode. This signal informs UART that remote modem is ready to receive data.

55.2.1.2.3 $\overline{\text{DSR}}$ - Data Set Ready

Input in DTE mode. Indicates to UART that remote modem is operational.

Output in DCE mode. Indicates to remote modem that UART is operational.

55.2.1.2.4 $\overline{\text{DCD}}$ - Data Carrier Detected

Input in DTE mode. Indicates to UART that a good carrier is being received from the remote modem.

Output in DCE mode. Indicates to remote device that a good carrier is being received from the UART.

55.2.1.2.5 $\overline{\text{DTR}}$ - Data Terminal Ready

Input in DCE mode. Indicates to UART (in DCE mode) that remote device (in DTE mode) is operational.

Output in DTE mode. Indicates to remote modem (in DCE mode) that UART (in DTE mode) is operational.

55.2.1.2.6 $\overline{\text{RI}}$ - Ring Indicator

Input in DTE mode. Indicates to UART that remote modem is detecting a ringing tone.

Output in DCE mode. Indicates to remote device that UART is detecting a ringing tone.

55.2.1.3 Interrupt Signals

55.2.1.3.1 *interrupt_uart* - UART Interrupt

Output interrupt request.

55.2.1.4 DMA Request Signals

55.2.1.4.1 *dma_req_rx* - Receiver DMA Request

Output DMA Request signal for receiver interface.

55.2.1.4.2 *dma_req_tx* - Transmitter DMA Request

Output DMA Request signal for transmitter interface. Set at 0 when TXDMAEN (UCR1[3]) is at 1 and TRDY (USR1[13]) is also at 1.

55.2.1.5 Clock Signals

55.2.1.5.1 *peripheral_clock* - Peripheral Clock

See Clocks for more information about *peripheral_clock* .

55.2.1.5.2 *module_clock* - Module Clock

See Clocks for more information about *module_clock* .

55.2.1.6 Special Signals

55.2.1.6.1 *stop_req* - Stop Mode

Input stop mode. Indicates to UART that ARM platform is going to enter in Stop Mode and clocks are going to stop running.

See Low Power Modes for more information about Stop Mode.

55.2.1.6.2 *doze_req* - Doze Mode

Input doze mode. ARM platform requests UART to switch in doze mode (power saving mode).

See Low Power Modes for more information about Doze Mode.

55.2.1.6.3 *debug_req* - Debug Mode

Input debug mode. Indicates UART it has to enter in debug mode.

See UART Operation in System Debug State, for more information about Debug Mode.

55.3 Functional Description

This section provides a complete functional description of the block.

55.3.1 Interrupts and DMA Requests

See the following table for the lists of all interrupt and DMA signals and associated interrupt and DMA sources of the UART. See register description section for explanation of interrupt/DMA enable and status.

Table 55-3. Interrupts and DMA

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
interrupt_uart	RRDYEN	UCR1 (bit 9)	RRDY	USR1 (bit 9)
	IDEN	UCR1 (bit 12)	IDLE	USR2 (bit 12)
	DREN	UCR4 (bit 0)	RDR	USR2 (bit 0)
	RXDSEN	UCR3 (bit 6)	RXDS	USR1 (bit 6)
	ATEN	UCR2 (bit 3)	AGTIM	USR1 (bit 8)
interrupt_uart	TXMPTYEN	UCR1 (bit 6)	TXFE	USR2 (bit 14)
	TRDYEN	UCR1 (bit 13)	TRDY	USR1 (bit 13)
	TCEN	UCR4 (bit 3)	TXDC	USR2 (bit 3)

Table continues on the next page...

Table 55-3. Interrupts and DMA (continued)

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
interrupt_uart	OREN	UCR4 (bit 1)	ORE	USR2 (bit 1)
	BKEN	UCR4 (bit 2)	BRCD	USR2 (bit 2)
	WKEN	UCR4 (bit 7)	WAKE	USR2 (bit 7)
	ADEN	UCR1 (bit 15)	ADET	USR2 (bit 15)
	ACIEN	UCR3 (bit 0)	ACST	USR2 (bit 11)
	ESCI	UCR2 (bit 15)	ESCF	USR1 (bit 11)
	ENIRI	UCR4 (bit 8)	IRINT	USR2 (bit 8)
	AIRINTEN	UCR3 (bit 5)	AIRINT	USR1 (bit 5)
	AWAKEN	UCR3 (bit 4)	AWAKE	USR1 (bit 4)
	FRAERREN	UCR3 (bit 11)	FRAERR	USR1 (bit 10)
	PARERREN	UCR3 (bit 12)	PARITYERR	USR1 (bit 15)
	RTSDEN	UCR1 (bit 5)	RTSD	USR1 (bit 12)
	RTSEN	UCR2 (bit 4)	RTSF	USR2 (bit 4)
	DTREN (DCE)	UCR3 (bit 13)	DTRF	USR2 (bit 13)
	RI (DTE)	UCR3 (bit 8)	RIDELT	USR2 (bit 10)
	DCD (DTE)	UCR3 (bit 9)	DCDDELT	USR2 (bit 6)
	DTRDEN	UCR3 (bit 3)	DTRD	USR1 (bit 7)
dma_req_rx	RXDMAEN	UCR1 (bit 8)	RRDY	USR1 (bit 9)
	ATDMAEN	UCR1 (bit 2)	AGTIM	USR1 (bit 8)
	IDDMAEN	UCR4 (bit 6)	IDLE	USR2 (bit 12)
dma_req_tx	TXDMAEN	UCR1 (bit 3)	TRDY	USR1 (bit 13)

55.3.2 Clocks

This section describes clocks and special clocking requirements of the UART.

55.3.2.1 Clock requirements

UART module receives 2 clocks, *peripheral_clock* and *module_clock*. The *peripheral_clock* is used as write clock of the TxFIFO, read clock of the RxFIFO and synchronization of the modem control input pins. It must always be running when UART is enabled. There is an exception in stop mode (see Clocking in Low-Power Modes).

The *module_clock* is for all the state machines, writing RxFIFO, reading TxFIFO, etc. It must always be running when UART is sending or receiving characters.This clock is used in order to allow frequency scaling on *peripheral_clock* without changing configuration of baud rate (*module_clock* staying at a fixed frequency).

The constraints on *peripheral_clock* and *module_clock* are as follows:

- *peripheral_clock* and *module_clock* can totally be asynchronous. Off course, they can also be synchronous.
- Due to the 16x oversampling of the incoming characters, *module_clock* frequency must always be greater or equal to 16x the maximum baud rate. For example, if max baud rate is 4 Mbit/s, *module_clock* must be greater or equal to 4 M x 16 = 64MHz.

NOTE

The restriction that *peripheral_clock* frequency must be higher or equal to 16x baud rate has been removed. There is no limitation on *peripheral_clock* frequencyto baud rate.

55.3.2.2 Maximum Baud Rate

The max baud rate the UART can support is determined by the max frequency of the *module_clock* and logic synthesis results.

For example, if the SoC can provide the fastest *module_clock* 66.5 MHz and the UART synthesis timing is acceptable under this constraint, the UART can transmit and receive serial data with the maximum baud rate 66.5M/16 = 4.15 Mbit/s.

The UART supports serial IR interface low speed. In the low speed IrDA mode, the max baud rate is 115.2Kbit/s. To support the 115.2Kbit/s, *module_clock* frequency must be higher or equal to 1.8432MHz.

55.3.2.3 Clocking in Low-Power Modes

The UART supports 2 low-power modes: DOZE and STOP.

In STOP mode (input pin *stop_req* is at '1'), the UART doesn't need any clock. In this mode the UART can wake-up the ARM platform with the asynchronous interrupts (refer to Low Power Modes). An application of this feature is when the system must be waken-up by the arrival of a frame of characters.

- If before entering in Stop mode the software has enabled RTSDEN interrupt, then when RTS will change state (put at '0' by external device started to send), the

asynchronous interrupt will wake-up the system, *peripheral_clock* and *module_clock* will be provided to the UART before first start bit, so that no data will be lost.
- If RTS doesn't change state (already at '0' before entering in Stop mode), then wake-up interrupt (AWAKE) will be sent at the arrival of first Start bit (on falling edge). In this case, the UART must receive the *peripheral_clock* and *module_clock* during the first half of start bit to correctly receive this character (for example, at 115.2 Kbit/s, UART must receive *peripheral_clock* and *module_clock* at maximum 4.3 microseconds after falling edge of Start bit). If the UART receives *peripheral_clock* and *module_clock* too late, first character will be lost, and so should be dropped. Also, if autobaud detection is enabled, the first character won't be correctly received and another autobaud detection will need to be initiated.

In Doze mode, UART behavior is programmable through DOZE bit (UCR1[1]). If DOZE bit is set to '1', then UART is disabled in Doze mode, and in consequence, UART clocks can be switched-off (after being sure UART is not transmitting nor receiving). On the contrary, if DOZE bit is set to '0', UART is enabled and it must receive *peripheral_clock* and *module_clock* .

55.3.3 General UART Definitions

Definitions of terms that occur the following discussions are given in this section.

- Bit Time-The period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency).
- Start bit-The bit time of a logic 0 that indicates the beginning of a data frame. A start bit begins with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1.
- Stop bit-1 bit time of logic 1 that indicates the end of a data frame.
- BREAK-A frame in which all of the data bits, including the stop bit, are logic 0. This type of frame is usually sent to signal the end of a message or the beginning of a new message.
- Mark - When no data is being sent, the serial port's transmit pin's voltage is 1 and is said to be in a MARK state.
- Space - The serial port can also be forced to keep the transmit pin at a 0 and is said to be the SPACE or BREAK state.
- Frame-A start bit followed by a specified number of data or information bits and terminated by a stop bit. The number of data or information bits depends on the format specified and must be the same for the transmitting device and the receiving device. The most common frame format is 1 start bit followed by 8 data bits (least significant bit first) and terminated by 1 stop bit. An additional stop bit and a parity bit also can be included.

- Framing Error-An error condition that occurs when the stop bit of a received frame is missing, usually when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors can go undetected if a data bit in the expected stop bit time happens to be a logic 1. A framing error is always present on the receiver side when the transmitter is sending BREAKs. However, when the UART is programmed to expect 2 stop bits and only the first stop bit is received, this is not a framing error by definition.
- Parity Error-An error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the RXD input. Parity error is calculated only after an entire frame is received.
- Idle-One in NRZ encoding format and selectable polarity in IrDA mode.
- Overrun Error-An error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART receive buffer (RxFIFO). An overrun error indicates that the software reading the buffer (RxFIFO) is not keeping up with the actual reception of characters on the RXD input.

55.3.3.1 $\overline{\text{RTS}}$ - UART Request To Send

The UART Request To Send input controls the transmitter. The modem or other terminal equipment signals the UART when it is ready to receive by setting '0' on the RTSpin.

Normally, the transmitter waits until this signal is active (low) before transmitting a character, however when the Ignore RTS (IRTS) bit is set, the transmitter sends a character as soon as it is ready to transmit. An interrupt (RTSD) can be posted on any transition of this pin and can wake the ARM platform from STOP mode on its assertion. When $\overline{\text{RTS}}$ is set to '1' during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode.

55.3.3.2 RTS Edge Triggered Interrupt

The input to the $\overline{\text{RTS}}$ pin can be programmed to generate an interrupt on a selectable edge.

See the table below for summary of the operation of the RTS edge triggered interrupt (RTSF).

To enable the $\overline{\text{RTS}}$ pin to generate an interrupt, set the request to send interrupt enable (RTSEN) bit (UCR2[4]) to 1. Writing 1 to the $\overline{\text{RTS}}$ edge triggered interrupt flag (RTSF) bit (USR2[4]) clears the interrupt flag. The interrupt can occur on the rising edge, falling

edge, or either edge of the $\overline{\text{RTS}}$ input. The request to send edge control (RTEC) field (UCR2[10:9]) programs the edge that generates the interrupt. When RTEC is set to 0x00 and RTSEN = 1, the interrupt occurs on the rising edge (default). When RTEC is set to 0x01 and RTSEN = 1, the interrupt occurs on the falling edge. When RTEC is set to 0x1X and RTSEN = 1, the interrupt occurs on either edge. This is a synchronous interrupt. The RTSF bit is cleared by writing 1 to it. Writing 0 to RTSF has no effect.

Table 55-4. $\overline{\text{RTS}}$ Edge Triggered Interrupt Truth Table

$\overline{\text{RTS}}$	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On…	interrupt_uart
X	0	X	X	0	Interrupt disabled	1
1->0	1	0	0	0	Rising edge	1
0->1	1	0	0	1	Rising edge	0
1->0	1	0	1	1	Falling edge	0
0->1	1	0	1	0	Falling edge	1
1->0	1	1	X	1	Either edge	0
0->1	1	1	X	1	Either edge	0

There is another $\overline{\text{RTS}}$ interrupt that is not programmable. The status bit RTSD asserts the *interrupt_uart* interrupt when the $\overline{\text{RTS}}$ delta interrupt enable = 1. This is an asynchronous interrupt. The RTSD bit is cleared by writing 1 to it. Writing 0 to the RTSD bit has no effect.

55.3.3.3 $\overline{\text{DTR}}$ - Data Terminal Ready

This signal indicates the general readiness of the Data Terminal Equipment (DTE). This signal is an input in DCE mode and an output in DTE mode. If the connection between the DCE and the DTE is established once, the $\overline{\text{DTR}}$ signal must remain active throughout the whole connection time.

In general the $\overline{\text{DTR}}$ and $\overline{\text{DSR}}$ signals are responsible for establishing the connection. $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ are responsible for the data transfer and the transfer direction in the case of a half-duplex configuration. The $\overline{\text{DTR}}$ signal is like a "main switch". If the $\overline{\text{DTR}}$ signal is inactive the $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ signals have no effect. In DCE mode, an interrupt (DTRD) can be posted on any transition of this pin and can wake the ARM platform from STOP mode on its assertion.

55.3.3.4 $\overline{\text{DSR}}$ - Data Set Ready

This signal indicates the general readiness of the DCE. This signal is an output in DCE mode and an input in DTE mode. The DCE uses this signal to inform the DTE that it is switched on, has completed all preparations and can communicate with the DTE.

In DTE mode, an interrupt (DTRD) can be posted on any transition of this pin and can wake the ARM platform from STOP mode on its assertion.

55.3.3.5 $\overline{\text{DTR/DSR}}$ Edge Triggered Interrupt

The $\overline{\text{DTR}}$ input pin (DCE mode) or $\overline{\text{DSR}}$ input pin (DTE mode) can be configured to cause an interrupt on a selectable edge.

See the table below for summary of the operation of the DTR/DSR edge triggered interrupt. To enable the interrupt, set the DTREN bit (UCR3[13]) to '1'. Write a "one" to the DTRF bit (USR2[13]) to clear the interrupt flag.

The interrupt can be configured to occur on either the rising, falling, or either edge of the $\overline{\text{DTR/DSR}}$ input. Write to the DPEC[1:0] bits (UCR3[15:14]) to program which edge will cause an interrupt. If the bits are set to 00b and DTREN = 1, the interrupt will occur on the rising edge (default). If the bits are set to 01b and DTREN = 1, the interrupt will occur on the falling edge. If the bits are set to 1Xb and DTREN = 1, the interrupt will occur on either edge.

Table 55-5. $\overline{\text{DTR/DSR}}$ Edge Triggered Interrupt Truth Table

$\overline{\text{DTR}}$ / $\overline{\text{DSR}}$	DTREN	DPEC[1]	DPEC[0]	DTRF	Interrupt occurs on:	interrupt_uart
X	0	X	X	0	turned off	1
1->0	1	0	0	0	rising edge	1
0->1	1	0	0	1	rising edge	0
1->0	1	0	1	1	falling edge	0
0->1	1	0	1	0	falling edge	1
1->0	1	1	X	1	either edge	0
0->1	1	1	X	1	either edge	0

55.3.3.6 $\overline{\text{DCD}}$ - Data Carrier Detect

This signal is an output in DCE mode and an input in DTE mode. If used, the DCE device uses this signal to inform the DTE it has detected the carrier signal and the connection will be set up. This signal remains active while the connection remains established.

In DTE mode this input can trigger an interrupt on changing state. This is achieved by setting to '1' the interrupt enable bit (DCD, UCR3[9]). The change state is reflected in DCDDELT (USR2[6]). Also, the state of the Data Carrier Detect input is mirrored in the status register DCDIN (USR2[5]).

55.3.3.7 $\overline{\text{RI}}$ - Ring Indicator

This signal is an output in DCE mode and an input in DTE mode. If used, the DCE device uses this signal to inform the DTE that a ring just occurred.

In DTE mode this input can trigger an interrupt on changing state. This is achieved by setting to '1' the interrupt enable bit (RI, UCR3[8]). The change state is reflected in RIDELT (USR2[10]). Also, the state of the Ring Indicator input is mirrored in the status register RIIN (USR2[9]).

55.3.3.8 $\overline{\text{CTS}}$ - Clear To Send

This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the $\overline{\text{CTS}}$ trigger level is programmed to trigger at 32 characters received and the receiver detects the valid start bit of the 33 character, it de-asserts this pin. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode.

55.3.3.9 Programmable $\overline{\text{CTS}}$ Deassertion

The $\overline{\text{CTS}}$ output can also be programmed to deassert when the RxFIFO reaches a certain level. Setting the CTS trigger level (UCR4[15:10]) at any value less than 32 deasserts the $\overline{\text{CTS}}$ pin on detection of the valid start bit of the N + 1 character (where N is the trigger level setting). However, the receiver continues to receive characters until the RxFIFO is full.

55.3.3.10 TXD - UART Transmit

This is the transmitter serial output. When operating in RS-232 mode, NRZ encoded data is . When operating in infrared mode, a 3/16 bit-period pulse is output for each 0 bit transmitted, and no pulse is output for each 1 bit transmitted.

For RS-232 applications, this pin must be connected to an RS-232 transmitter. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode. See Figure 55-2.

55.3.3.11 RXD - UART Receive

This is the receiver serial input. When operating in RS-232 mode, NRZ encoded data is expected. When operating in infrared mode, a narrow pulse is expected for each 0 bit received and no pulse is expected for each 1 bit received.

External circuitry must convert the IR signal to an electrical signal. RS-232 applications require an external RS-232 receiver to convert voltage levels. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode. See the figure below.

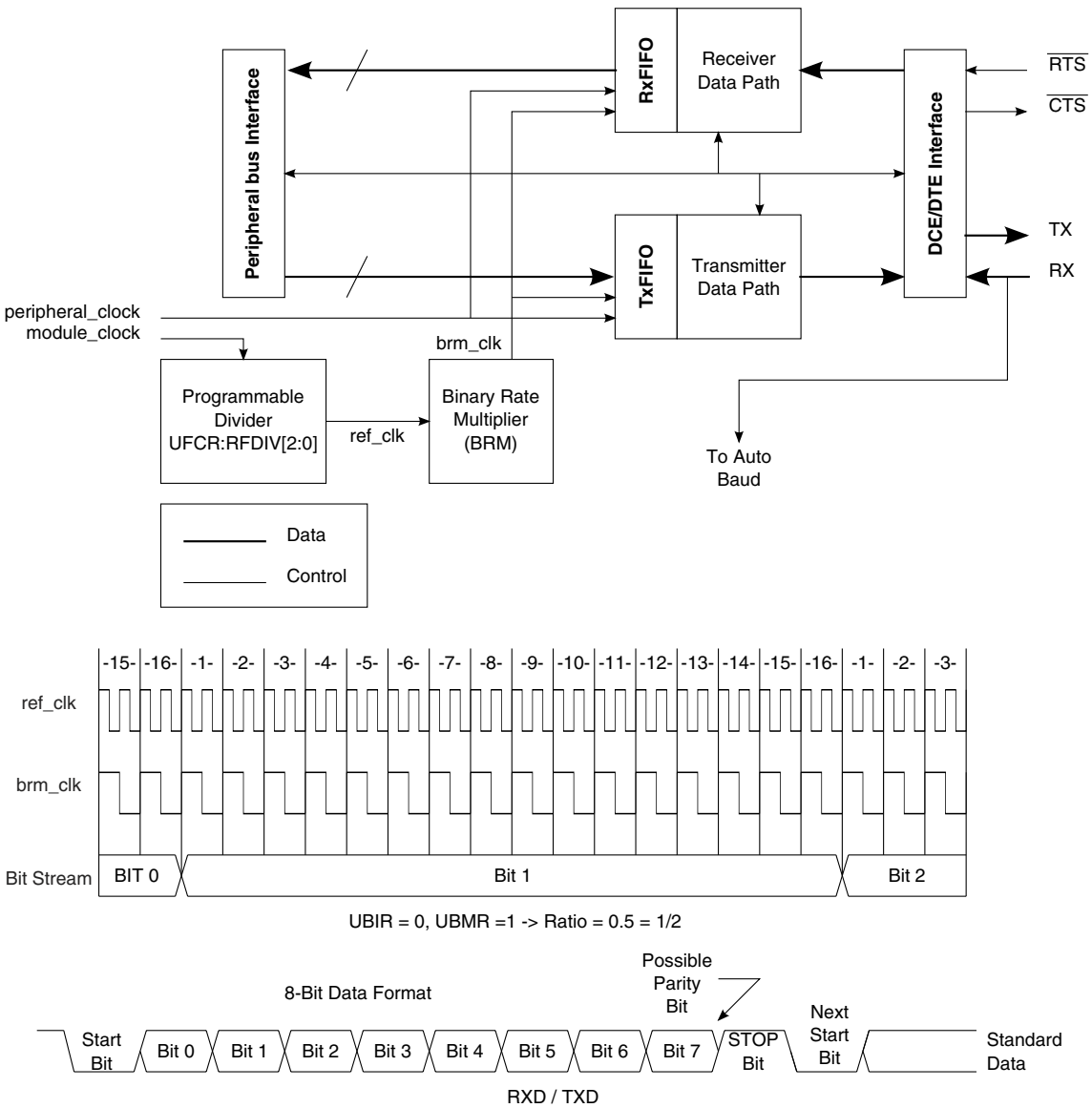


Figure 55-2. UART Simplified Block and Clock Generation Diagrams

55.3.4 Transmitter

The transmitter accepts a parallel character from the ARM platform and transmits it serially. The start, stop, and parity (when enabled) bits are added to the character.

When the ignore RTS bit (IRTS) is set, the transmitter sends a character as soon as it is ready to transmit. $\overline{\text{RTS}}$ can be used to provide flow-control of the serial data. When $\overline{\text{RTS}}$ is set to '1', the transmitter finishes sending the character in progress (if any), stops, and waits for $\overline{\text{RTS}}$ to be set to '0' again. Generation of BREAK characters and parity errors (for debugging purposes) is supported. The transmitter operates from the clock provided by the BRM. Normal NRZ encoded data is transmitted when the IR interface is disabled.

The transmitter FIFO (TxFIFO) contains 32 bytes. The data is written to TxFIFO by writing to the UTXD register with the byte data to the [7:0] bits. The data is written consecutively if the TxFIFO is not full. It is read (internally) consecutively if the TxFIFO is not empty. TXFULL bit (UTS[4]) can be used to control whether TXFIFO is full or not. The TxFIFO can be written regardless of the transmitter is disabled or enabled. If the UART is disabled, user can still write data into the TxFIFO correctly. But in this case the write access will yield to a transfer error.

55.3.4.1 Transmitter FIFO Empty Interrupt Suppression

The transmitter FIFO empty interrupt suppression logic suppresses the TXFE interrupt between writes to the TxFIFO.

When TxFIFO is empty, the software can either send one or several characters. If the software sends one character, it would write the character into the UTXD register, then that character is immediately transferred to the transmitter shift register, assuming the transmitter is already enabled. Without interrupt suppression logic, the TXFE interrupt flag would be set immediately. But, with this logic, the interrupt flag is set when the last bit of the character has been transmitted, for example, before the transmission of the parity bit (if exists) and the stop bit(s).

So, the suppression logic doesn't immediately send the TXFE interrupt flag. It allows the software to write another character to the TxFIFO before the interrupt flag is asserted.

When the transmitter shift register empties before another character is written to the TxFIFO, the interrupt flag is asserted. Writing data to the TxFIFO would release the interrupt flag. The interrupt flag is asserted on the following conditions:

- System Reset
- UART software reset

- When a single character has been written to Transmitter FIFO and then the Transmitter FIFO and the Transmitter Shift Register become empty until another character is written to the Transmitter FIFO
- The last character in the TxFIFO is transferred to the shift register, when TxFIFO contains two or more characters. See the figure below.

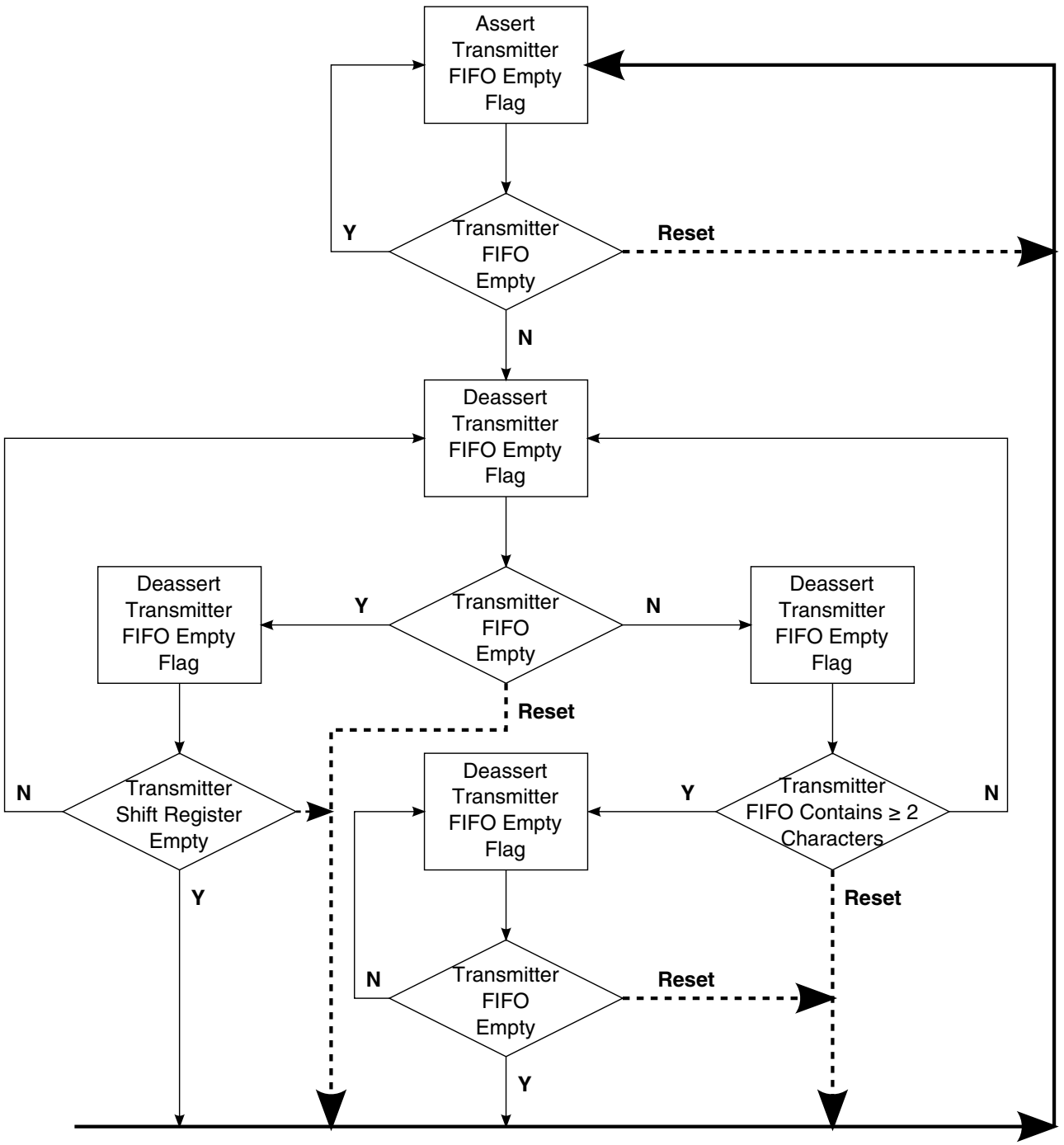


Figure 55-3. Transmitter FIFO Empty Interrupt Suppression Flow Chart

55.3.4.2 Transmitting a Break Condition

Asserting SNDBRK bit of the UCR1 Register forces the transmitter to send a break character (continuous zeros). The transmitter will finish sending the character in progress (if any) before sending break until this bit is reset.

The user is responsible to ensure that this bit is high for long enough to generate a valid BREAK. The transmitter samples SNDBRK after every bit is transmitted. Following completion of the BREAK transmission, the UART will transmit two mark bits. The user can continue to fill the FIFO and any character remaining will be transmitted when the break is terminated.

55.3.5 Receiver

See the figure below for the receiver flow chart.

The receiver accepts a serial data stream and converts it into parallel characters. When enabled, it searches for a start bit, qualifies it, and samples the following data bits at the bit-center. Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples.

Once the start bit is found, the data bits, parity bit (if enabled), and stop bits (either 1 or 2 depending on user selection) are shifted in. Parity is checked and its status reported in the URXD register when parity is enabled. Frame errors and BREAKs are also checked and reported. When a new character is ready to be read by the ARM platform from the RxFIFO, the receive data ready (RDR = USR2[0]) bit is asserted and an interrupt is posted (if DREN = UCR4[0] = 1). If the receiver trigger level is set to 2 (RXTL[5:0] = UFCR[5:0] = 2), and 2 chars have been received into RxFIFO, the receiver ready interrupt flag (RRDY = USR1[9]) is asserted and an interrupt is posted if the receiver ready interrupt enable bit is set (RRDYEN = UCR1[9] = 1). If the UART Receiver Register (URXD) is read once, and in consequence there is only 1 character in the RxFIFO, the interrupt generated by the RDR bit is automatically cleared. The RRDY bit is cleared when the data in the RxFIFO falls below the programmed trigger level.

Normal NRZ encoded data is expected when the IR interface is disabled.The RxFIFO contains 32 half-word entries. Characters received are written consecutively into this FIFO. If the FIFO is full and a 33rd characters is received, this character will be ignored and the USR2[ORE] bit will be set.

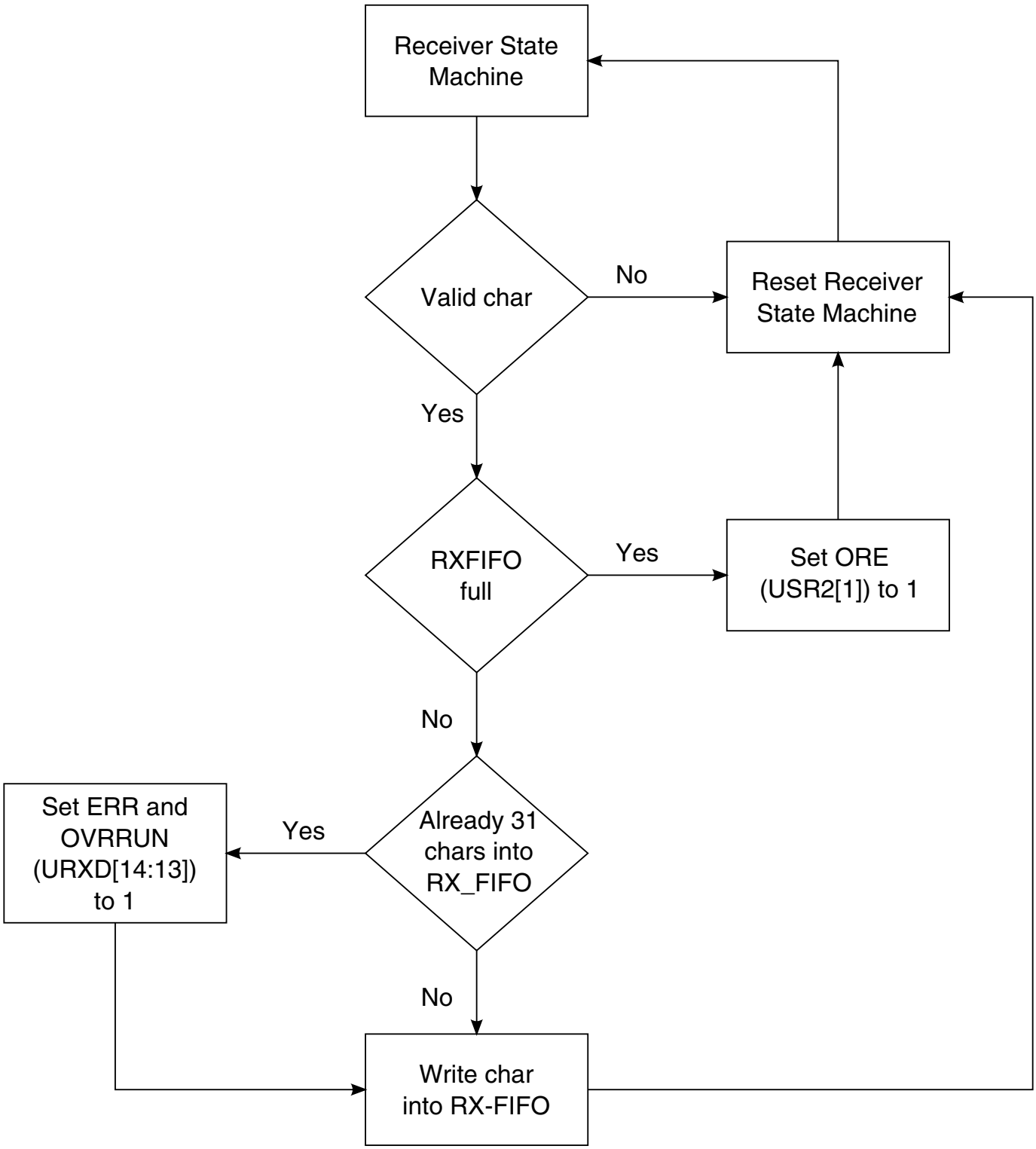


Figure 55-4. Receiver Flow Chart

55.3.5.1 Idle Line Detect

The receiver logic block includes the ability to detect an idle line. Idle lines indicate the end or the beginning of a message.

For an idle condition to occur:

- RxFIFO must be empty and
- RXD pin must be idle for more than a configured number of frames (ICD[1:0] = UCR1[11:10]).

When the idle condition detected interrupt enable (IDEN = UCR1[12]) is set and the line is idle for 4 (default), 8, 16, or 32 (maximum) frames, the detection of an idle condition flags an interrupt (see the table below). When an idle condition is detected, the IDLE (USR2[12]) bit is set. Clear the IDLE bit by writing 1 to it. Writing 0 to the IDLE bit has no effect.

Table 55-6. Detection Truth Table

IDEN	ICD [1]	ICD [0]	IDLE	*interrupt_uart*
0	X	X	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames

NOTE: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal. This table shows how this interrupt affects the *interrupt_uart* signal.

During a normal message there is no idle time between frames. When all of the information bits in a frame are logic 1s, the start bit ensures that at least one logic 0 bit time occurs for each frame so that the IDLE bit is not asserted.

55.3.5.2 Aging Character Detect

The receiver block also includes the possibility to detect when at least one character has been sitting into the RxFIFO for a time corresponding to 8 characters. This aging character capability allows the UART to inform the ARM platform that there is less character into the RxFIFO than the Rx trigger and, no new character has been detected on the RXD line.

The aging capability is a timer which starts to count as soon as the RxFIFO is not empty and its trigger level is not reached (RRDY=0). This counter is reset when either a RxFIFO read is performed or another character starts to present on the RXD line. If none of those two events occurs, the bit AGTIM (USR1[8]) is set when the counter has

measured a time corresponding to 8 characters. AGTIM is cleared by writing a 1 to it. AGTIM can flag an interrupt to ARM platform on *interrupt_uart* if ATEN (UCR2[3]) has been set.

To summarize, AGTIM is set when:

- There is at least one character into RxFIFO.
- No read has occurred on RxFIFO and RXD line has stayed high, for a time corresponding to 8 characters.
- The RxFIFO trigger is not reached (RRDY=0)

55.3.5.3 Receiver Wake

The WAKE bit (USR2[7]) is set when the receiver detects a qualified Start bit. For this, two conditions must be fulfilled, firstly a falling edge on RXD line must be detected and secondly the RXD line must stays at low level for more than a half-bit duration.

When the wake interrupt enable WKEN (UCR4[7]) bit is enabled, the receiver flags an interrupt (*interrupt_uart*) if the WAKE status bit is set. The WAKE bit is cleared by writing 1 to it. Writing 0 to the WAKE bit has no effect. The WAKE status bit can be asserted in either serial RS-232 mode or IR mode. The generation of the WAKE interrupt needs the clock *module_clock* .

When the asynchronous wake interrupt (AWAKE) is enabled (AWAKEN = UCR3[4] = 1), and the ARM platform is in STOP mode, and UART clocks have been shut-off, then a falling edge detected on the receive pin (RXD) asserts the AWAKE bit (USR1[4]) and the *interrupt_uart* interrupt to wake the ARM platform from STOP mode. Re-enable UART clocks and clear the AWAKE bit by writing 1 to it. Writing 0 to the AWAKE bit has no effect. When IR interface is enabled (UCR1[7]=1), the AWAKE bit is always not asserted. The generation of the asynchronous AWAKE interrupt does not need any clocks.

In IR mode, if the asynchronous IR WAKE interrupt is enabled (AIRINTEN = UCR3[5] = 1), and if the ARM platform is in STOP mode (UART clocks are off when ARM platform in STOP mode), then the detection of a falling edge on the receive pin (RXD_IR), asserts the AIRINT bit (USR1[5]), and the *interrupt_uart* interrupt. This interrupt wakes the ARM platform from STOP mode. Software re-enables UART clocks and clear the AIRINT bit by writing 1 to it. Writing 0 to the AIRINT bit has no effect. When IR interface is disabled (UCR1[7]=0), the AIRINT bit is always not asserted. The generation of the asynchronous AIRINT interrupt does not need any clocks.

Recommended procedure for programming the asynchronous interrupts is to first clear them by writing 1 to the appropriate bit in the UART Status Register 1 (USR1). Poll or enable the interrupt for the Receiver IDLE Interrupt Flag (RXDS) in the USR1. When asserted, the RXDS bit indicates to the software that the receiver state machine is in the idle state, the next state is idle, and the RXD pin is idle (high). After following this procedure, enable the asynchronous interrupt and enter STOP mode.

55.3.5.4 Receiving a BREAK Condition

A BREAK condition is received when the receiver detects all 0s (including a 0 during the bit time of the stop bit) in a frame. The BREAK condition asserts the BRCD bit (USR2[2]) and writes only the first BREAK character to the RxFIFO. Clear the BRCD bit by writing 1 to it. Writing 0 to the BRCD bit has no effect.

Asserting BRCD would generate an interrupt on *interrupt_uart*. The interrupt generation can be masked using the control bit BKEN (UCR4[2]). Receiving a break condition will also effect the following bits in the receiver register URXD:

URXD(11) = BRK. While high this bit indicates that the current char was detected as a break.

URXD(12) = FRMERR. The frame error bit will always be set when BRK is set.

URXD(10) = PRERR. If odd parity was selected the parity error bit will also be set when BRK is set.

URXD(14) = ERR. The error detect bit indicates that the character present in the rx data field has an error status. This can be asserted by a break.

55.3.5.5 Vote Logic

The vote logic block provides jitter tolerance and noise immunity by sampling with respect to a 16x clock (*brm_clk*) and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of the *brm_clk*.

See Figure 55-5. The receiver is provided with the majority vote value, which is 2 out of the 3 samples. For examples of the majority vote results of the vote logic, see the following table.

Table 55-7. Majority Vote Results

Samples	Vote
000	0
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of *brm_clk*, however the receiver uses 16x oversampling to take its value in the middle of the sample character.

The receiver starts to count when the Start bit is set however it does not capture the contents of the RxFIFO at the time the Start bit is set. The start bit is validated when 0s are received for 7 consecutive 1/16 of bit times following the 1-to-0 transition. Once the counter reaches 0xF, it starts counting on the next bit and captures it in the middle of the sampling frame (see Table 55-7). All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register (SIPO_OUT) data is parallel shifted to the RxFIFO.

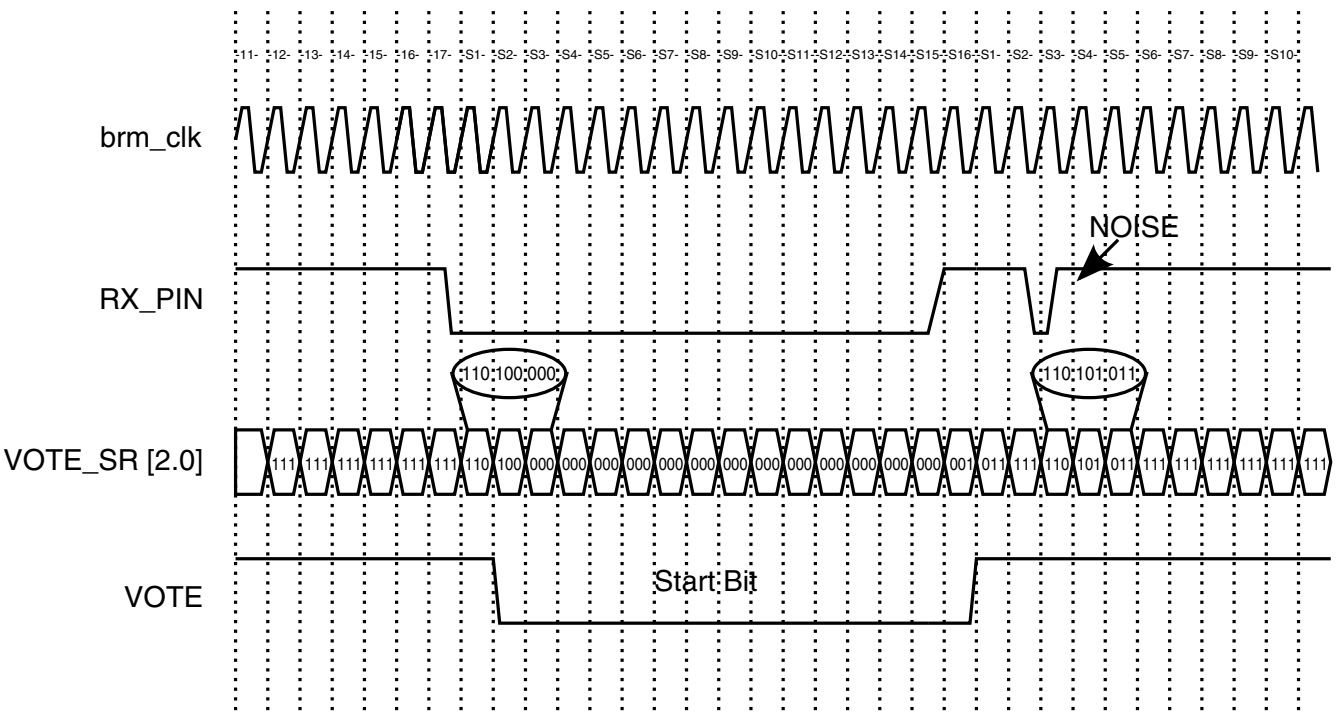


Figure 55-5. Majority Vote Results

A new feature has been recently implemented, it allows to re-synchronize the counter on each edge of RXD line. This is automatic and allows to improve the immunity of UART against signal distortion.

There is a special case when the *brm_clk* frequency is too low and is unable to capture a 0 pulse in IrDA. In this case, the software must set the IRSC (UCR4[5]) bit so that the reference clock (after internal divider) is used for the voting logic. The pulse is validated by counting the length of the pulse.

Refer to Infrared Interface for more details.

55.3.5.6 Baud Rate Automatic Detection Logic

When the baud rate automatic detection logic is enabled, the UART locks onto the incoming baud rate. To enable this feature, set the automatic detection of baud rate bit (ADBR = UCR1[14] = 1) and write 1 to the ADET bit (USR2[15]) to clear it.

When ADET=0 and ADBR =1, the detection starts. Then, once the beginning of start bit (transition from 1-to-0 of RXD) has been detected, UART start a counter (UBRC) working at reference frequency. Once the end of start bit is detected (transition from 0-to-1 of RXD), the value of UBRC - 1 is directly copied into UBMR register. UBIR register is filled with 0x000F.

So, at the end of start bit, registers gets following values:

```
UBRC = number of reference clock periods (after divider) during Start bit.
UBIR = 0x000F
UBMR = UBRC - 1
```

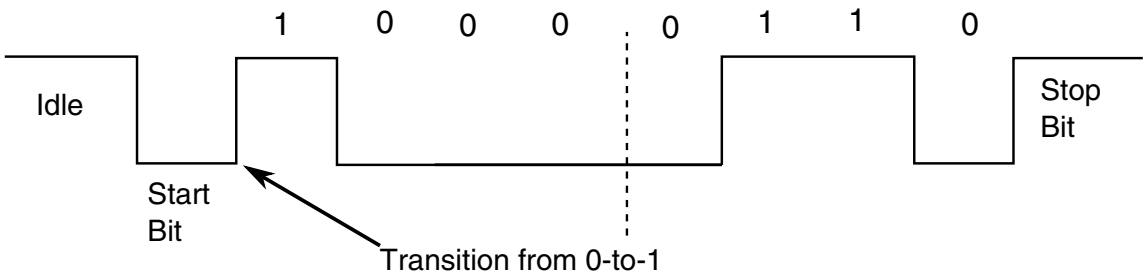
The updated values of the 3 registers can be read.

See Table 55-8 for list of parameters for baud rate detection and Figure 55-6 for baud rate detection protocol diagram.

If any of the UART BRM registers are simultaneously written by the baud rate automatic detection logic and by the peripheral data bus, the peripheral data bus would have lower priority.

Table 55-8. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	*interrupt_uart*
0	X	Manual Configuration	1
1	0	Auto Detection Started	1
1	1	Auto Detection Complete	0
NOTE: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal.			



Note: LSB Transmitted first.

Figure 55-6. Baud Rate Detection Protocol Diagram

55.3.5.6.1 Baud Rate Automatic Detection Protocol

The receiver must receive an ASCII character "A" or "a" to verify proper detection of the incoming baud rate. When an ASCII character "A" (0x41) or "a" (0x61) is received and no error occurs, the Automatic Detect baud rate bit is set (ADET=1) and if the interrupt is enabled (ADEN=UCR1[15]=1), an interrupt *interrupt_uart* is generated.

When an ASCII character "A" or "a" is not received (because of a bit error or the reception of another character), the auto detection sequence restarts and waits for another 1-to-0 transition.

As long as ADET = 0 and ADBR = 1, the UART continues to try to lock onto the incoming baud rate. Once the ASCII character "A" or "a" is detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated baud rate.

The UART interrupt is active (*interrupt_uart* = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the automatic baud rate detection interrupt enable bit (ADEN = 0). Before starting an automatic baud rate detection sequence, set ADET = 0 and ADBR = 1.

The RxFIFO must contain the ASCII character "A" or "a" following the automatic baud rate detection interrupt.

The 16-bit UART Baud Rate Count Register (UBRC) is reset to 4 and stays at 0xFFFF when an overflow occurs. The UBRC register counts (measures) the duration of start bit. When the start bit is detected and counted, the UART Baud Rate Count Register retains its value until the next automatic baud rate detection sequence is initiated.

The Baud Rate Count Register counts only when auto detection is enabled.

55.3.5.6.2 New Baud Rate Determination

In order to fight against the problems caused by the distortion and the noise on the RXD line, the duration of the baud rate measurement has been extended.

Previously, as described above, this determination was based on the measurement of the START bit duration. Now, this measurement is based on the duration of START bit + bit0. Bit0 is the first bit following the START bit. In fact, the counter which is started at the falling edge of START bit is no longer stopped at next rising edge (end of START bit), but it is stopped at the next falling edge (end of bit0). As the character sent is always a "A" (41h) or a "a" (61h), this second falling edge will always be present and it will indicate the end of bit0. Once this counter is stopped, the result is divided by 2 and used by the BRM to determine the incoming baud rate.

NOTE

UBRC register contains the result of this division by two, in consequence it reflects the measurement of the duration of one bit.

55.3.5.6.2.1 New Autobaud Counter Stopped bit and Interrupt

A new bit has been added in USR2 register: ACST (USR2[11]). This bit is set immediately after the determination of the baud rate.

So,

- if ADNIMP is not set (default), ACST is set to 1 after the end of bit0,
- If ADNIMP is set to 1, ACST is set to 1 at the end of START bit.

If ACIEN (UCR3[0]) is set to 1, ACST will flag an interrupt on *interrupt_uart* signal. This interrupt informs the ARM platform that the BRM has just been set with the result of the bit length measurement. If needed, the ARM platform can perform a read of UBMR (or UBRC) register and determine by itself the baud rate measured. Then the ARM platform has the possibility to correct the BRM registers with the nearest standardized baud rate.

NOTE

ACST is set only if ADBR is set to 1, for example, the UART is autobauding.

Clear the ACST bit by writing 1 to it. Writing 0 to the ACST bit has no effect.

55.3.6 Escape Sequence Detection

An escape sequence typically consists of 3 characters entered in rapid succession (such as +++). Because these are valid characters by themselves, the time between characters determines if it is a valid escape sequence. Too much time between two of the "+" characters is interpreted as two "+" characters, and not part of an escape sequence.

The software chooses the escape character and writes its value to the UART Escape Character Register (UESC). The software must also enable escape detection feature by setting ESCEN (UCR2[11]) to 1. The hardware compares this value to incoming characters in the RxFIFO. When an escape character is detected, the internal escape timer starts to count. The software specifies a time-out value for the maximum allowable time between 2 successive escape characters (see the table below). The escape timer is programmable in intervals of 2 ms to a maximum interval of 8.192 seconds.

Table 55-9. Escape Timer Scaling

UTIM Register	Maximum Time Between Specified Escape Characters
0x000	2 ms
0x001	4 ms
0x002	6 ms
0x003	8 ms
0x004	10 ms
...	...
0F8	498 ms
0F9	500 ms
...	...
9C3	5 s
...	...
FFD	8.188 s
FFE	8.190 s
FFF	8.192 s
NOTE: To calculate the time interval: (UTIM_Value + 1) x 0.002 = Time_Interval Example: (09C3 + 1) x 0.002 = 5 s.	

The escape sequence detection feature is available for all the reference frequencies. Before using Escape Sequence Detection, the user must fill the ONEMS register. This 24-bit register must contain the value of the UART internal frequency divided by 1000. The internal frequency is obtained after the UART internal divider which is applied on *module_clock* clock.

Example I:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 2 with the internal divider: UFCR[9:7] = 3'b100

$$ONEMS = \frac{66.5 \times 10^6}{2 \times 1000} = 33250 = 81E2h$$

Figure 55-7. Calculation of Frequency for ONEMS Register

Example II:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 1with the internal divider: UFCR[9:7] = 3'b101

$$ONEMS = \frac{66.5 \times 10^6}{1000} = 66500 = 103C4h$$

Figure 55-8. Calculation of Frequency for ONEMS Register

The escape sequence detection interrupt is asserted when the escape sequence interrupt enable (ESCI) bit is set and an escape sequence is detected (ESCF set). Clear the ESCF bit by writing 1 to it. Writing 0 to the ESCF bit has no effect.

55.4 Binary Rate Multiplier (BRM)

The BRM sub-block receives *ref_clk* (*module_clock* clock after divider). From this clock, and with integer and non-integer division, BRM generates a 16x baud rate clock whose frequency is 16 times of baud rate.

The uart transmitter will shift data out based on this 16x baud rate clock. The uart receiver will sample the serial data line based on this 16x baud rate clock. The input and output frequency ratio is programmed in the UART BRM Incremental Register (UBIR) and UART BRM MOD Register (UBMR). The output frequency is divided by the input frequency to produce this ratio. For integer division, set the UBIR = 0x000F and write the divisor to the UBMR register. All values written to these registers must be one less than the actual value to eliminate division by 0 (undefined), and to increase the maximum range of the registers.

Updating the BRM registers requires writing to both registers. The UBIR register must be written before writing to the UBMR register. If only one register is written to by the software, the BRM continues to use the previous values.

The following examples show how to determine what values are to be programmed into UBIR and UBMR for a given reference frequency and desired baud rate. The following equation can be used to help determine these values:

$$\text{BaudRate} = \frac{\text{Ref Freq}}{\left(16 \times \frac{\text{UBMR} + 1}{\text{UBIR} + 1} \right)}$$

Figure 55-9. Frequency and Baud Rate for UBIR and UBMR

With:

Reference Frequency (Hz): UART Reference Frequency (*module_clock* after RFDIV divider)

Baud Rate (bit/s): Desired baud rate.

Integer Division ÷ 21

Reference Frequency = 19.44 MHz

UBIR = 0x000F

UBMR = 0x0014

Baud Rate = 925.7 kbit/s

NOTE

Observe that each value written to the registers is one less than the actual value.

Non-Integer Division

```
Reference Frequency = 16 MHz
Desired Baud Rate = 920 Kbits/s
```

$$\frac{UBMR + 1}{UBIR + 1} = \frac{RefFreq}{16 \times BaudRate} = \frac{16 \times 10^6}{16 \times 920 \times 10^3} = 1.087$$

```
Ratio = 1.087 = 1087 / 1000
UBIR = 999 (decimal)= 0x3E7
UBMR = 1086 (decimal)= 0x43E
Non-Integer Division
Reference Frequency = 25 MHz
Desired Baud Rate = 920 kbit/s
Ratio = 1.69837 = 625 / 368
UBIR = 367 (decimal)= 0x16F
UBMR = 624 (decimal)= 0x270
```

Non-Integer Division

```
Reference Frequency: 30 MHz
Desired Baud Rate = 115.2 kbit/s
Ratio = 16.276043 = 65153 / 4003
UBIR = 4002 (decimal) = 0x0FA2
UBMR = 65152 (decimal) = 0xFE80
```

55.5 Infrared Interface

55.5.1 Generalities-Infrared

The Infrared interface is selected when IREN (UCR1[7]) is set to 1.

The Infrared Interface is compatible with IrDA Serial Infrared Physical Layer Specification. In this specification, a "zero" is represented by a positive pulse, and a "one" is represented by no pulse (line remains low).

In the UART:

In TX: For each "zero" to be transmitted, a narrow positive pulse which is 3/16 of a bit time is generated. For each "one" to be transmitted no pulse is generated (output is low). External circuitry has to be provided to drive an Infrared LED.

In RX: When receiving, a narrow negative pulse is expected for each "zero" transmitted while no pulse is expected for each "one" transmitted (input is high).

NOTE

Rx part of IR block expects to receive an inverted signal compared to IrDA specification. Circuitry external to the IC transforms the Infrared signal to an electrical signal.

The IR interface has an edge triggered interrupt (IRINT). This interrupt validates a zero bit being received. This interrupt is enabled by writing a "one" to ENIRI bit.

The behavior of Infrared Interface is determined by 3 bits INVT (UCR3[1]), INVR (UCR4[9]) and IRSC (UCR4[5]).

55.5.2 Inverted Transmission and Reception bits (INVT & INVR)

The values of INVT and INVR depend of the IrDA transceiver connected on the TXD_IR and RXD_IR pins of the UART. If this transceiver is not inverting on both paths Tx and Rx, a Zero is represented by a positive pulse and a One is represented by no pulse (line remains low). In this case, the bit INVT must be set to 0 and the bit INVR must be set to 1 (because Rx IR block expects an inverted signal).

On the contrary user must set INVT=1 and INVR=0 if both paths of the transceiver are inverting, that is, a Zero is represented as a negative pulse and a One is represented by no pulse (line remains high). The transceiver can also be inverting on only one path (Tx or Rx), in this case INVT and INVR must be together equal to 1 or to 0, depending on which path is inverted.

55.5.3 InfraRed Special Case (IRSC) Bit

The value to apply to IRSC bit is based on 2 parameters: the baud rate and the Minimum Pulse Duration (MPD) of the transceiver.

According to IrDA Standard Specification, for SIR (Serial IR) baud rates from 2.4 Kbit/s to 115.2 Kbit/s this nominal pulse duration is equal to 3/16 of a bit duration (at the selected baud rate). But, for all the baud rates a Minimum Pulse Duration is also specified. According to IrDA Standard, a Zero is represented by a light pulse, so the IrDA transceiver can't emit a light pulse shorter than the MPD. For SIR, the MPD is constant and equal to 1.41 us.

But user must take into account the electrical MPD associated to the transceiver on the receiver path. Typically this value is 2.0 us, but for some manufacturers MPD can go down to 1.0 us.

In order to understand the meaning of IRSC bit, one must understand how the RX path work in IrDA mode.

When UART is in IrDA mode, a Zero is not only detected by the state of the RXD_IR line, but also with the duration of the pulse. This pulse duration can be measured with 2 different clocks. In this case, clock is selected with the IRSC bit.

- If IRSC = 0, the clock used is the BRM clock.
- If IRSC = 1, the clock used is the UART internal clock (UART clock after the divider (RFDIV)).

In normal operation, IRSC=0. This means at any time, the user must ensure that the frequency of BRM_clock is high enough to measure the pulse. In the UART and for IRSC=0, the pulse must last at least 2 BRM clock cycles.

If this condition is not fulfilled, IRSC must be set to 1.

Let's take 2 examples, with the Minimum Pulse Duration equals to the MPD of the IrDA specification (in SIR).

Calculation of BRM Clock Period (Clock Period < 1.41 µs)

The user wants to receive IrDA data at 115.2 Kbit/s. The UBIR and UBMR registers are set in order to create the BRM_clock with a frequency of 16*baud rate = 16 * 115.2 = 1.843 MHz. But at the same time, in order to correctly detect the pulse, the user must be sure that 2* BRM_clock period is lower than 1.41 us. Lets check:

BRM_clock period = 1/1843000 = 542 ns

So 2*BRM_clock period = 1.09 us < 1.41 us. It is fine.

Calculation of BRM Clock Period (Clock Period > 1.41 µs)

This time the user wants to receive at 19.2 Kbit/s. So, the BRM_clock is set to 16*19200 = 307.2 kHz. Let's check if 2* BRM_clock period < 1.41 us:

1. BRM_clock period =1/307200 = 3.25 us

   So 2*BRM_clock period = 6.50 us >> 1.41 us. It doesn't work.

   So, in this case, the BRM clock can't be used to measure the pulse duration and the user must select the UART internal clock by setting IRSC =1.

NOTE

Like for Escape character detection, when IR Special Case is enabled (IRSC=1), the UART must measure a duration. In order to do that, the user must fill the ONEMS register. Refer to Escape Sequence Detection.

55.5.4 IrDA interrupt

Serial infrared mode (SIR) uses an edge triggered interrupt flag IRINT (USR2[8]). When INVR =0, detection of a falling edge on the RXD pin asserts the IRINT bit. When INVR=1, detection of a rising edge on the RXD pin asserts the IRINT bit. When IRINT and ENIRI bits are both asserted, the *interrupt_uart* interrupt is asserted. Clear the IRINT bit by writing 1 to it. Writing 0 to the IRINT bit has no effect.

55.5.5 Conclusion about IrDA

Before using the UART in IrDA, the baud rate limit must be calculated. This baud rate limit will inform the user if IRSC bit has to be set or not.

Let's determine this limit:

As already described, if IRSC = 0, the following condition must always be fulfilled

$$2 \times BRMClockPeriod < MinPulseDuration$$

Figure 55-10. Calculation of Baud Rate

So,

$$BRMClockFrequency > \frac{2}{MPD}$$

So, knowing BRM_clock frequency = 16 * Baud Rate, we get:

$$BaudRate > \frac{1}{8 \times MinPulseDuration}$$

So, the user needs to set IRSC = 0 when:

- If Minimum Pulse Duration = 2.5 us and Baud Rate > 50 Kbit/s.
- If Minimum Pulse Duration = 2.0 us and Baud Rate > 62.5 Kbit/s.
- If Minimum Pulse Duration = 1.41 us and Baud Rate > 88.6 Kbit/s.

NOTE

For baud rates lower than the limit, IRSC must be set to1.

55.5.6 Programming IrDA Interface

55.5.6.1 High Speed

As an example, the following sequence can be used to program the IrDA interface in order to send and receive characters at 115.2 Kbit/s.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 115.2 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to ARM platform when 1 char is received into the Rx FIFO (RDR)

Registers values and Programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
UTS = 0x0000
UFCR = 0x0981
TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock  is 30 MHz)
RXTL[5:0] = 0x01: Default value
UBIR = 0x0202
UBMR = 0x20BE Baud rate = 115.2 kbit/s with internal clock = 30 MHz
UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset
UCR3 = 0x0000

         UCR4 = 0x8201
CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is ready to send a character as soon as there is a write into UTXD register. And an interrupt will be sent to ARM platform when a character is received.

55.5.6.2 Low Speed

This time, we keep the same assumptions but the speed is now 9.6 Kbit/s. So, this baud rate is below the limit (even with a Min. Pulse Duration of 2.5 us) and thus IRSC must be set to 1.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 9.6 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to ARM platform when 1 char is received into the Rx FIFO (RDR).

Registers values and Programming orders:

```
UCR1 = 0x0085
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
UFCR = 0x0981
UFCR[15:10] = TXTL[5:0] = 0x02: Default value
RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock  is 30 MHz)
UFCR[5:0] = RXTL[5:0] = 0x01: Default value
UBIR = 0x00FF
UBMR = 0xC354 Baud rate = 9.6 kbit/s with internal clock = 30 MHz
UCR2 = 0x4027
UCR2[14] = IRTS = 1: Ignore level of RTS input signal
UCR2[5] = WS = 1: Characters are 8-bit length
UCR2[2] = TXEN = 1: Enable Rx path
UCR2 [1] = RXEN = 1: Enable Tx path
UCR2[0] = SRST_B = 1: No software reset
UCR3 = 0x0000
UCR3[1] = INVT = 0: Positive pulse represents 0.
UCR4 = 0x8221
UCR4[15:10] = CTSTL[5:0] = 0x20: Default value
UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)
UCR4[5] = IRSC = 1: Because data rate is below the limit and thus the UART internal clock is
used to measure the pulse duration.
UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)
```

The UART is now ready to send a character as soon as there is a write into UTXD register. An interrupt will be sent to ARM platform when a character is received.

55.6 Low Power Modes

These modes are controlled by the signals *doze_req* and *stop_req*. The control/status/data registers won't change when getting into/out of low power modes.

Table 55-10. UART Low Power State Operation

	Normal State (*doze_req* = 1'b0 & *stop_req* = 1'b0)	Doze State (*doze_req* = 1'b1)		Stop State (*stop_req* = 1'b1)
		DOZE bit = 0	DOZE bit = 1	
UART-Clock	ON	ON	ON	OFF
UART Serial / IrDA	ON	ON	OFF	OFF

55.6.1 UART Operation in System Doze Mode

While in Doze State (when *doze_req* input pin is set to 1'b1), the UART behavior depends on the DOZE (UCR1[1]) control bit.

While the DOZE bit is negated, the UART serial interface is enabled. While the system is in the Doze State, and the DOZE bit is asserted, the UART is disabled. If the Doze State is entered with the DOZE bit asserted while the UART serial interface was receiving or transmitting data, it will complete the receive/transmit of the current character and signal to the far-end transmitter/receiver to stop sending/receiving.

55.6.2 UART Operation in System Stop Mode

The internal baud rate clocks of the transmitter and receiver are gated off if the *stop_req* signal to UART is asserted. Even though the clocks at the input of the UART continue to run during system Stop mode, the UART will not do any transmission or reception.

The following UART interrupts wake the ARM platform processor from STOP mode:

- RTS (RTSD)
- IrDA Asynchronous WAKE (AIRINT)
- Asynchronous WAKE (AWAKE)
- RI (RIDELT in DTE mode only)
- DCD (DCDDELT in DTE mode only)
- DTR (DTRD in DCE mode only)
- DSR (DTRD in DTE mode only)

When an asynchronous WAKE (awake) interrupt exits the ARM platform from STOP mode, make sure that a dummy character is sent first because the first character may not be received correctly.

55.6.3 Power Saving Method in UART

The RXEN (UCR2[1]), TXEN (UCR2[2]) and UARTEN (UCR1[0]) bits are set by the user and provide software control of low-power modes.

Setting the UARTEN (UCR1[0]) bit to 0 shuts off the receiver and transmitter logic and the associated clocks.

If the UART is used only in transmit mode, UARTEN and TXEN must be set to 1. If the UART is used only in receive mode, UARTEN and RXEN must be set to 1. Setting TXEN or RXEN to 0 allows to save a lot of power.

55.7 UART Operation in System Debug State

The bit UTS [11] controls whether the UART will respond to the input signal *debug_req*, or whether it will continue to run as normal.

If the UART is programmed to respond to *debug_req*:

1. The UART will halt all operations upon detecting the *debug_req* input.
2. A transfer in progress, either to/from a core (using the IP Bus interface) or to/from an external device, will be completed before halting. This means a single byte/word transfer, not an entire FIFO. Reception of any further data from an external device will be disabled.
3. Internal registers will continue to be writable and readable using the IP Bus interface. A read will leave the contents unaffected.
4. The RX FIFO is affected in debug mode in the following way:
   - All writes into the RX FIFO are prevented.
   - The bit RXDBG (UTS[9]) is used to select the readability of the RX FIFO during debug mode:

     RXDBG = 0: hold the read pointer at the location it had upon entering debug mode, and URXD register returns only the data value at that location, no matter how many reads attempted.

     RXDBG = 1, selectable at any time: Allow to read the characters received in Rx FIFO. It will not be possible to re-read previously read locations, nor will it be possible to readjust the read pointer to the value it had prior to entering debug mode.

55.8 Reset

This section describes how to reset the block and explains special requirements related to reset.

55.8.1 Hardware reset

All of registers, FIFOs, state machines and sequential elements can be reset to their initial values by hardware reset or power on reset.

55.8.2 Software reset

The status registers USR1 and USR2, BRM registers UBIR and UBMR, TxFIFO and RxFIFO, and transmitter and receiver state machines can be reset by software reset. Internal logic will remain the software reset signal at active for about 4 *module_clock* cycles.

Programmer can follow the following software reset sequence:

1. Clear the $\overline{\text{SRST}}$ bit (UCR2[0])
2. Wait for software reset complete: poll SOFTRST bit (UTS[0]) until it is 0.
3. Re-program baud rate registers: Re-write UBIR and UBMR.

55.9 Transfer Error

The UART can generate a transfer error on the peripheral bus in the following cases:

- Core is writing into a read-only register.
- Core is accessing (read or write) an unused location within the assigned address space reserved to UART.
- Core is writing into UTXD register with transmit interface disabled (TXEN=0 or UARTEN=0)
- Core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0)

55.10 Functional Timing

This section includes timing diagrams for functional signaling.

55.10.1 IrDA Mode

According to IrDA specification, the low speed (115.2Kbit/s and below) IR frame format is compatible with UART frame.

In this figure, an example data 0x65 is used.

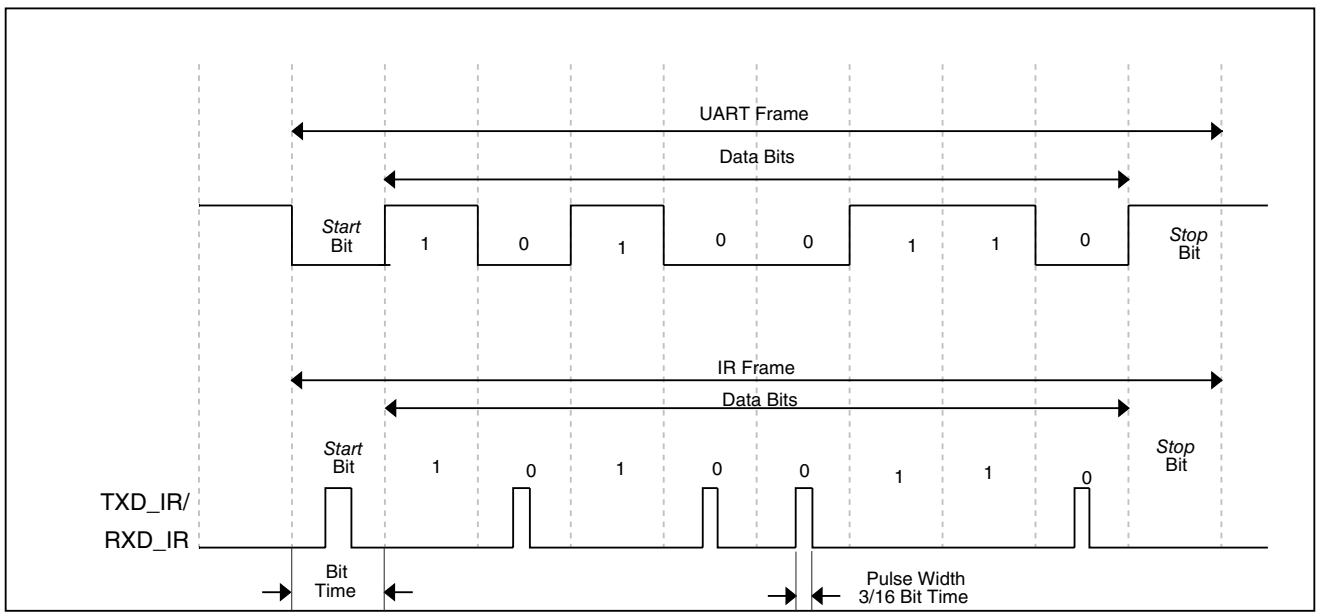


Figure 55-11. Timing diagram of Low Speed IR (<=115.2 Kbit/s) Data Line

55.11 Initialization

55.11.1 Programming the UART in RS-232 mode

As an example, the following sequence can be used to program the UART in order to send and receive characters in RS-232 mode.

Assumptions:

- Input uart clock = 100 MHz
- Baud rate = 921.6Kbps
- Data bits = 8 bits
- Parity = Even
- Stop bits = 1 bit
- Flow control = Hardware

Main program:

1. UCR1 = 0x0001

   Enable the UART.

2. UCR2 = 0x2127

   Set hardware flow control, data format and enable transmitter and receiver.

3. UCR3 = 0x0704

   Set UCR3[RXDMUXSEL] = 1.

4. UCR4 = 0x7C00

   Set CTS trigger level to 31,

5. UFCR = 0x089E

   Set internal clock divider = 5 (divide input uart clock by 5). So the reference clock is 100MHz/5 = 20MHz.

   Set TXTL = 2 and RXTL = 30.

6. UBIR = 0x08FF
7. UBMR = 0x0C34

   In the above two steps, set baud rate to 921.6Kbps based on the 20MHz reference clock.

8. UCR1 = 0x2201>

   Enable the TRDY and RRDY interrupts.

Interrupt service routine for the transmitter:

- Write characters into UTXD

The TRDY interrupt will be automatically de-asserted when the data level of the TxFIFO exceeds the TXTL=2. Note: For the first time the interrupt may be de-asserted after 4 characters are written into the TxFIFO because of the shift register.

Interrupt service routine for the receiver:

- Read characters from URXD

The RRDY interrupt will be automatically de-asserted when the data level of the RxFIFO is below the RXTL=30.

55.12 References

- EIA/TIA-232-F Interface Standard

  http://www.eia.org, http://www.tiaonline.org/standards

- IrDA Standard

  http://www.irda.org

55.13 Programmable Registers

UART supports 8-bit, 16-bit and 32-bit accesses to 32-bit memory-mapped addresses. Any access to unmapped memory location will yield a transfer error.

All registers except the ONEMS described in this section are 16-bit registers. The ONEMS register is a 24-bit register.

- For 32-bit write accesses, the upper two bytes will not be taken into account.
- For 32-bit read accesses the upper two bytes will return 0.

The ONEMS register is expanded from 16 bits to 24 bits in order to support the high frequency of the BRM internal clock *ref_clk* (*module_clock* after divider). The ONEMS register can be accessed as 8 bits, 16 bits or 32 bits.

- For 32-bit write accesses, the most significant byte of the ONEMS will be discarded.
- For 32-bit read accesses, the most significant byte of the ONEMS will be read as 0.

UART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5000_C000	UART Receiver Register (UART-3_URXD)	32	R	0000_0000h	55.13.1/ 3059
5000_C040	UART Transmitter Register (UART-3_UTXD)	32	W	0000_0000h	55.13.2/ 3061
5000_C080	UART Control Register 1 (UART-3_UCR1)	32	R/W	0000_0000h	55.13.3/ 3062
5000_C084	UART Control Register 2 (UART-3_UCR2)	32	R/W	0000_0001h	55.13.4/ 3064

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5000_C088	UART Control Register 3 (UART-3_UCR3)	32	R/W	0000_0700h	55.13.5/ 3067
5000_C08C	UART Control Register 4 (UART-3_UCR4)	32	R/W	0000_8000h	55.13.6/ 3069
5000_C090	UART FIFO Control Register (UART-3_UFCR)	32	R/W	0000_0801h	55.13.7/ 3071
5000_C094	UART Status Register 1 (UART-3_USR1)	32	R/W	0000_2040h	55.13.8/ 3073
5000_C098	UART Status Register 2 (UART-3_USR2)	32	R/W	0000_4028h	55.13.9/ 3075
5000_C09C	UART Escape Character Register (UART-3_UESC)	32	R/W	0000_002Bh	55.13.10/ 3078
5000_C0A0	UART Escape Timer Register (UART-3_UTIM)	32	R/W	0000_0000h	55.13.11/ 3078
5000_C0A4	UART BRM Incremental Register (UART-3_UBIR)	32	R/W	0000_0000h	55.13.12/ 3079
5000_C0A8	UART BRM Modulator Register (UART-3_UBMR)	32	R/W	0000_0000h	55.13.13/ 3079
5000_C0AC	UART Baud Rate Count Register (UART-3_UBRC)	32	R	0000_0004h	55.13.14/ 3080
5000_C0B0	UART One Millisecond Register (UART-3_ONEMS)	32	R/W	0000_0000h	55.13.15/ 3081
5000_C0B4	UART Test Register (UART-3_UTS)	32	R/W	0000_0060h	55.13.16/ 3082
53FB_C000	UART Receiver Register (UART-1_URXD)	32	R	0000_0000h	55.13.1/ 3059
53FB_C040	UART Transmitter Register (UART-1_UTXD)	32	W	0000_0000h	55.13.2/ 3061
53FB_C080	UART Control Register 1 (UART-1_UCR1)	32	R/W	0000_0000h	55.13.3/ 3062
53FB_C084	UART Control Register 2 (UART-1_UCR2)	32	R/W	0000_0001h	55.13.4/ 3064
53FB_C088	UART Control Register 3 (UART-1_UCR3)	32	R/W	0000_0700h	55.13.5/ 3067
53FB_C08C	UART Control Register 4 (UART-1_UCR4)	32	R/W	0000_8000h	55.13.6/ 3069
53FB_C090	UART FIFO Control Register (UART-1_UFCR)	32	R/W	0000_0801h	55.13.7/ 3071
53FB_C094	UART Status Register 1 (UART-1_USR1)	32	R/W	0000_2040h	55.13.8/ 3073

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FB_C098	UART Status Register 2 (UART-1_USR2)	32	R/W	0000_4028h	55.13.9/ 3075
53FB_C09C	UART Escape Character Register (UART-1_UESC)	32	R/W	0000_002Bh	55.13.10/ 3078
53FB_C0A0	UART Escape Timer Register (UART-1_UTIM)	32	R/W	0000_0000h	55.13.11/ 3078
53FB_C0A4	UART BRM Incremental Register (UART-1_UBIR)	32	R/W	0000_0000h	55.13.12/ 3079
53FB_C0A8	UART BRM Modulator Register (UART-1_UBMR)	32	R/W	0000_0000h	55.13.13/ 3079
53FB_C0AC	UART Baud Rate Count Register (UART-1_UBRC)	32	R	0000_0004h	55.13.14/ 3080
53FB_C0B0	UART One Millisecond Register (UART-1_ONEMS)	32	R/W	0000_0000h	55.13.15/ 3081
53FB_C0B4	UART Test Register (UART-1_UTS)	32	R/W	0000_0060h	55.13.16/ 3082
53FC_0000	UART Receiver Register (UART-2_URXD)	32	R	0000_0000h	55.13.1/ 3059
53FC_0040	UART Transmitter Register (UART-2_UTXD)	32	W	0000_0000h	55.13.2/ 3061
53FC_0080	UART Control Register 1 (UART-2_UCR1)	32	R/W	0000_0000h	55.13.3/ 3062
53FC_0084	UART Control Register 2 (UART-2_UCR2)	32	R/W	0000_0001h	55.13.4/ 3064
53FC_0088	UART Control Register 3 (UART-2_UCR3)	32	R/W	0000_0700h	55.13.5/ 3067
53FC_008C	UART Control Register 4 (UART-2_UCR4)	32	R/W	0000_8000h	55.13.6/ 3069
53FC_0090	UART FIFO Control Register (UART-2_UFCR)	32	R/W	0000_0801h	55.13.7/ 3071
53FC_0094	UART Status Register 1 (UART-2_USR1)	32	R/W	0000_2040h	55.13.8/ 3073
53FC_0098	UART Status Register 2 (UART-2_USR2)	32	R/W	0000_4028h	55.13.9/ 3075
53FC_009C	UART Escape Character Register (UART-2_UESC)	32	R/W	0000_002Bh	55.13.10/ 3078
53FC_00A0	UART Escape Timer Register (UART-2_UTIM)	32	R/W	0000_0000h	55.13.11/ 3078
53FC_00A4	UART BRM Incremental Register (UART-2_UBIR)	32	R/W	0000_0000h	55.13.12/ 3079

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FC_00A8	UART BRM Modulator Register (UART-2_UBMR)	32	R/W	0000_0000h	55.13.13/ 3079
53FC_00AC	UART Baud Rate Count Register (UART-2_UBRC)	32	R	0000_0004h	55.13.14/ 3080
53FC_00B0	UART One Millisecond Register (UART-2_ONEMS)	32	R/W	0000_0000h	55.13.15/ 3081
53FC_00B4	UART Test Register (UART-2_UTS)	32	R/W	0000_0060h	55.13.16/ 3082
53FF_C000	UART Receiver Register (UART-4_URXD)	32	R	0000_0000h	55.13.1/ 3059
53FF_C040	UART Transmitter Register (UART-4_UTXD)	32	W	0000_0000h	55.13.2/ 3061
53FF_C080	UART Control Register 1 (UART-4_UCR1)	32	R/W	0000_0000h	55.13.3/ 3062
53FF_C084	UART Control Register 2 (UART-4_UCR2)	32	R/W	0000_0001h	55.13.4/ 3064
53FF_C088	UART Control Register 3 (UART-4_UCR3)	32	R/W	0000_0700h	55.13.5/ 3067
53FF_C08C	UART Control Register 4 (UART-4_UCR4)	32	R/W	0000_8000h	55.13.6/ 3069
53FF_C090	UART FIFO Control Register (UART-4_UFCR)	32	R/W	0000_0801h	55.13.7/ 3071
53FF_C094	UART Status Register 1 (UART-4_USR1)	32	R/W	0000_2040h	55.13.8/ 3073
53FF_C098	UART Status Register 2 (UART-4_USR2)	32	R/W	0000_4028h	55.13.9/ 3075
53FF_C09C	UART Escape Character Register (UART-4_UESC)	32	R/W	0000_002Bh	55.13.10/ 3078
53FF_C0A0	UART Escape Timer Register (UART-4_UTIM)	32	R/W	0000_0000h	55.13.11/ 3078
53FF_C0A4	UART BRM Incremental Register (UART-4_UBIR)	32	R/W	0000_0000h	55.13.12/ 3079
53FF_C0A8	UART BRM Modulator Register (UART-4_UBMR)	32	R/W	0000_0000h	55.13.13/ 3079
53FF_C0AC	UART Baud Rate Count Register (UART-4_UBRC)	32	R	0000_0004h	55.13.14/ 3080
53FF_C0B0	UART One Millisecond Register (UART-4_ONEMS)	32	R/W	0000_0000h	55.13.15/ 3081
53FF_C0B4	UART Test Register (UART-4_UTS)	32	R/W	0000_0060h	55.13.16/ 3082

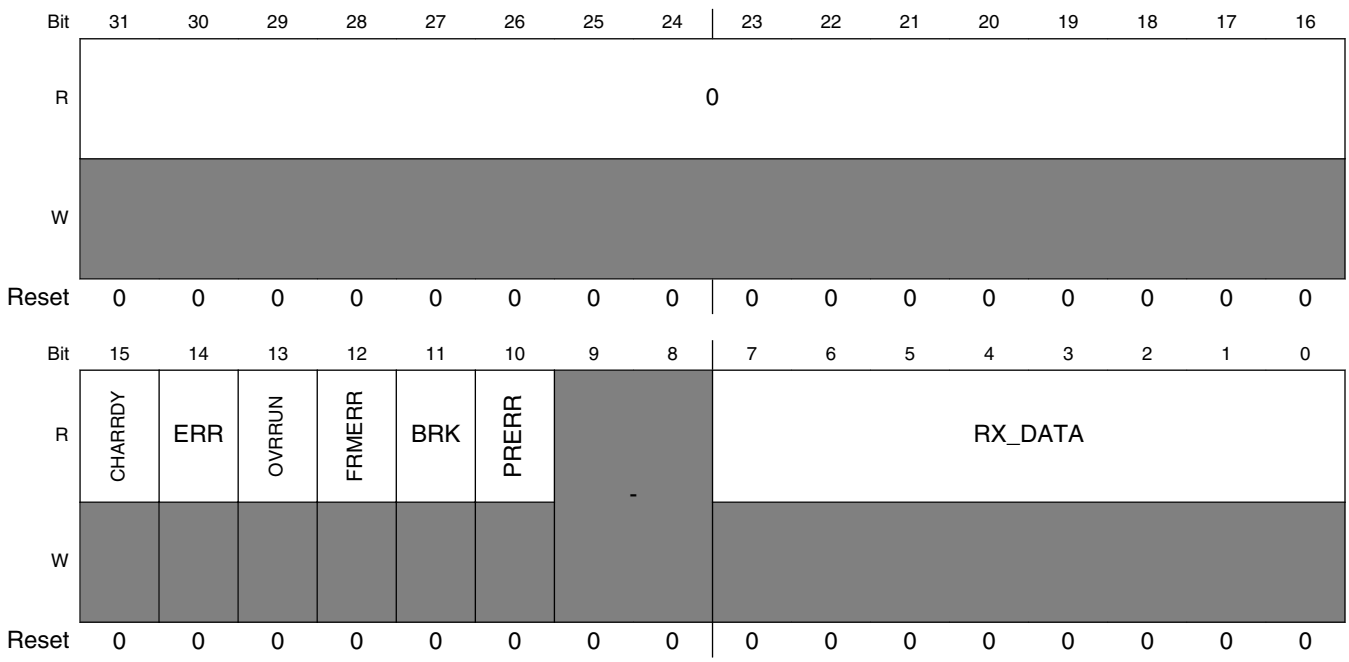
UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
63F9_0000	UART Receiver Register (UART-5_URXD)	32	R	0000_0000h	55.13.1/ 3059
63F9_0040	UART Transmitter Register (UART-5_UTXD)	32	W	0000_0000h	55.13.2/ 3061
63F9_0080	UART Control Register 1 (UART-5_UCR1)	32	R/W	0000_0000h	55.13.3/ 3062
63F9_0084	UART Control Register 2 (UART-5_UCR2)	32	R/W	0000_0001h	55.13.4/ 3064
63F9_0088	UART Control Register 3 (UART-5_UCR3)	32	R/W	0000_0700h	55.13.5/ 3067
63F9_008C	UART Control Register 4 (UART-5_UCR4)	32	R/W	0000_8000h	55.13.6/ 3069
63F9_0090	UART FIFO Control Register (UART-5_UFCR)	32	R/W	0000_0801h	55.13.7/ 3071
63F9_0094	UART Status Register 1 (UART-5_USR1)	32	R/W	0000_2040h	55.13.8/ 3073
63F9_0098	UART Status Register 2 (UART-5_USR2)	32	R/W	0000_4028h	55.13.9/ 3075
63F9_009C	UART Escape Character Register (UART-5_UESC)	32	R/W	0000_002Bh	55.13.10/ 3078
63F9_00A0	UART Escape Timer Register (UART-5_UTIM)	32	R/W	0000_0000h	55.13.11/ 3078
63F9_00A4	UART BRM Incremental Register (UART-5_UBIR)	32	R/W	0000_0000h	55.13.12/ 3079
63F9_00A8	UART BRM Modulator Register (UART-5_UBMR)	32	R/W	0000_0000h	55.13.13/ 3079
63F9_00AC	UART Baud Rate Count Register (UART-5_UBRC)	32	R	0000_0004h	55.13.14/ 3080
63F9_00B0	UART One Millisecond Register (UART-5_ONEMS)	32	R/W	0000_0000h	55.13.15/ 3081
63F9_00B4	UART Test Register (UART-5_UTS)	32	R/W	0000_0060h	55.13.16/ 3082

55.13.1 UART Receiver Register (UARTx_URXD)

The UART will yield a transfer error on the peripheral bus when core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0).

Addresses: UART-3_URXD is 5000_C000h base + 0h offset = 5000_C000h
UART-1_URXD is 53FB_C000h base + 0h offset = 53FB_C000h
UART-2_URXD is 53FC_0000h base + 0h offset = 53FC_0000h
UART-4_URXD is 53FF_C000h base + 0h offset = 53FF_C000h
UART-5_URXD is 63F9_0000h base + 0h offset = 63F9_0000h



UARTx_URXD field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 CHARRDY	Character Ready. This read-only bit indicates an invalid read when the FIFO becomes empty and software tries to read the same old data. This bit should not be used for polling for data written to the RX FIFO. 0 Character in RX_DATA field and associated flags are invalid. 1 Character in RX_DATA field and associated flags valid and ready for reading.
14 ERR	**Error Detect.** Indicates whether the character present in the RX_DATA field has an error (OVRRUN, FRMERR, BRK or PRERR) status. The ERR bit is updated and valid for each received character. 0 No error status was detected 1 An error status was detected

Table continues on the next page...

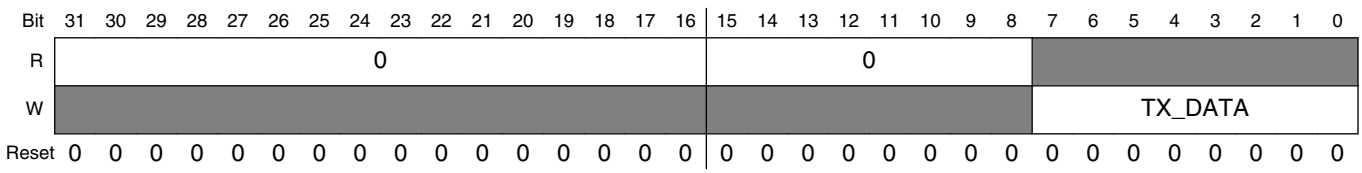
UARTx_URXD field descriptions (continued)

Field	Description
13 OVRRUN	**Receiver Overrun.** This read-only bit, when HIGH, indicates that the corresponding character was stored in the last position (32nd) of the Rx FIFO. Even if a 33rd character has not been detected, this bit will be set to '1' for the 32nd character. 0 No RxFIFO overrun was detected 1 A RxFIFO overrun was detected
12 FRMERR	**Frame Error.** Indicates whether the current character had a framing error (a missing stop bit) and is possibly corrupted. FRMERR is updated for each character read from the RxFIFO. 0 The current character has no framing error 1 The current character has a framing error
11 BRK	**BREAK Detect.** Indicates whether the current character was detected as a BREAK character. The data bits and the stop bit are all 0. The FRMERR bit is set when BRK is set. When odd parity is selected, PRERR is also set when BRK is set. BRK is valid for each character read from the RxFIFO. 0 The current character is not a BREAK character 1 The current character is a BREAK character
10 PRERR	**Parity Error flag**. Indicates whether the current character was detected with a parity error and is possibly corrupted. PRERR is updated for each character read from the RxFIFO. When parity is disabled, PRERR always reads as 0. 0 = No parity error was detected for data in the RX_DATA field 1 = A parity error was detected for data in the RX_DATA field
9–8 -	**Reserved**
7–0 RX_DATA	**Received Data**. Holds the received character. In 7-bit mode, the most significant bit (MSB) is forced to 0. In 8-bit mode, all bits are active.

55.13.2 UART Transmitter Register (UARTx_UTXD)

The UART will yield a transfer error on the peripheral bus when core is writing into UART_URXD register with transmit interface disabled (TXEN=0 or UARTEN=0). Memory space between UART_URXD and UART_UTXD registers is reserved. Any read or write access to this space will be considered as an invalid access and yield a transfer error.

Addresses: UART-3_UTXD is 5000_C000h base + 40h offset = 5000_C040h
UART-1_UTXD is 53FB_C000h base + 40h offset = 53FB_C040h
UART-2_UTXD is 53FC_0000h base + 40h offset = 53FC_0040h
UART-4_UTXD is 53FF_C000h base + 40h offset = 53FF_C040h
UART-5_UTXD is 63F9_0000h base + 40h offset = 63F9_0040h



UARTx_UTXD field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–0 TX_DATA	**Transmit Data**. Holds the parallel transmit data inputs. In 7-bit mode, D7 is ignored. In 8-bit mode, all bits are used. Data is transmitted least significant bit (LSB) first. A new character is transmitted when the TX_DATA field is written. The TX_DATA field must be written only when the TRDY bit is high to ensure that corrupted data is not sent.

55.13.3 UART Control Register 1 (UARTx_UCR1)

Addresses: UART-3_UCR1 is 5000_C000h base + 80h offset = 5000_C080h

UART-1_UCR1 is 53FB_C000h base + 80h offset = 53FB_C080h

UART-2_UCR1 is 53FC_0000h base + 80h offset = 53FC_0080h

UART-4_UCR1 is 53FF_C000h base + 80h offset = 53FF_C080h

UART-5_UCR1 is 63F9_0000h base + 80h offset = 63F9_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADEN	ADBR	TRDYEN	IDEN	ICD		RRDYEN	RXDMAEN	IREN	TXMPTYEN	RTSDEN	SNDBRK	TXDMAEN	ATDMAEN	DOZE	UARTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UCR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 ADEN	**Automatic Baud Rate Detection Interrupt Enable**. Enables/Disables the automatic baud rate detect complete (ADET) bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the automatic baud rate detection interrupt 1 Enable the automatic baud rate detection interrupt
14 ADBR	**Automatic Detection of Baud Rate**. Enables/Disables automatic baud rate detection. When the ADBR bit is set and the ADET bit is cleared, the receiver detects the incoming baud rate automatically. The ADET flag is set when the receiver verifies that the incoming baud rate is detected properly by detecting an ASCII character "A" or "a" (0x61 or 0x41). 0 Disable automatic detection of baud rate 1 Enable automatic detection of baud rate
13 TRDYEN	**Transmitter Ready Interrupt Enable**. Enables/Disables the transmitter Ready Interrupt (TRDY) when the transmitter has one or more slots available in the TxFIFO. The fill level in the TXFIFO at which an interrupt is generated is controlled by TxTL bits. When TRDYEN is negated, the transmitter ready interrupt is disabled. **NOTE:** An interrupt will be issued as long as TRDYEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TRDY interrupt. 0 Disable the transmitter ready interrupt 1 Enable the transmitter ready interrupt

Table continues on the next page...

UARTx_UCR1 field descriptions (continued)

Field	Description
12 IDEN	**Idle Condition Detected Interrupt Enable**. Enables/Disables the IDLE bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the IDLE interrupt 1 Enable the IDLE interrupt
11–10 ICD	**Idle Condition Detect**. Controls the number of frames RXD is allowed to be idle before an idle condition is reported. 00 Idle for more than 4 frames 01 Idle for more than 8 frames 10 Idle for more than 16 frames 11 Idle for more than 32 frames
9 RRDYEN	**Receiver Ready Interrupt Enable**. Enables/Disables the RRDY interrupt when the RxFIFO contains data. The fill level in the RxFIFO at which an interrupt is generated is controlled by the RXTL bits. When RRDYEN is negated, the receiver ready interrupt is disabled. 0 Disables the RRDY interrupt 1 Enables the RRDY interrupt
8 RXDMAEN	**Receive Ready DMA Enable**. Enables/Disables the receive DMA request *dma_req_rx* when the receiver has data in the RxFIFO. The fill level in the RxFIFO at which a DMA request is generated is controlled by the RXTL bits. When negated, the receive DMA request is disabled. 0 Disable DMA request 1 Enable DMA request
7 IREN	**Infrared Interface Enable**. Enables/Disables the IR interface. See the IR interface description in Infrared Interface, for more information. 0 Disable the IR interface 1 Enable the IR interface
6 TXMPTYEN	**Transmitter Empty Interrupt Enable**. Enables/Disables the transmitter FIFO empty (TXFE) interrupt. *interrupt_uart*. When negated, the TXFE interrupt is disabled. **NOTE:** An interrupt will be issued as long as TXMPTYEN and TXFE are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXFE interrupt. 0 Disable the transmitter FIFO empty interrupt 1 Enable the transmitter FIFO empty interrupt
5 RTSDEN	**RTS Delta Interrupt Enable**. Enables/Disables the RTSD interrupt. The current status of the $\overline{\text{RTS}}$ pin is read in the RTSS bit. 0 Disable RTSD interrupt 1 Enable RTSD interrupt
4 SNDBRK	**Send BREAK**. Forces the transmitter to send a BREAK character. The transmitter finishes sending the character in progress (if any) and sends BREAK characters until SNDBRK is reset. Because the transmitter samples SNDBRK after every bit is transmitted, it is important that SNDBRK is asserted high for a sufficient period of time to generate a valid BREAK. After the BREAK transmission completes, the UART transmits 2 mark bits. The user can continue to fill the TxFIFO and any characters remaining are transmitted when the BREAK is terminated.

Table continues on the next page...

UARTx_UCR1 field descriptions (continued)

Field	Description
	0 Do not send a BREAK character 1 Send a BREAK character (continuous 0s)
3 TXDMAEN	**Transmitter Ready DMA Enable**. Enables/Disables the transmit DMA request *dma_req_tx* when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO that generates the *dma_req_tx* is controlled by the TXTL bits. **NOTE:** A DMA request will be issued as long as TXDMAEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the transmit DMA request. 0 Disable transmit DMA request 1 Enable transmit DMA request
2 ATDMAEN	**Aging DMA Timer Enable**. Enables/Disables the receive DMA request *dma_req_rx* for the aging timer interrupt (triggered with AGTIM flag in USR1[8]). 0 Disable AGTIM DMA request 1 Enable AGTIM DMA request
1 DOZE	**DOZE**. Determines the UART enable condition in the DOZE state. When *doze_req* input pin is at '1', (the ARM Platform executes a doze instruction and the system is placed in the Doze State), the DOZE bit affects operation of the UART. While in the Doze State, if this bit is asserted, the UART is disabled. See the description in Low Power Modes. 0 The UART is enabled when in DOZE state 1 The UART is disabled when in DOZE state
0 UARTEN	**UART Enable**. Enables/Disables the UART. If UARTEN is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to a logic 1. UARTEN must be set to 1 before any access to UTXD and URXD registers, otherwise a transfer error is returned. This bit can be set to 1 along with other bits in this register. There is no restriction to the sequence of programing this bit and other control registers. 0 Disable the UART 1 Enable the UART

55.13.4 UART Control Register 2 (UARTx_UCR2)

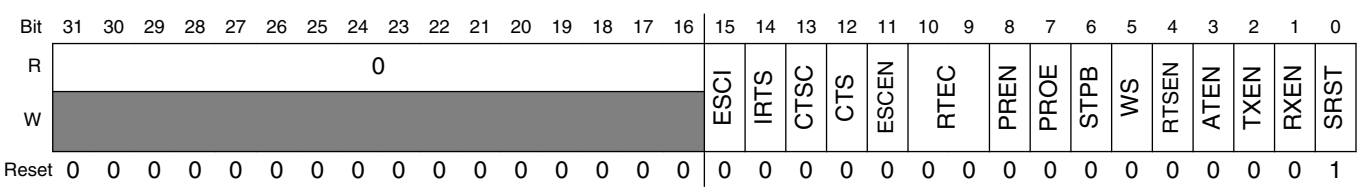
Addresses: UART-3_UCR2 is 5000_C000h base + 84h offset = 5000_C084h

UART-1_UCR2 is 53FB_C000h base + 84h offset = 53FB_C084h

UART-2_UCR2 is 53FC_0000h base + 84h offset = 53FC_0084h

UART-4_UCR2 is 53FF_C000h base + 84h offset = 53FF_C084h

UART-5_UCR2 is 63F9_0000h base + 84h offset = 63F9_0084h



UARTx_UCR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 ESCI	**Escape Sequence Interrupt Enable**. Enables/Disables the ESCF bit to generate an interrupt. 0 Disable the escape sequence interrupt 1 Enable the escape sequence interrupt
14 IRTS	**Ignore RTS Pin**. Forces the RTS input signal presented to the transmitter to always be asserted (set to low), effectively ignoring the external pin. When in this mode, the RTS pin serves as a general purpose input. 0 Transmit only when the RTS pin is asserted 1 Ignore the RTS pin
13 CTSC	CTS **Pin Control**. Controls the operation of the $\overline{CTS}$ output pin. When CTSC is asserted, the $\overline{CTS}$ output pin is controlled by the receiver. When the RxFIFO is filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) is validated, the $\overline{CTS}$ output pin is negated to indicate to the far-end transmitter to stop transmitting. When the trigger level is programmed for less than 32, the receiver continues to receive data until the RxFIFO is full. When the CTSC bit is negated, the $\overline{CTS}$ output pin is controlled by the CTS bit. On reset, because CTSC is cleared to 0, the $\overline{CTS}$ pin is controlled by the CTS bit, which again is cleared to 0 on reset. This means that on reset the $\overline{CTS}$ signal is negated. 0 The $\overline{CTS}$ pin is controlled by the CTS bit 1 The $\overline{CTS}$ pin is controlled by the receiver
12 CTS	**Clear to Send**. Controls the $\overline{CTS}$ pin when the CTSC bit is negated. CTS has no function when CTSC is asserted. 0 The $\overline{CTS}$ pin is high (inactive) 1 The $\overline{CTS}$ pin is low (active)
11 ESCEN	**Escape Enable**. Enables/Disables the escape sequence detection logic. 0 Disable escape sequence detection 1 Enable escape sequence detection
10–9 RTEC	**Request to Send Edge Control**. Selects the edge that triggers the RTS interrupt. This has no effect on the RTS delta interrupt. RTEC has an effect only when RTSEN = 1 (see Table 55-4). 00 Trigger interrupt on a rising edge 01 Trigger interrupt on a falling edge 1X Trigger interrupt on any edge
8 PREN	**Parity Enable**. Enables/Disables the parity generator in the transmitter and parity checker in the receiver. When PREN is asserted, the parity generator and checker are enabled, and disabled when PREN is negated. 0 Disable parity generator and checker 1 Enable parity generator and checker
7 PROE	**Parity Odd/Even**. Controls the sense of the parity generator and checker. When PROE is high, odd parity is generated and expected. When PROE is low, even parity is generated and expected. PROE has no function if PREN is low.

Table continues on the next page...

UARTx_UCR2 field descriptions (continued)

Field	Description
	0 Even parity 1 Odd parity
6 STPB	**Stop**. Controls the number of stop bits after a character. When STPB is low, 1 stop bit is sent. When STPB is high, 2 stop bits are sent. STPB also affects the receiver. 0 The transmitter sends 1 stop bit. The receiver expects 1 or more stop bits. 1 The transmitter sends 2 stop bits. The receiver expects 2 or more stop bits.
5 WS	**Word Size**. Controls the character length. When WS is high, the transmitter and receiver are in 8-bit mode. When WS is low, they are in 7-bit mode. The transmitter ignores bit 7 and the receiver sets bit 7 to 0. WS can be changed in-between transmission (reception) of characters, however not when a transmission (reception) is in progress, in which case the length of the current character being transmitted (received) is unpredictable. 0 7-bit transmit and receive character length (not including START, STOP or PARITY bits) 1 8-bit transmit and receive character length (not including START, STOP or PARITY bits)
4 RTSEN	**Request to Send Interrupt Enable.** Controls the RTS edge sensitive interrupt. When RTSEN is asserted and the programmed edge is detected on the $\overline{\text{RTS}}$ pin (the RTSF bit is asserted), an interrupt will be generated on the *interrupt_uart* pin. (See Table 55-4.) 0 Disable request to send interrupt 1 Enable request to send interrupt
3 ATEN	**Aging Timer Enable**. This bit is used to enable the aging timer interrupt (triggered with AGTIM) 0 AGTIM interrupt disabled 1 AGTIM interrupt enabled
2 TXEN	**Transmitter Enable**. Enables/Disables the transmitter. When TXEN is negated the transmitter is disabled and idle. When the UARTEN and TXEN bits are set the transmitter is enabled. If TXEN is negated in the middle of a transmission, the UART disables the transmitter immediately, and starts marking 1s. The transmitter FIFO cannot be written when this bit is cleared. 0 Disable the transmitter 1 Enable the transmitter
1 RXEN	**Receiver Enable**. Enables/Disables the receiver. When the receiver is enabled, if the RXD input is already low, the receiver does not recognize BREAK characters, because it requires a valid 1-to-0 transition before it can accept any character. 0 Disable the receiver 1 Enable the receiver
0 SRST	**Software Reset**. Once the software writes 0 to $\overline{\text{SRST}}$, the software reset remains active for 4 *module_clock* cycles before the hardware deasserts $\overline{\text{SRST}}$. The software can only write 0 to $\overline{\text{SRST}}$. Writing 1 to $\overline{\text{SRST}}$ is ignored. 0 Reset the transmit and receive state machines, all FIFOs and register USR1, USR2, UBIR, UBMR, UBRC , URXD, UTXD and UTS[6-3]. 1 No reset

55.13.5 UART Control Register 3 (UARTx_UCR3)

Addresses: UART-3_UCR3 is 5000_C000h base + 88h offset = 5000_C088h
UART-1_UCR3 is 53FB_C000h base + 88h offset = 53FB_C088h
UART-2_UCR3 is 53FC_0000h base + 88h offset = 53FC_0088h
UART-4_UCR3 is 53FF_C000h base + 88h offset = 53FF_C088h
UART-5_UCR3 is 63F9_0000h base + 88h offset = 63F9_0088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DPEC		DTREN	PARERREN	FRAERREN	DSR	DCD	RI	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	DTRDEN	RXDMUXSEL	INVT	ACIEN
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

UARTx_UCR3 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–14 DPEC	**DTR/DSR Interrupt Edge Control**. These bits control the edge of $\overline{\text{DTR}}$ (DCE) or $\overline{\text{DSR}}$ (DTE) on which an interrupt will be generated. An interrupt will only be generated if the DTREN bit is set. 00 interrupt generated on rising edge 01 interrupt generated on falling edge 1X interrupt generated on either edge
13 DTREN	**Data Terminal Ready Interrupt Enable**. When this bit is set, it will enable the status bit DTRF (USR2 [13]) (DTR/DSR edge sensitive interrupt) to cause an interrupt. 0 Data Terminal Ready Interrupt Disabled 1 Data Terminal Ready Interrupt Enabled
12 PARERREN	**Parity Error Interrupt Enable. Enables/Disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt.** 0 Disable the parity error interrupt 1 Enable the parity error interrupt
11 FRAERREN	**Frame Error Interrupt Enable. Enables/Disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt.** 0 Disable the frame error interrupt 1 Enable the frame error interrupt

Table continues on the next page...

UARTx_UCR3 field descriptions (continued)

Field	Description
10 DSR	**Data Set Ready**. This bit is used by software to control the DSR/DTR output pin for the modem interface. In DCE mode it applies to $\overline{DSR}$ and in DTE mode it applies to $\overline{DTR}$. 0 DSR/ DTR pin is logic zero 1 DSR/ DTR pin is logic one
9 DCD	**Data Carrier Detect**. In DCE mode this bit is used by software to control the $\overline{DCD}$ output pin for the modem interface. In DTE mode, when this bit is set, it will enable the status bit DCDDELT (USR2 (6)) to cause an interrupt. 0 $\overline{DCD}$ pin is logic zero (DCE mode) 1 $\overline{DCD}$ pin is logic one (DCE mode) 0 DCDDELT interrupt disabled (DTE mode) 1 DCDDELT interrupt enabled (DTE mode)
8 RI	**Ring Indicator**. In DCE mode this bit is used by software to control the $\overline{RI}$ output pin for the modem interface. In DTE mode, when this bit is set, it will enable the status bit RIDELT (USR2 (10)) to cause an interrupt. 0 $\overline{RI}$ pin is logic zero (DCE mode) 1 $\overline{RI}$ pin is logic one (DCE mode) 0 RIDELT interrupt disabled (DTE mode) 1 RIDELT interrupt enabled (DTE mode)
7 ADNIMP	**Autobaud Detection Not Improved-. Disables new features of autobaud detection (See Baud Rate Automatic Detection Protocol, for more details).** 0 Autobaud detection new features selected 1 Keep old autobaud detection mechanism
6 RXDSEN	Receive Status Interrupt Enable. Controls the receive status interrupt (*interrupt_uart*). When this bit is enabled and RXDS status bit is set, the interrupt *interrupt_uart* will be generated. 0 Disable the RXDS interrupt 1 Enable the RXDS interrupt
5 AIRINTEN	Asynchronous IR WAKE Interrupt Enable. Controls the asynchronous IR WAKE interrupt. An interrupt is generated when AIRINTEN is asserted and a pulse is detected on the RXD pin. 0 Disable the AIRINT interrupt 1 Enable the AIRINT interrupt
4 AWAKEN	Asynchronous WAKE Interrupt Enable. Controls the asynchronous WAKE interrupt. An interrupt is generated when AWAKEN is asserted and a falling edge is detected on the RXD pin. 0 Disable the AWAKE interrupt 1 Enable the AWAKE interrupt
3 DTRDEN	**Data Terminal Ready Delta Enable**. Enables / Disables the asynchronous DTRD interrupt. When DTRDEN is asserted and an edge (rising or falling) is detected on $\overline{DTR}$ (in DCE mode) or on $\overline{DSR}$ (in DTE mode), then an interrupt is generated. 0 Disable DTRD interrupt 1 Enable DTRD interrupt
2 RXDMUXSEL	RXD Muxed Input Selected. Selects proper input pins for serial and Infrared input signal.

Table continues on the next page...

UARTx_UCR3 field descriptions (continued)

Field	Description
	NOTE: In this chip, UARTs are used in MUXED mode, so that this bit should always be set.
1 INVT	In **IrDA mode**, when INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s. 0 TXDActive low transmission 1 **TXD** Active high transmission
0 ACIEN	**Autobaud Counter Interrupt Enable.** This bit is used to enable the autobaud counter stopped interrupt (triggered with ACST (USR2[11]). 0 ACST interrupt disabled 1 ACST interrupt enabled

55.13.6 UART Control Register 4 (UARTx_UCR4)

Addresses: UART-3_UCR4 is 5000_C000h base + 8Ch offset = 5000_C08Ch
UART-1_UCR4 is 53FB_C000h base + 8Ch offset = 53FB_C08Ch
UART-2_UCR4 is 53FC_0000h base + 8Ch offset = 53FC_008Ch
UART-4_UCR4 is 53FF_C000h base + 8Ch offset = 53FF_C08Ch
UART-5_UCR4 is 63F9_0000h base + 8Ch offset = 63F9_008Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CTSTL						INVR	ENIRI	WKEN	IDDMAEN	IRSC	LPBYP	TCEN	BKEN	OREN	DREN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UARTx_UCR4 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–10 CTSTL	**CTS Trigger Level**. Controls the threshold at which the $\overline{\text{CTS}}$ pin is deasserted by the RxFIFO. After the trigger level is reached and the $\overline{\text{CTS}}$ pin is deasserted, the RxFIFO continues to receive data until it is full. The CTSTL bits are encoded as shown in the Settings column. Settings 0 to 32 are in use. All other settings are Reserved.

Table continues on the next page...

UARTx_UCR4 field descriptions (continued)

Field	Description
	000000 0 characters received 000001 1 characters in the RxFIFO ... — ... — 100000 32 characters in the RxFIFO (maximum)
9 INVR	**In IrDA mode**,when cleared, the infrared logic block expects an active low or negative IR 3/16 pulse for 0s and 1s are expected for 1s. When INVR is set (INVR 1), the infrared logic block expects an active high or positive IR 3/16 pulse for 0s and 0s are expected for 1s. 0 **RXD** active low detection 1 **RXD** active high detection
8 ENIRI	**Serial Infrared Interrupt Enable**. Enables/Disables the serial infrared interrupt. 0 Serial infrared Interrupt disabled 1 Serial infrared Interrupt enabled
7 WKEN	**WAKE Interrupt Enable**. Enables/Disables the WAKE bit to generate an interrupt. The WAKE bit is set at the detection of a start bit by the receiver. 0 Disable the WAKE interrupt 1 Enable the WAKE interrupt
6 IDDMAEN	**DMA IDLE Condition Detected Interrupt Enable** Enables/Disables the receive DMA request *dma_req_rx* for the IDLE interrupt (triggered with IDLE flag in USR2[12]). 0 DMA IDLE interrupt disabled 1 DMA IDLE interrupt enabled
5 IRSC	**IR Special Case**. Selects the clock for the vote logic. When set, IRSC switches the vote logic clock from the sampling clock to the UART reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency. See InfraRed Special Case (IRSC) Bit. 0 The vote logic uses the sampling clock (16x baud rate) for normal operation 1 The vote logic uses the UART reference clock
4 LPBYP	**Low Power Bypass**. Allows to bypass the low power new features in UART. To use during debug phase. 0 Low power features enabled 1 Low power features disabled
3 TCEN	**TransmitComplete Interrupt Enable**. Enables/Disables the TXDC bit to generate an interrupt (*interrupt_uart* = 0) **NOTE:** An interrupt will be issued as long as TCEN and TXDC are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXDC interrupt. 0 Disable TXDC interrupt 1 Enable TXDC interrupt
2 BKEN	**BREAK Condition Detected Interrupt Enable**. Enables/Disables the BRCD bit to generate an interrupt. 0 Disable the BRCD interrupt 1 Enable the BRCD interrupt
1 OREN	**Receiver Overrun Interrupt Enable**. Enables/Disables the ORE bit to generate an interrupt.

Table continues on the next page...

UARTx_UCR4 field descriptions (continued)

Field	Description
	0 Disable ORE interrupt 1 Enable ORE interrupt
0 DREN	**Receive Data Ready Interrupt Enable**. Enables/Disables the RDR bit to generate an interrupt. 0 Disable RDR interrupt 1 Enable RDR interrupt

55.13.7 UART FIFO Control Register (UARTx_UFCR)

Addresses: UART-3_UFCR is 5000_C000h base + 90h offset = 5000_C090h

UART-1_UFCR is 53FB_C000h base + 90h offset = 53FB_C090h

UART-2_UFCR is 53FC_0000h base + 90h offset = 53FC_0090h

UART-4_UFCR is 53FF_C000h base + 90h offset = 53FF_C090h

UART-5_UFCR is 63F9_0000h base + 90h offset = 63F9_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXTL						RFDIV			DCEDTE	RXTL					
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

UARTx_UFCR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–10 TXTL	**Transmitter Trigger Level**. Controls the threshold at which a maskable interrupt is generated by the TxFIFO. A maskable interrupt is generated whenever the data level in the TxFIFO falls below the selected threshold. The bits are encoded as shown in the Settings column. Settings 0 to 32 are in use. All other settings are Reserved. 000000 Reserved 000001 Reserved 000010 TxFIFO has 2 or fewer characters ... — ... —

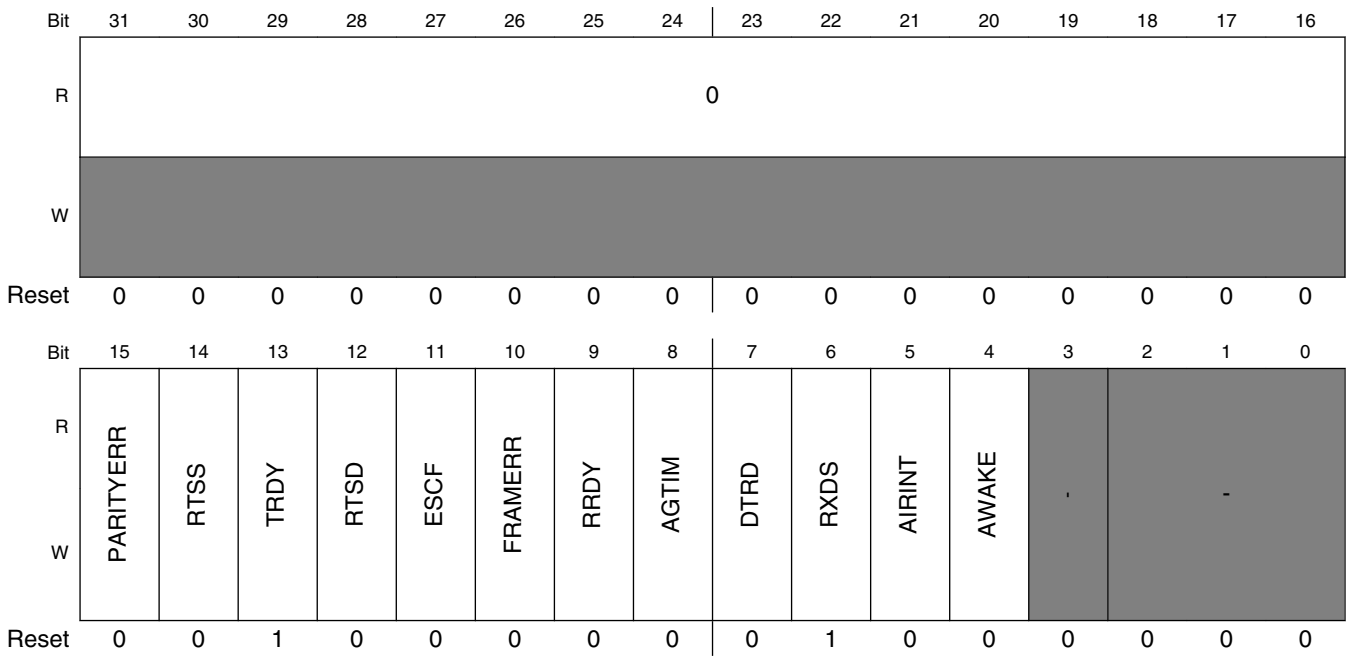
Table continues on the next page...

UARTx_UFCR field descriptions (continued)

Field	Description
	011111 TxFIFO has 31 or fewer characters 100000 TxFIFO has 32 characters (maximum)
9–7 RFDIV	Reference Frequency Divider. Controls the divide ratio for the reference clock. The input clock is *module_clock*. The output from the divider is *ref_clk* which is used by BRM to create the 16x baud rate oversampling clock (*brm_clk*). 000 Divide input clock by 6 001 Divide input clock by 5 010 Divide input clock by 4 011 Divide input clock by 3 100 Divide input clock by 2 101 Divide input clock by 1 110 Divide input clock by 7 111 Reserved
6 DCEDTE	**DCE/DTE mode select**. Select UART as data communication equipment (DCE mode) or as data terminal equipment (DTE mode). 0 DCE mode selected 1 DTE mode selected
5–0 RXTL	**Receiver Trigger Level**. Controls the threshold at which a maskable interrupt is generated by the RxFIFO. A maskable interrupt is generated whenever the data level in the RxFIFO reaches the selected threshold. The RXTL bits are encoded as shown in the Settings column. Setting 0 to 32 are in use. All other settings are Reserved. 000000 0 characters received 000001 RxFIFO has 1 character ... — ... — 011111 RxFIFO has 31 characters 100000 RxFIFO has 32 characters (maximum)

55.13.8 UART Status Register 1 (UARTx_USR1)

Addresses: UART-3_USR1 is 5000_C000h base + 94h offset = 5000_C094h
UART-1_USR1 is 53FB_C000h base + 94h offset = 53FB_C094h
UART-2_USR1 is 53FC_0000h base + 94h offset = 53FC_0094h
UART-4_USR1 is 53FF_C000h base + 94h offset = 53FF_C094h
UART-5_USR1 is 63F9_0000h base + 94h offset = 63F9_0094h



UARTx_USR1 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 PARITYERR	**Parity Error Interrupt Flag**. Indicates a parity error is detected. PARITYERR is cleared by writing 1 to it. Writing 0 to PARITYERR has no effect. When parity is disabled, PARITYERR always reads 0. At reset, PARITYERR is set to 0. 0 No parity error detected 1 Parity error detected (write 1 to clear)
14 RTSS	$\overline{\text{RTS}}$ **Pin Status**. Indicates the current status of the $\overline{\text{RTS}}$ pin. A "snapshot" of the pin is taken immediately before RTSS is presented to the data bus. RTSS cannot be cleared because all writes to RTSS are ignored. At reset, RTSS is set to 0. 0 The $\overline{\text{RTS}}$ pin is high (inactive) 1 The $\overline{\text{RTS}}$ pin is low (active)
13 TRDY	**Transmitter Ready Interrupt / DMA Flag**. Indicates that the TxFIFO emptied below its target threshold and requires data. TRDY is automatically cleared when the data level in the TxFIFO exceeds the threshold set by TXTL bits. At reset, TRDY is set to 1.

Table continues on the next page...

UARTx_USR1 field descriptions (continued)

Field	Description
	0 The transmitter does not require data 1 The transmitter requires data (interrupt posted)
12 RTSD	**RTS Delta.** Indicates whether the $\overline{\text{RTS}}$ pin changed state. It (RTSD) generates a maskable interrupt. When in STOP mode, RTS assertion sets RTSD and can be used to wake the processor. The current state of the $\overline{\text{RTS}}$ pin is available on the RTSS bit. Clear RTSD by writing 1 to it. Writing 0 to RTSD has no effect. At reset, RTSD is set to 0. 0 $\overline{\text{RTS}}$ pin did not change state since last cleared 1 $\overline{\text{RTS}}$ pin changed state (write 1 to clear)
11 ESCF	**Escape Sequence Interrupt Flag**. Indicates if an escape sequence was detected. ESCF is asserted when the ESCEN bit is set and an escape sequence is detected in the RxFIFO. Clear ESCF by writing 1 to it. Writing 0 to ESCF has no effect. 0 No escape sequence detected 1 Escape sequence detected (write 1 to clear).
10 FRAMERR	**Frame Error Interrupt Flag**. Indicates that a frame error is detected. The *interrupt_uart* interrupt will be generated if a frame error is detected and the interrupt is enabled. Clear FRAMERR by writing 1 to it. Writing 0 to FRAMERR has no effect. 0 No frame error detected 1 Frame error detected (write 1 to clear)
9 RRDY	**Receiver Ready Interrupt / DMA Flag**. Indicates that the RxFIFO data level is above the threshold set by the RXTL bits. (See the RXTL bits description in UART FIFO Control Register (UART_UFCR) for setting the interrupt threshold.) When asserted, RRDY generates a maskable interrupt or DMA request. RRDY is automatically cleared when data level in the RxFIFO goes below the set threshold level. At reset, RRDY is set to 0. 0 No character ready 1 Character(s) ready (interrupt posted)
8 AGTIM	Ageing Timer Interrupt Flag. Indicates that data in the RxFIFO has been idle for a time of 8 character lengths (where a character length consists of 7 or 8 bits, depending on the setting of the WS bit in UCR2, with the bit time corresponding to the baud rate setting) and FIFO data level is less than RxFIFO threshold level (RXTL in the UFCR). Clear by writing a 1 to it. 0 AGTIM is not active 1 AGTIM is active (write 1 to clear)
7 DTRD	**DTR Delta.** Indicates whether $\overline{\text{DTR}}$ (in DCE mode) or $\overline{\text{DSR}}$ (in DTE mode) pins changed state. DTRD generates a maskable interrupt if DTRDEN (UCR3[3]) is set. Clear DTRD by writing 1 to it. Writing 0 to DTRD has no effect. 0 $\overline{\text{DTR}}$ (DCE) or $\overline{\text{DSR}}$ (DTE) pin did not change state since last cleared 1 $\overline{\text{DTR}}$ (DCE) or $\overline{\text{DSR}}$ (DTE) pin changed state (write 1 to clear)
6 RXDS	Receiver IDLE Interrupt Flag. Indicates that the receiver state machine is in an IDLE state, the next state is IDLE, and the receive pin is high. RXDS is automatically cleared when a character is received. RXDS is active only when the receiver is enabled. 0 Receive in progress 1 Receiver is IDLE
5 AIRINT	Asynchronous IR WAKE Interrupt Flag. Indicates that the IR WAKE pulse was detected on the RXD pin. Clear AIRINT by writing 1 to it. Writing 0 to AIRINT has no effect.

Table continues on the next page...

UARTx_USR1 field descriptions (continued)

Field	Description
	0 No pulse was detected on the RXD IrDA pin 1 A pulse was detected on the RXD IrDA pin
4 AWAKE	Asynchronous WAKE Interrupt Flag. Indicates that a falling edge was detected on the RXD pin. Clear AWAKE by writing 1 to it. Writing 0 to AWAKE has no effect. 0 No falling edge was detected on the RXD Serial pin 1 A falling edge was detected on the RXD Serial pin
3 -	Reserved
2–0 -	Reserved

55.13.9 UART Status Register 2 (UARTx_USR2)

Addresses: UART-3_USR2 is 5000_C000h base + 98h offset = 5000_C098h
UART-1_USR2 is 53FB_C000h base + 98h offset = 53FB_C098h
UART-2_USR2 is 53FC_0000h base + 98h offset = 53FC_0098h
UART-4_USR2 is 53FF_C000h base + 98h offset = 53FF_C098h
UART-5_USR2 is 63F9_0000h base + 98h offset = 63F9_0098h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADET	TXFE	DTRF	IDLE	ACST	RIDELT	RIIN	IRINT	WAKE	DCDDELT	DCDIN	RTSF	TXDC	BRCD	ORE	RDR
Reset	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0

UARTx_USR2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15 ADET	**Automatic Baud Rate Detect Complete**. Indicates that an "A" or "a" was received and that the receiver detected and verified the incoming baud rate. Clear ADET by writing 1 to it. Writing 0 to ADET has no effect.

Table continues on the next page...

UARTx_USR2 field descriptions (continued)

Field	Description
	0 ASCII "A" or "a" was not received 1 ASCII "A" or "a" was received (write 1 to clear)
14 TXFE	**Transmit Buffer FIFO Empty**. Indicates that the transmit buffer (TxFIFO) is empty. TXFE is cleared automatically when data is written to the TxFIFO. Even though TXFE is high, the transmission might still be in progress. 0 The transmit buffer (TxFIFO) is not empty 1 The transmit buffer (TxFIFO) is empty
13 DTRF	**DTR edge triggered interrupt flag**. This bit is asserted, when the programmed edge is detected on the $\overline{DTR}$ pin (DCE mode) or on $\overline{DSR}$ (DTE mode). This flag can cause an interrupt if DTREN (UCR3[13]) is enabled. 0 Programmed edge not detected on DTR/DSR 1 Programmed edge detected on DTR/DSR (write 1 to clear)
12 IDLE	**Idle Condition**. Indicates that an idle condition has existed for more than a programmed amount frame (see Idle Line Detect. An interrupt can be generated by this IDLE bit if IDEN (UCR1[12]) is enabled. IDLE is cleared by writing 1 to it. Writing 0 to IDLE has no effect. 0 No idle condition detected 1 Idle condition detected (write 1 to clear)
11 ACST	**Autobaud Counter Stopped**. In autobaud detection (ADBR=1), indicates the counter which determines the baud rate was running and is now stopped. This means either START bit is finished (if ADNIMP=1), or Bit 0 is finished (if ADNIMP=0). See New Autobaud Counter Stopped bit and Interrupt, for more details. An interrupt can be flagged on *interrupt_uart* if ACIEN=1. 0 Measurement of bit length not finished (in autobaud) 1 Measurement of bit length finished (in autobaud). (write 1 to clear)
10 RIDELT	**Ring Indicator Delta**. This bit is used in DTE mode to indicate that the Ring Indicator input ($\overline{RI}$) has changed state. This flag can generate an interrupt if RI (UCR3[8]) is enabled. RIDELT is cleared by writing 1 to it. Writing 0 to RIDELT has no effect. 0 Ring Indicator input has not changed state 1 Ring Indicator input has changed state (write 1 to clear)
9 RIIN	**Ring Indicator Input**. This bit is used in DTE mode to reflect the status if the Ring Indicator input ($\overline{RI}$). The Ring Indicator input is used to indicate that a ring has occurred. In DCE mode this bit is always zero. 0 Ring Detected 1 No Ring Detected
8 IRINT	**Serial Infrared Interrupt Flag.** When an edge is detected on the RXD pin during SIR Mode, this flag will be asserted. This flag can cause an interrupt which can be masked using the control bit ENIRI: UCR4 [8]. 0 no edge detected 1 valid edge detected (write 1 to clear)
7 WAKE	**Wake**. Indicates the start bit is detected. WAKE can generate an interrupt that can be masked using the WKEN bit. Clear WAKE by writing 1 to it. Writing 0 to WAKE has no effect. 0 start bit not detected 1 start bit detected (write 1 to clear)

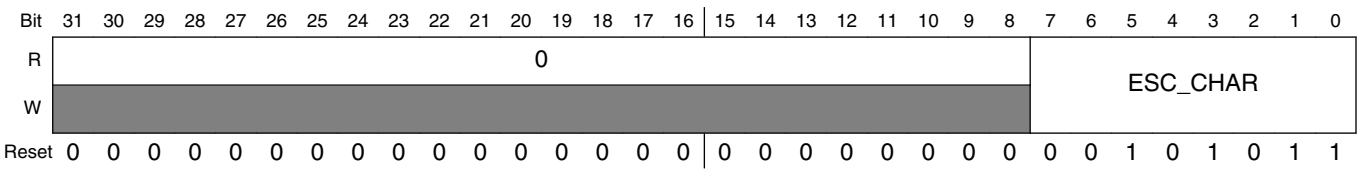
Table continues on the next page...

UARTx_USR2 field descriptions (continued)

Field	Description
6 DCDDELT	**Data Carrier Detect Delta**. This bit is used in DTE mode to indicate that the Data Carrier Detect input ($\overline{DCD}$) has changed state. This flag can cause an interrupt if DCD (UCR3[9]) is enabled. When in STOP mode, this bit can be used to wake the processor. In DCE mode this bit is always zero. 0 Data Carrier Detect input has not changed state 1 Data Carrier Detect input has changed state (write 1 to clear)
5 DCDIN	**Data Carrier Detect Input**. This bit is used in DTE mode reflect the status of the Data Carrier Detect input ($\overline{DCD}$). The Data Carrier Detect input is used to indicate that a carrier signal has been detected. In DCE mode this bit is always zero. 0 Carrier signal Detected 1 No Carrier signal Detected
4 RTSF	**RTS Edge Triggered Interrupt Flag. Indicates if** a programmed edge is detected on the $\overline{RTS}$ pin. The RTEC bits select the edge that generates an interrupt (see Table 55-4). RTSF can generate an interrupt that can be masked using the RTSEN bit. Clear RTSF by writing 1 to it. Writing 0 to RTSF has no effect. 0 Programmed edge not detected on $\overline{RTS}$ 1 Programmed edge detected on $\overline{RTS}$ (write 1 to clear)
3 TXDC	**Transmitter Complete**. Indicates that the transmit buffer (TxFIFO) and Shift Register is empty; therefore the transmission is complete. TXDC is cleared automatically when data is written to the TxFIFO. 0 Transmit is incomplete 1 Transmit is complete
2 BRCD	**BREAK Condition Detected**. Indicates that a BREAK condition was detected by the receiver. Clear BRCD by writing 1 to it. Writing 0 to BRCD has no effect. 0 No BREAK condition was detected 1 A BREAK condition was detected (write 1 to clear)
1 ORE	**Overrun Error**. When set to 1, ORE indicates that the receive buffer (RxFIFO) was full (32 chars inside), and a 33rd character has been fully received. This 33rd character has been discarded. Clear ORE by writing 1 to it. Writing 0 to ORE has no effect. 0 No overrun error 1 Overrun error (write 1 to clear)
0 RDR	**Receive Data Ready**-Indicates that at least 1 character is received and written to the RxFIFO. If the URXD register is read and there is only 1 character in the RxFIFO, RDR is automatically cleared. 0 No receive data ready 1 Receive data ready

55.13.10 UART Escape Character Register (UARTx_UESC)

Addresses: UART-3_UESC is 5000_C000h base + 9Ch offset = 5000_C09Ch
UART-1_UESC is 53FB_C000h base + 9Ch offset = 53FB_C09Ch
UART-2_UESC is 53FC_0000h base + 9Ch offset = 53FC_009Ch
UART-4_UESC is 53FF_C000h base + 9Ch offset = 53FF_C09Ch
UART-5_UESC is 63F9_0000h base + 9Ch offset = 63F9_009Ch

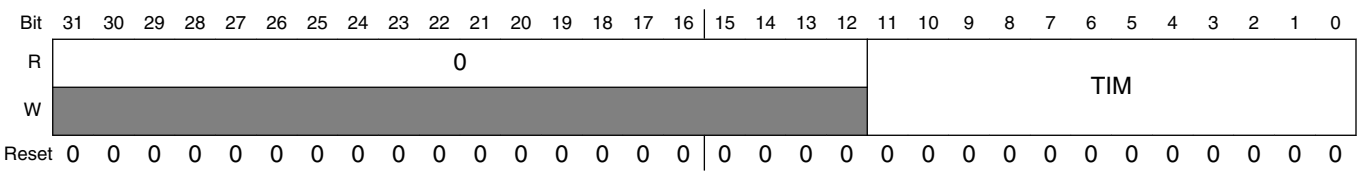


UARTx_UESC field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value zero. Reserved
7–0 ESC_CHAR	**UART Escape Character**. Holds the selected escape character that all received characters are compared against to detect an escape sequence.

55.13.11 UART Escape Timer Register (UARTx_UTIM)

Addresses: UART-3_UTIM is 5000_C000h base + A0h offset = 5000_C0A0h
UART-1_UTIM is 53FB_C000h base + A0h offset = 53FB_C0A0h
UART-2_UTIM is 53FC_0000h base + A0h offset = 53FC_00A0h
UART-4_UTIM is 53FF_C000h base + A0h offset = 53FF_C0A0h
UART-5_UTIM is 63F9_0000h base + A0h offset = 63F9_00A0h



UARTx_UTIM field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value zero. Reserved
11–0 TIM	UART Escape Timer. Holds the maximum time interval (in ms) allowed between escape characters. The escape timer register is programmable in intervals of 2 ms. See Escape Sequence Detection and Table 55-9 for more information on the UART escape sequence detection. Reset value 0x000 = 2 ms up to 0xFFF = 8.192 s.

55.13.12 UART BRM Incremental Register (UARTx_UBIR)

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write 0x000F value into the UBIR after finishing detecting baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[1].

Please note software reset will reset the register to its reset value.

Addresses: UART-3_UBIR is 5000_C000h base + A4h offset = 5000_C0A4h
UART-1_UBIR is 53FB_C000h base + A4h offset = 53FB_C0A4h
UART-2_UBIR is 53FC_0000h base + A4h offset = 53FC_00A4h
UART-4_UBIR is 53FF_C000h base + A4h offset = 53FF_C0A4h
UART-5_UBIR is 63F9_0000h base + A4h offset = 63F9_00A4h

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

UARTx_UBIR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 INC	Incremental Numerator. Holds the numerator value minus one of the BRM ratio (see Binary Rate Multiplier (BRM)). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this field using byte accesses is not recommended and is undefined.

1. Note: The write priority in the new design is not same as the original UART. In the orginal design, software has higher priotiry than hardware when writing this register at the same time.

55.13.13 UART BRM Modulator Register (UARTx_UBMR)

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write a proper value into the UBMR based on detected baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[1].

Please note software reset will reset the register to its reset value.

Addresses: UART-3_UBMR is 5000_C000h base + A8h offset = 5000_C0A8h

UART-1_UBMR is 53FB_C000h base + A8h offset = 53FB_C0A8h

UART-2_UBMR is 53FC_0000h base + A8h offset = 53FC_00A8h

UART-4_UBMR is 53FF_C000h base + A8h offset = 53FF_C0A8h

UART-5_UBMR is 63F9_0000h base + A8h offset = 63F9_00A8h

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

UARTx_UBMR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 MOD	**Modulator Denominator.** Holds the value of the denominator minus one of the BRM ratio (see Binary Rate Multiplier (BRM)). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this register using byte accesses is not recommended and undefined.

1. Note: The write priority in the new design is not same as the original UART. In the orginal design, software has higher priotiry than hardware when writing this register at the same time.

55.13.14 UART Baud Rate Count Register (UARTx_UBRC)

Addresses: UART-3_UBRC is 5000_C000h base + ACh offset = 5000_C0ACh

UART-1_UBRC is 53FB_C000h base + ACh offset = 53FB_C0ACh

UART-2_UBRC is 53FC_0000h base + ACh offset = 53FC_00ACh

UART-4_UBRC is 53FF_C000h base + ACh offset = 53FF_C0ACh

UART-5_UBRC is 63F9_0000h base + ACh offset = 63F9_00ACh

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

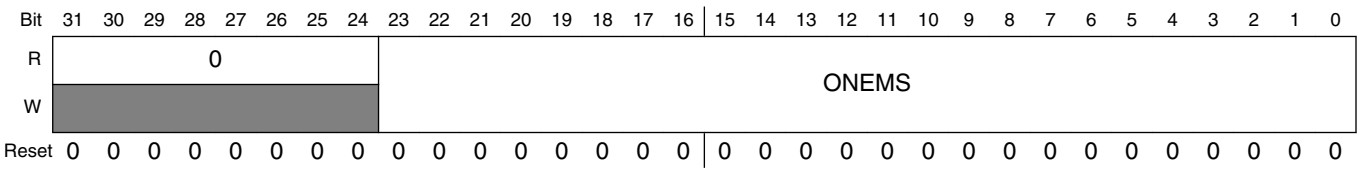
UARTx_UBRC field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value zero. Reserved
15–0 BCNT	**Baud Rate Count Register.** This read only register is used to count the start bit of the incoming baud rate (if ADNIMP=1), or start bit + bit0 (if ADNIMP=0). When the measurement is done, the Baud Rate Count Register contains the number of UART internal clock cycles (clock after divider) present in an incoming bit. BCNT retains its value until the next Automatic Baud Rate Detection sequence has been initiated. The 16 bit Baud Rate Count register is reset to 4 and stays at hex FFFF in the case of an overflow.

55.13.15 UART One Millisecond Register (UARTx_ONEMS)

This register has been expanded from 16 bits to 24 bits. In previous versions, the 16-bit ONEMS can only support the maximum 65.535MHz (0xFFFFx1000) ref_clk. To support 4Mbps Bluetooth application with 66.5MHz module_clock, the value 0x103C4 (66.5M/1000) should be written into this register. In this case, the 16 bits are not enough to contain the 0x103C4. So this register was expanded to 24 bits to support high frequency of the ref_clk.

Addresses: UART-3_ONEMS is 5000_C000h base + B0h offset = 5000_C0B0h
UART-1_ONEMS is 53FB_C000h base + B0h offset = 53FB_C0B0h
UART-2_ONEMS is 53FC_0000h base + B0h offset = 53FC_00B0h
UART-4_ONEMS is 53FF_C000h base + B0h offset = 53FF_C0B0h
UART-5_ONEMS is 63F9_0000h base + B0h offset = 63F9_00B0h

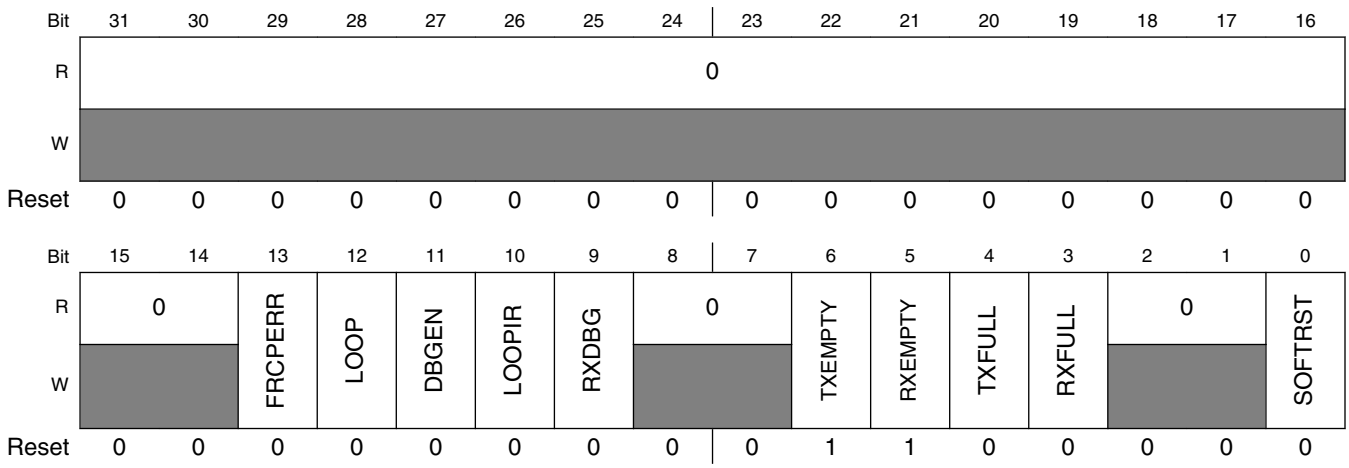


UARTx_ONEMS field descriptions

Field	Description
31–24 Reserved	This read-only field is reserved and always has the value zero. Reserved
23–0 ONEMS	**One Millisecond Register.** This 24-bit register must contain the value of the UART internal frequency (*ref_clk* in Figure 55-1) divided by 1000. The internal frequency is obtained after the UART BRM internal divider (F (*ref_clk*) = F(*module_clock*) / RFDIV). In fact this register contains the value corresponding to the number of UART BRM internal clock cycles present in one millisecond. The ONEMS (and UTIM) registers value are used in the escape character detection feature (Escape Sequence Detection) to count the number of clock cycles left between two escape characters. The ONEMS register is also used in infrared special case mode (IRSC = UCR4[5] = 1'b1), see InfraRed Special Case (IRSC) Bit.

55.13.16 UART Test Register (UARTx_UTS)

Addresses: UART-3_UTS is 5000_C000h base + B4h offset = 5000_C0B4h
UART-1_UTS is 53FB_C000h base + B4h offset = 53FB_C0B4h
UART-2_UTS is 53FC_0000h base + B4h offset = 53FC_00B4h
UART-4_UTS is 53FF_C000h base + B4h offset = 53FF_C0B4h
UART-5_UTS is 63F9_0000h base + B4h offset = 63F9_00B4h



UARTx_UTS field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value zero. Reserved
13 FRCPERR	Force Parity Error. Forces the transmitter to generate a parity error if parity is enabled. FRCPERR is provided for system debugging. 0 Generate normal parity 1 Generate inverted parity (error)
12 LOOP	Loop TX and RX for Test. Controls loopback for test purposes. When LOOP is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by LOOP. If RXDMUXSEL (UCR3[2]) is set to 1, the loopback is applied on serial and IrDA signals. If RXDMUXSEL is set to 0, the loopback is only applied on serial signals. 0 Normal receiver operation 1 Internally connect the transmitter output to the receiver input
11 DBGEN	$\overline{\text{debug\_enable}}$**.** This bit controls whether to respond to the *debug_req* input signal. 0 UART will go into debug mode when *debug_req* is HIGH 1 UART will not go into debug mode even if *debug_req* is HIGH
10 LOOPIR	**Loop TX and RX for IR Test (LOOPIR)**. This bit controls loopback from transmitter to receiver in the InfraRed interface.

Table continues on the next page...

UARTx_UTS field descriptions (continued)

Field	Description
	0 No IR loop 1 Connect IR transmitter to IR receiver
9 RXDBG	**RX_fifo_debug_mode.** This bit controls the operation of the RX fifo read counter when in debug mode. 0 rx fifo read pointer does not increment 1 rx_fifo read pointer increments as normal
8–7 Reserved	This read-only field is reserved and always has the value zero. Reserved
6 TXEMPTY	TxFIFO Empty. Indicates that the TxFIFO is empty. 0 The TxFIFO is not empty 1 The TxFIFO is empty
5 RXEMPTY	RxFIFO Empty. Indicates the RxFIFO is empty. 0 The RxFIFO is not empty 1 The RxFIFO is empty
4 TXFULL	TxFIFO FULL. Indicates the TxFIFO is full. 0 The TxFIFO is not full 1 The TxFIFO is full
3 RXFULL	RxFIFO FULL. Indicates the RxFIFO is full. 0 The RxFIFO is not full 1 The RxFIFO is full
2–1 Reserved	This read-only field is reserved and always has the value zero. Reserved
0 SOFTRST	Software Reset. Indicates the status of the software reset ($\overline{\text{SRST}}$ bit of UCR2). 0 Software reset inactive 1 Software reset active

Chapter 56
Universal Serial Bus OTG HOST1 (USBOH1)

56.1 Introduction

The USBOH1 module contains all of the functionality required to support 2 independent USB ports, compatible with the USB 2.0 specification. A block diagram of USBOH1 shows in Figure 56-1.

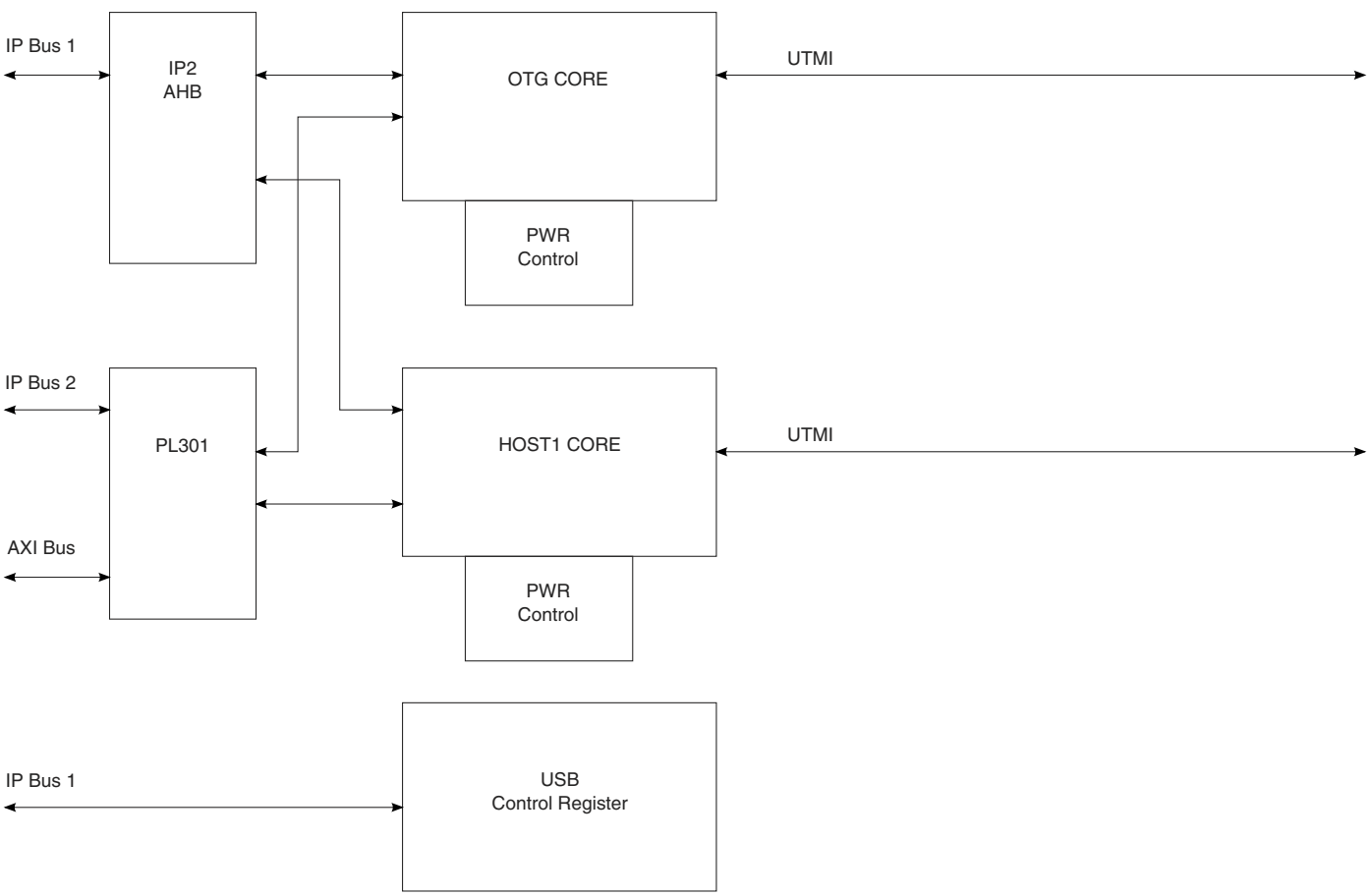


Figure 56-1. USBOH1 Block Diagram

56.1.1 Overview

The USBOH1 module provides high performance USB On-The-Go (OTG) functionality, compliant with the USB 2.0 specification, the OTG supplement. The module consists of 2 independent USB cores.

56.1.2 Features

The USBOH1 module includes the following features:

- High Speed / Full Speed / Low Speed Host only core (HOST 1)
  - HS/FS/LS UTMI compliant interface
- High speed OTG core
  - HS/FS/LS UTMI compliant interface
  - High Speed, Full Speed and Low Speed operation in HOST mode
  - High Speed and Full Speed operation in Peripheral mode
  - Hardware support for OTG signaling, Session Request Protocol and Host Negotiation Protocol
  - Up to 8 bidirectional endpoints in Peripheral mode
- Low power mode with local and remote wake-up capability
- Embedded DMA controller

56.1.3 Modes of Operation

Each USB core interfaces can be configured for High Speed operation (480 Mbps) and/or Full/Low speed operation (12/1.5 Mbps).

This chapter details the configuration options.

56.1.3.1 Operational Modes

56.1.3.1.1 Normal Mode

Each USB core controls its corresponding port. Each port can work in one or more modes

- Host Port 1:

This port supports UTMI Transceiver.

- OTG port:

  This port supports UTMI Transceiver

56.1.3.1.2 Low Power Mode

Each of the USB cores has an associated power control module that is controlled by the USB core and clocked on a 32 KHz clock. When a USB bus is idle, the transceiver can be placed in low power mode (suspend), after which the clocks to the USB core can be stopped. The 32 KHz low power clock must remain active as it is needed for wakeup detection.

Either the local CPU or the remote USB Host/Peripheral can initiate a wake-up sequence to resume USB communication.

56.2 External Signal Description

56.2.1 External Signal Overview

See Table 56-1 for the list of signals entering and existing this module to peripherals within the chip.

Table 56-1. Signal Properties

Name	Direction	Reset state	Pull up	Width	Function
OTG External Signals					
otg_utmi_reset	Output	0	-	1	UTMI Reset
otg_utmi_term_sel	Output	-	-	1	-
otg_utmi_xcvr_sel	Output	-	-	2	-
otg_utmi_line_state	Input	-	-	2	-
otg_utmi_op_mode	Output	-	-	2	-
otg_utmi_suspendm	Output	-	-	1	-
otg_utmi_data_in	Output	-	-	16	-
otg_utmi_tx_valid	Output	-	-	1	-
otg_utmi_tx_validh	Output	-	-	1	-
otg_utmi_tx_ready	Input	-	-		-
otg_utmi_data_out	Input	-	-	16	-

Table continues on the next page...

Table 56-1. Signal Properties (continued)

Name	Direction	Reset state	Pull up	Width	Function
otg_utmi_rx_valid	Input	-	-	1	-
otg_utmi_rx_validh	Input	-	-	1	-
otg_utmi_rx_active	Input	-	-	1	-
otg_utmi_rx_err	Input	-	-	1	-
otg_utmi_host_disconnect	Input	-	-	1	-
otg_utmi_dp_pulldown	Output	-	-	1	-
otg_utmi_dm_pulldown	Output	-	-	1	-
otg_utmi_drv_vbus	Output	-	-	1	Not connected in Chiplevel. Replaced by ipp_do_otg_pwr.
otg_utmi_chrg_vbus	Output	-	-	1	-
otg_utmi_dischrg_vbus	Output	-	-	1	-
otg_utmi_fsl_ser_mode	Output	-	-	1	-
otg_utmi_tx_enablez	Output	-	-	1	-
otg_utmi_tx_dat	Output	-	-	1	-
otg_utmi_tx_se0	Output	-	-	1	-
otg_utmi_rxrcv	Input	-	-	1	-
otg_utmi_rxdm	Input	-	-	1	-
otg_utmi_rxdp	Input	-	-	1	-
otg_utmi_id_dig	Input	-	-	1	-
otg_utmi_id_pullup	Output	-	-	1	-
otg_utmi_vbus_valid	Input	-	-	1	-
otg_utmi_a_valid	Input	-	-	1	-
otg_utmi_b_valid	Input	-	-	1	-
otg_utmi_sess_end	Input	-	-	1	-
otg_utmi_tx_bit_stuff_en	Output	-	-	1	-
otg_utmi_tx_bit_stuff_en_h	Output	-	-	1	-
otg_utmi_databus16_8	Output	-	-	1	-
otg_utmi_sieclock	Input	-	-	1	-
otg_utmi_on_clock	Output	-	-	1	-
otg_phy_hsdrvtimingp	Output	-	-	1	-
otg_phy_hsdrvtimingn	Output	-	-	2	-

Table continues on the next page...

Table 56-1. Signal Properties (continued)

Name	Direction	Reset state	Pull up	Width	Function
otg_phy_hsdrvamplitude	Output	-	-	2	-
otg_phy_hsdrvslope	Output	-	-	4	-
otg_phy_hsdedvsel	Output	-	-	2	-
otg_phy_hstedvsel	Output	-	-	2	-
otg_phy_fstunevsel	Output	-	-	3	-
otg_phy_icpctrl	Output	-	-	2	-
otg_phy_calbp	Output	-	-	1	-
otg_phy_extcal	Output	-	-	5	-
otg_phy_fsrftsel	Output	-	-	2	-
otg_phy_lsrftsel	Output	-	v	2	-
otg_phy_enpre	Output	-	-	1	-
otg_phy_preemdepth	Output	-	-	1	-
otg_phy_plldivvalue	Output	-	-	2	-
otg_phy_vload	Output	-	-	1	-
otg_phy_vcontrol	Output	-	-	4	-
otg_phy_conf2	Output	-	-	1	-
otg_phy_conf3	Output	-	-	1	-
otg_phy_chgrdeten	Output	-	-	1	-
otg_phy_chgrdeton	Output	-	-	1	-
otg_phy_chgrdet	Input	-	-	1	-
otg_phy_vstatus	Input	-	-	8	-
Host1 Signals					
uh1_utmi_reset	Output	0	-	1	UTMI Reset
uh1_utmi_term_sel	Output	-	v	1	-
uh1_utmi_xcvr_sel	Output	-	-	2	-
uh1_utmi_line_state	Input	-	-	2	-
uh1_utmi_op_mode	Output	-	-	2	-
uh1_utmi_suspendm	Output	-	-	1	-
uh1_utmi_data_in	Output	-	-	16	-
uh1_utmi_tx_valid	Output	-	-	1	-
uh1_utmi_tx_validh	Output	-	-	1	-
uh1_utmi_tx_ready	Input	-	-		-
uh1_utmi_data_out	Input	-	-	16	-
uh1_utmi_rx_valid	Input	-	-	1	-

Table continues on the next page...

Table 56-1. Signal Properties (continued)

Name	Direction	Reset state	Pull up	Width	Function
uh1_utmi_rx_validh	Input	-	-	1	-
uh1_utmi_rx_active	Input	-	-	1	-
uh1_utmi_rx_err	Input	-	-	1	-
uh1_utmi_host_disconnect	Input	-	-	1	-
uh1_utmi_dp_pulldown	Output	-	-	1	-
uh1_utmi_dm_pulldown	Output	-	-	1	-
uh1_utmi_drv_vbus	Output	-	-	1	Not connected in Chiplevel. Replaced by ipp_do_uh1_pwr.
uh1_utmi_chrg_vbus	Output	-	-	1	-
uh1_utmi_dischrg_vbus	Output	-	-	1	-
uh1_utmi_fsl_ser_mode	Output	-	-	1	-
uh1_utmi_tx_enablez	Output	-	-	1	-
uh1_utmi_tx_dat	Output	-	-	1	-
uh1_utmi_tx_se0	Output	-	-	1	-
uh1_utmi_rxrcv	Input	-	-	1	-
uh1_utmi_rxdm	Input	-	-	1	-
uh1_utmi_rxdp	Input	-	-	1	-
uh1_utmi_id_dig	Input	-	-	1	-
uh1_utmi_id_pullup	Output	-	-	1	-
uh1_utmi_vbus_valid	Input	-	-	1	-
uh1_utmi_a_valid	Input	-	-	1	-
uh1_utmi_b_valid	Input	-	-	1	-
uh1_utmi_sess_end	Input	-	-	1	-
uh1_utmi_tx_bit_stuff_en	Output	-	-	1	-
uh1_utmi_tx_bit_stuff_en_h	Output	-	-	1	-
uh1_utmi_databus16_8	Output	-	-	1	-
uh1_utmi_sieclock	Input	-	-	1	-
uh1_utmi_on_clock	Output	-	-	1	-
uh1_phy_hsdrvtimingp	Output	-	-	1	-
uh1_phy_hsdrvtimingn	Output	-	-	2	-

Table continues on the next page...

Table 56-1. Signal Properties (continued)

Name	Direction	Reset state	Pull up	Width	Function
uh1_phy_hsdrvamplitude	Output	-	-	2	-
uh1_phy_hsdrvslope	Output	-	-	4	-
uh1_phy_hsdedvsel	Output	-	-	2	-
uh1_phy_hstedvsel	Output	-	-	2	-
uh1_phy_fstunevsel	Output	-	-	3	-
uh1_phy_icpctrl	Output	-	-	2	-
uh1_phy_calbp	Output	-	-	1	-
uh1_phy_extcal	Output	-	-	5	-
uh1_phy_fsrftsel	Output	-	-	2	-
uh1_phy_lsrftsel	Output	-	-	2	-
uh1_phy_enpre	Output	-	-	1	-
uh1_phy_preemdepth	Output	-	-	1	-
uh1_phy_plldivvalue	Output	-	-	2	-
uh1_phy_vload	Output	-	-	1	-
uh1_phy_vcontrol	Output	-	-	4	-
uh1_phy_conf2	Output	-	-	1	-
uh1_phy_conf3	Output	-	-	1	-
uh1_phy_chgrdeten	Output	-	-	1	-
uh1_phy_chgrdeton	Output	-	-	1	-
uh1_phy_chgrdet	Input	-	-	1	-
uh1_phy_vstatus	Input	-	-	8	-
ipp_do_otg_pwr	Output	0	-	1	USB OTG port power control.
ipp_ind_otg_oc	Input	-	-	1	USB OTG overcurrent.
ipp_do_uh1_pwr	Output	0	-	1	USB Host1 port power control.
ipp_ind_uh1_oc	Input	-	-	1	USB Host1 overcurrent.

56.2.2 Detailed Signal Descriptions

Detailed Signal Description for each module lists in Functional Description.

56.3 Functional Description

This sections describes the functionality and the topology of the different building blocks of the USB module.

56.3.1 USB HOST Controller 1

Host Controller 1 is an instantiation of the core. The core is configured for High Speed / Full Speed / Low Speed operation.

56.3.2 USB OTG Controller

The OTG controller offers HS/FS/LS capabilities in Host mode and HS/FS in device mode.

56.3.2.1 Host Mode

The controller supports direct connection of a HS/FS/LS device (without external hub) with internal UTMI transceiver. Although there is no separate Transaction Translator block in the system. The transaction translator function normally associated with a USB 2.0 high speed hub has been implemented within the DMA and Protocol engine blocks to support connection to full and low speed devices.

56.3.2.2 Peripheral (Device) Mode

- Up to 8 bidirectional endpoints
- High/Full speed operation
- Supports HNP, SRP.
- Remote wakeup capable

56.3.3 USB Power Control Module

The HS-USB module supports suspend and wakeup functionality, but the circuit is considered application specific and therefore not part of the IP. An external circuit has been designed to place external transceivers in suspend mode, and wake them up either

on a local request (CPU initiated) or on remote request by detecting activity on the USB line. The wake-up logic can optionally wake the CPU when it is in sleep mode at the time of the request.

56.3.3.1 Entering Suspend Mode

Suspend mode is always entered under control of driver software by setting the appropriate bit in PORTSC register. Once the controller is suspended, the clocks to the USB block can be stopped.

56.3.3.2 Wake-up Events

The power control module monitors the USB bus when the USB core is in the suspend state. Depending on whether the core is on Host or Device mode, a number of wakeup conditions are detected. Upon detection of a wakeup condition, an interrupt (asynchronous) is generated on the CPU complex. This interrupt will also re-activate the clocks if these were stopped during the suspend.

56.3.3.2.1 Host Mode Events

The Host controller wakes-up on the following events:

- Remote Wakeup request

  A peripheral can request the host to re-activate the bus by driving wake-up signaling on the Dm/Dp lines. The power control module will detect a J-K transition on the Dm/Dp lines and signal the wakeup request to the core.

- Wake on Over-current

  If wake on overcurrent is enabled in the PORTSC registers, the power control module will signal a wakeup condition to the USB core.

- Wake on Disconnect

  The Power Control module detects disconnect events by monitoring the Dp/Dm lines. When a disconnect event is detected (Dm = Dp = 0) and the Wake on Disconnect is enabled in the PORTSC register, the core will be notified.

- Wake on Connect

Similar to the Wake on Disconnect, the power control module detects a connect event (Dm or Dp High) and signals this to the USB core by setting the pwrctl_wakeup signal if enabled in the PORTSC register.

56.3.3.2.2 Device Mode Events

When the OTG controller is configured for peripheral operation, the power control module will detected following events:

- Detection of bus activity

  Any non-idle condition on the USB bus will activate the wakeup output of the power control module to notify the USB core of the wakeup event.

- Device Connection detection

  When using on-chip UTMI PHY, the USB module can enter into suspend mode if nothing is attached to the OTG port. After a device is connected to the port, the ID pin changes from 1 to 0, activating the wakeup output of the power control module which notifies the USB core of the wakeup event. This event is generated only if the proper bit is set in USB_CTL register.

- Host Connection detection

  When using on-chip UTMI PHY, the USB module can enter into suspend mode if nothing is attached to the OTG port. After a device is connected to the port, the VBUS pin changes from 0 to 1, activating the wakeup output of the power control module which notifies the USB core of the wakeup event.This event is generated only when proper bit is set in USB_CTL register.

56.3.4 USB Interrupts

56.3.4.1 USB Core Interrupts

Each USB core uses one dedicated vector in the Interrupt Table. The vector numbers associated with each of the cores can be found in the Interrupt section.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB Cores. Please refer to the USB Core documentation for details.

56.3.4.2 USB Wake-up Interrupts

Each USB Core has an associated wake-up interrupt. The wake-up interrupts are generated outside the USB cores' but use the same vector as the corresponding Cores' interrupt. These interrupt are generated by the Power Control Modules which run on the 32KHz standby clock.The wake-up interrupt is designed to work even when the USB and CPU clocks are disabled, such that a wake-up condition on the USB bus can re-activate the CPU clocks. Therefore, this interrupt request propagates thru combinational logic to the CPU's interrupt module.

Because the wake-up interrupt is generated and cleared on a 32 KHz clock, this interrupt request will respond very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt will mask the request instantaneously as this is clocked by the CPU clock. The software should than wait for at least 3 32 KHz clock cycles before re-enabling this interrupt to allow sufficient time for the request flag to clear. As this interrupt is only used during low-power modes of the USB, it is sufficient to enable the wake-up interrupt just prior to enter USB suspend mode.

56.4 Initialization/Application Information

This section described the detailed application knowledge for Host1 and OTG ports. It can be generally divided in two parts, one is for Host and the other is for Device. Host part is applied to two ports, Device part is only applied to OTG port. In following register description, Device related content is just for OTG, and Host related content is for both ports.

56.4.1 Software Model

The Device API provides a framework of routines to control the USB-HS OTG High-Speed USB On-The-Go peripheral in USB device applications. It includes an application to respond to the Chapter 9 device framework commands issued by a USB host.

The USB-HS OTG High-Speed USB On-The-Go Device API is designed to significantly simplify the software tasks required to develop a USB device application. The API presents a high-level data transfer interface to the user's application code. All the register, interrupt and DMA interactions with the USB-HS OTG High-Speed USB On-The-Go core are managed by the API. The API also includes routines that handle all the USB device framework (Chapter 9) commands which are required for all USB devices.

The Host Stack provides a layered software architecture to control all aspects of a USB bus system. The Host Controller Device (HCD) interface controls the functions of an embedded EHCI host controller. The USB driver layer provides all the USB driver functions to enumerate, manage and schedule a USB bus system, while the upper layers of the stack support standard USB device class interfaces to the device drives running on your embedded system.

For details on the HS-USB OTG High-Speed USB On-The-Go Software Stack refer the documentation provided with the software products.

- **ANSI-C OTG software Stack provides Host and Device application support.** USB software included with the HS-USB OTG High-Speed USB On-The-Go core is tested with the hardware.
- **OTG Application Program Interface (API) handles OTG protocols.** Connect and disconnect events are handled as well as the OTG Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) state machines. The OTG code calls the Host or Device API functions based on the connection state of the OTG state machines.
- **Host API to speed up Host software development.** Simple API calls allow direct interaction with USB pipes. Additional layers support bus enumeration, bus management and a growing set of supported USB classes.
- **Device API to speed up peripheral development.** USB peripheral characteristics such as endpoints, configurations, interfaces, and alternate settings are controlled by supplied ANSI-C firmware. Chapter 9 Device Frame work command set reduces software development time. Simple API interface allows quick coding of USB device applications.

56.4.1.1 Device Data Structure

The function of the device operation3 is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list **transfer descriptors**, pointed to by a **queue head**, the device controller will perform the data transfers.

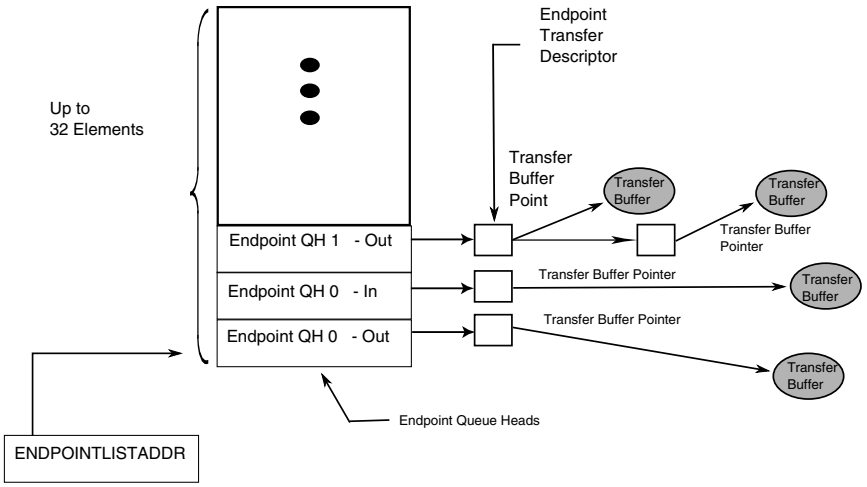


Figure 56-2. End Point Queue Head Organization

The HS-USB OTG High-Speed USB On-The-Go Device API incorporates and abstracts for the application developer all of the information contained in the device operational model.

56.4.1.2 Host Data Structure

The host data structures are used to communicate control, status, and data between software and the Host Controller. The **Periodic Frame List** is an array of pointers for the periodic schedule. A sliding window on the Periodic Frame List is used. The **Isochronous Transfer List** is where all the control and bulk transfers are managed. The HS-USB OTG High-Speed USB On-The-Go Host API incorporates and abstracts for the application developer all of the information contained in the host operational model.

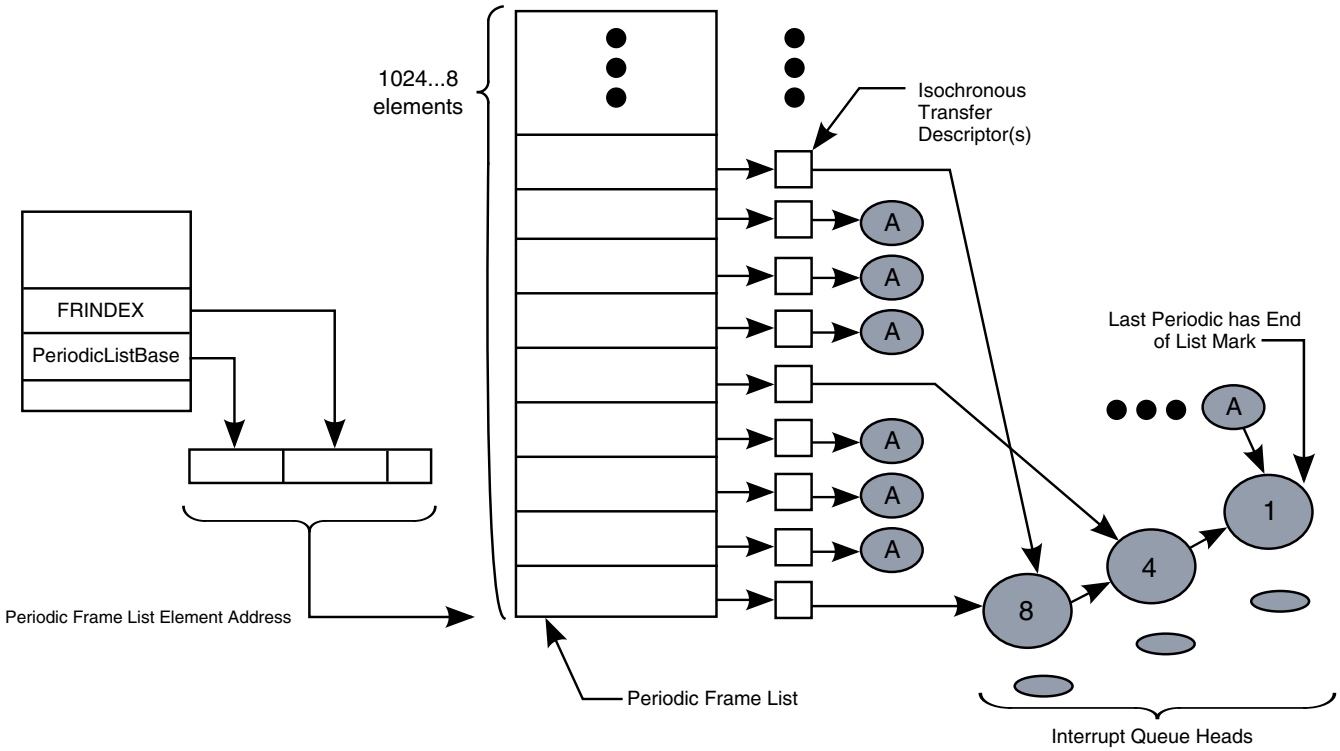


Figure 56-3. Periodic Schedule Organization

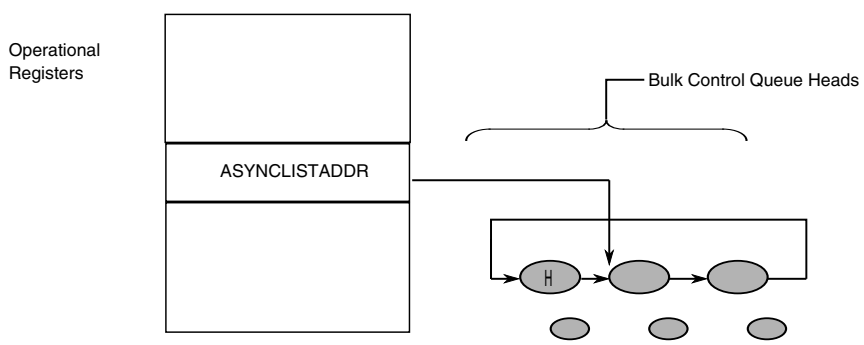


Figure 56-4. Asynchronous Schedule Organization

56.4.2 Register Interface

Slave accesses from the controlling processor enables access to the configuration, control, and status registers. One function of the system address map is the registers base address, which must begin on a **DWord** (32-bit) boundary. Register offset definitions are listed in the table below.

Configuration, control and status registers are divided into three categories, identification, capability and operational registers.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.
- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are comprised of dynamic control or status registers that may be read only, read/write, or read/write to clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

NOTE

Host mode EHCI compatibility begins at offset 0x100. If it is necessary to begin the EHCI register set at offset 0x000, the identification registers are disabled from the address map by connecting the upper most address bit of the slave interface to a logic level '1' and adjusting the offsets below accordingly.

Table 56-2. Interface Register Sets

Offset	Register Set	Explanation
000h to 0fch	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
100h to 124h	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
140h to 1FCh	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

56.4.2.1 Configuration, Control and Status register set

Table 56-3. Device/Host Capability Registers

Offset	Size (bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
000h	4	ID	Identification Register	÷	÷	÷	÷
004h	4	HWGENERAL	General Hardware Parameters	÷	÷	÷	÷
008h	4	HWHOST	Host Hardware Parameters		÷	÷	÷
00Ch	4	HWDEVICE	Device Hardware Parameters	÷	÷		
010h	4	HWTXBUF	TX Buffer Hardware Parameters	÷	÷	÷	÷
014h	4	HWRXBUF	RX Buffer Hardware Parameters	÷	÷	÷	÷
018h	4	HWTTTXBUF	TT-TX Buffer Hardware Parameters				÷
01Ch	4	HWTTRXBUF	TT-RX Buffer Hardware Parameters				÷
020h-07Ch	96	Reserved	N/A				
080h	4	GPTIMER0LD	General Purpose Timer #0 Load Register				
084h	4	GPTIMER0CTRL	General Purpose Timer #0 Control Register				
088h	4	GPTIMER1LD	General Purpose Timer #1 Load Register				
08ch	4	GPTIMER1CTRL	General Purpose Timer #1 Control Register				
090h	4	SBUSCFG	Control for the system bus interface				
094h-0FCh	108	Reserved	N/A				
100h	1	CAPLENGTH	Capability Register Length	÷	÷	÷	÷
101h	1	Reserved	N/A				
102h	2	HCIVERSION	Host Interface Version Number		÷	÷	÷
104h	4	HCSPARAMS	Host Ctrl. Structural Parameters		÷	÷	÷
108h	4	HCCPARAMS	Host Ctrl. Capability Parameters		÷	÷	÷
10Ch - 11Ch	20	Reserved	N/A				
120h	2	DCIVERSION	Dev. Interface Version Number	÷	÷		

Table continues on the next page...

Table 56-3. Device/Host Capability Registers (continued)

Offset	Size (bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
122h	2	Reserved	N/A	÷			
124h	4	DCCPARAMS	Device Ctrl. Capability Parameters	÷	÷		
128h - 13Ch	24	Reserved	N/A				
140h	4	USBCMD	USB Command	÷	÷	÷	÷
144h	4	USBSTS	USB Status	÷	÷	÷	÷
148h	4	USBINTR	USB Interrupt Enable	÷	÷	÷	÷
14Ch	4	FRINDEX	USB Frame Index	÷	÷	÷	÷
150h	4	Reserved	4G Segment Selector				
154h	4	PERIODICLISTBASE	Frame List Base Address		÷	÷	÷
		Device Addr	USB Device Address	÷	÷		
158h	4	ASYNCLISTADDR	Next Asynchronous List Address		÷	÷	÷
		Endpointlist Addr	Address at Endpoint list in memory	÷	÷		
15Ch	4	ASYNCTTSTS	Asynchronous Buffer Status For Embedded TT.				÷
160h	4	BURSTSIZE	Programmable Burst Size	÷	÷	÷	÷
164h	4	TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning		÷	÷	÷
168h	4	TXTTFILLTUNING	Host TT Transmit Pre-Buffer Packet Tuning				÷
16Ch	4	IC_USB	IC_USB enable and voltage negotiation				
170-17Ch	16	N/A	Reserved				
180h	4	CONFIGFLAG	Configured Flag Register		÷	÷	÷
184h	4	PORTSC1	Port Status/Control 1	÷	÷	÷	÷
188h	4	PORTSC2	Port Status/Control 2				÷
…	4	PORTSCx	Port Status/Control x				÷
1A0h	4	PORTSC8	Port Status/Control 8				÷
1A4h	4	OTGSC	On-The-Go (OTG) Status and Control		÷		
1A8h	4	USBMODE	USB Device Mode	÷	÷	÷	÷
1ACh	4	ENPDTSETUPSTAT	Endpoint Setup Status	÷	÷		
1B0h	4	ENDPTPRIME	Endpoint Initialization	÷	÷		

Table continues on the next page...

Table 56-3. Device/Host Capability Registers (continued)

Offset	Size (bytes)	Mnemonic	Register Name	DEV	OTG	SPH	MPH
1B4h	4	ENDPTFLUSH	Endpoint De-Initialize	÷	÷		
1B8h	4	ENDPTSTATUS	Endpoint Status	÷	÷		
1BCh	4	ENDPTCOMPLETE	Endpoint Complete	÷	÷		
1C0h	4	ENDPTCTRL0	Endpoint Control 0	÷	÷		
1C4h	4	ENDPTCTRL1	Endpoint Control 1	÷	÷		
…	4	ENDPTCTRLx	Endpoint Control x	÷	÷		
1FCh	4	ENDPTCTRL15	Endpoint Control 15	÷	÷		

NOTE

Italic Text indicates a deviation from EHCI for Device

56.4.2.2 Summary of register layouts

Table 56-4. HS-USB register summary

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h ID	ID																															
004h HWGENERAL	reserved																				HWGENERAL										
008h HWHOST	TTPER								TTASY								reserved												NPORT			HC
00Ch HWDEVICE	reserved																										DEVP					DC
010h HWTXBUF	1	reserved							TXCHANADD								TXADD										TXBURST					
014h HWRXBUF	reserved																RXADD										RXBURST					
020h Reserved	reserved																															
Reserved	reserved																															
080h GPTIMER0LD	reserved								GPTLD																							
084h GPTIMER0CTRL	GPTRUN	GPTRS	reserved					GPTMODE	GPTCNT																							
088h GPTIMER1LD	reserved								GPTLD																							
08ch GPTIMER1CTRL	GPTRUN	GPTRS	reserved					GPTMODE	GPTCNT																							
090h SBUSCFG	reserved																														AHBBRST	

Table continues on the next page...

Table 56-4. HS-USB register summary (continued)

Reserved	reserved																					
100h CAPLENGTH		CAPLENGTH																				
101h Reserved		reserved																				
102h HCIVERSION		HCIVERSION																				
104h HCSPARAMS	reserved	N_TT	N_PTT	reserved	PI	N_CC	N_PCC	reserved	PPC	N_PORTS												
108h HCCPARAMS	reserved	EEC{[7:0]	IST[7:4]	R	R	PF	ADC															
10Ch Reserved	reserved																					
Reserved	reserved																					
11Fh Reserved	reserved																					
120h DCIVERSION		DCIVERSION																				
122h Reserved		reserved																				
124h DCCPARAMS	reserved	HC	DC	R	R	DEN																
128h Reserved	reserved																					
Reserved	reserved																					
13Ch Reserved	reserved																					
140h USBCMD	reserved	ITC	FS	ATDTW	SUTW	R	ASP	R	ASP1	ASP0	L	IAA	AS	PS	FS	FS	RS	R S				
144h USBSTS	reserved	TI1	TI0	reserved	UP I	UA I	R	NAKI	A	P	RCL	HCH	reserved	SL	SR	URI	AAI	SE	FR	PCI	UE	UI
148h USBINTR	reserved	TIE1	TIE0	reserved	UP IE	UA IE	R	NAKE	reserved	SL	SR	UR	AAE	SE	FR	PC	UE	U E				
14Ch FRINDEX	reserved	FRINDEX[13:0]																				
150h Reserved	reserved																					
154h PERIODICLISTBASE	PERBASE[31:12]	reserved																				
Device Addr	USBADR[31:25]	USBADRA	reserved																			
158h ASYNCLISTADDR	ASYBASE[31:5]	reserved																				
Endpointlist Addr	EPBASE[31:11]	reserved																				
15Ch ASYNCTTSTS	reserved	TTAC	TTAS																			

Table continues on the next page...

Table 56-4. HS-USB register summary (continued)

160h BURSTSIZE	reserved																TXPBURST								RXPBURST							
164h TXFILLTUNING	reserved										TXFIFOTHRES						R			TXSCHHEALTH					TXSCHOH							
168h TXTTFILLTUNING	reserved																			TXTTSCHHEALTH					R		TXTTSCHOH					
16Ch IC_USB	IC8	IC_VDD8			IC7	IC_VDD7			IC6	IC_VDD6			IC5	IC_VDD5			IC4	IC_VDD4			IC3	IC_VDD3			IC2	IC_VDD2			IC1	IC_VDD1		
170 ULPI Viewport																																
174 Reserved	reserved																															
Reserved	reserved																															
17Ch Reserved	reserved																															
180h CONFIGFLAG	set to zero																															1
184h PORTSC1	PTS		ST	PTW	PSPD		PTS2	PFS	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	P	LS		HS	P	SUS	FP	OCC	OCA	PE	P	CSC	CCS
188h PORTSC2	PTS		ST	PTW	PSPD		PTS2	PFS	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	P	LS		HS	P	SUS	FP	OCC	OCA	PE	P	CSC	CCS
PORTSCx	PTS		ST	PTW	PSPD		PTS2	PFS	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	P	LS		HS	P	SUS	FP	OCC	OCA	PE	P	CSC	CCS
1A0h PORTSC8	PTS		ST	PTW	PSPD		PTS2	PFS	PHCD	WKOC	WKDS	WKCN	PTC				PIC		PO	P	LS		HS	P	SUS	FP	OCC	OCA	PE	P	CSC	CCS
1A4h OTGSC	R	DPI	1 msE	BSEI	BSVI	ASVIE	AVVIE	ID IE	R	DPI	1 msS	BSEI	BSVI	ASVIS	AVVIS	ID IS	R	DP	1 msT	BS	BS	ASV	AVV	ID	reserved			DP	OT	R	VC	VD
1A8h USBMODE	reserved																SR	Reserved										SDI	SLOM	E	CM	
1ACh ENPDTSETUPSTAT	reserved																ENDPTSETUPSTAT															
1B0h ENDPTPRIME	PETB[15:0]																PERB[15:0]															
1B4h ENDPTFLUSH	FETB[15:0]																FERB[15:0]															
1B8h ENDPTSTATUS	ETBR[15:0]																ERBR[15:0]															
1BCh ENDPTCOMPLETE	ETCE[15:0]																ERCE[15:0]															
1C0h ENDPTCTRL0	reserved								TX	reserved			TXT		R	TX	reserved								RX	reserved			RXT		R	RX S
1C4h ENDPTCTRL1	reserved								TX	TXR	TXI	R	TXT		TXD	TX	reserved								RX	RX	RX	R	RXT		RXD	RX S
ENDPTCTRLx	reserved								TX	TXR	TXI	R	TXT		TXD	TX	reserved								RX	RX	RX	R	RXT		RXD	RX S

Table continues on the next page...

Table 56-4. HS-USB register summary (continued)

1FCh ENDPTCTRL15	reserved	TX	TXR	TXI	R	TXT	TXD	TX	reserved	RX	RX	RX	R	RXT	RXD	RX	S

56.4.2.3 Identification Registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

56.4.2.4 Device/Host Capability Registers

Device/Host Capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation.

56.4.2.5 Device/Host Timer Registers (Non-EHCI)

The host/device controller drivers can measure time related activities using these timer registers.

These registers are not part of the standard EHCI controller.

56.4.2.6 Device/Host Operational Registers

Operational registers are comprised of dynamic control or status registers that may be read only, read/write, or read/write to clear. The following sections define the use of these registers.

56.4.2.7 OTG Operations

56.4.2.7.1 Register Bits

In the previous section, the Register interface has behaviors described for device mode and behaviors described for host mode. However, during OTG operations it is necessary to perform tasks independent of the controller mode.

Note also from Endpoint Controln (USB_ENDPTCTRL*n*) that the only way to transition the controller mode out of host or device mode is with the controller reset bit. Therefore, it is also necessary for the OTG tasks to be performed independent of a controller reset as well as independent of the controller mode.

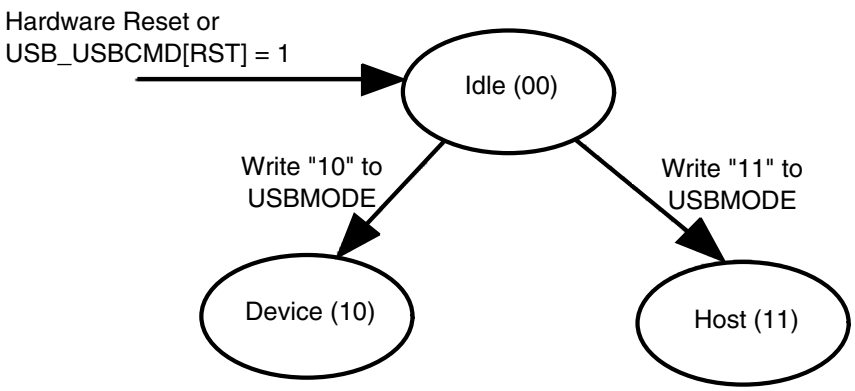


Figure 56-5. Controller Mode

To this end, the listed below are the register bits that are used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USBCMD register:

All Identification Registers

All Device/Host Capability Registers

OTGSC: All bits

PORTSC:

Physical Interface Select

Physical Interface Serial Select

Physical Interface Data Width

Physical Interface Low Power

Physical Interface Wake Signals

Port Indicators

Port Power

56.4.2.7.2 Hardware Assist

The hardware assist provides automated response and sequencing that may not be possible to software with significant interrupt latency response times. The use of this additional circuitry is optional and can be used to assist the 3 sequences below.

56.4.2.7.2.1 Auto-Reset

When the HAAR is set to one, the host will automatically start a reset after a connect event. This shortcuts the normal process where software is notified of the connect event and starts the reset. Software will still receive notification of the connect event but should not write the reset bit when the HAAR is set. Software will be notified again after the reset is complete via the enable change bit in the PORTSC register which cause a port change interrupt.

This assist will ensure the OTG parameter TB_ACON_BSE0_MAX = 1ms is met.

56.4.2.7.2.2 Data-Pulse

Writing a one to HADP will start a data pulse of approximately 7ms in duration and then automatically cease the data pulsing. During the data pulse, the DP will be set and then cleared. This automation relieves software from accurately controlling the data-pulse duration. During the data pulse, the HCD can poll to see that the HADP and DP bit have returned low to recognize the completion or simply launch the data pulse and wait to see if a VBUS Valid interrupt occurs when the A-side supplies bus power.

This assist will ensure data pulsing meets the OTG requirement of > 5ms and < 10ms.

56.4.2.7.2.3 B-Disconnect to A-Connect

During HNP, the B-disconnect occurs from the OTG A_suspend state and within 3 ms, the A device must enable the pullup on the DP leg in the A-peripheral state. When HABA is set, the Host Controller port is in suspend mode, and the device disconnects, then this hardware assist begins.

1. Reset the OTG core.
2. Write the OTG core into device mode.
3. Write the device run bit to a 1 and enable necessary interrupts including:

USB Reset Enable (URE) ; enables interrupt on usb bus reset to device

Sleep Enable (SLE) ; enables interrupt on device suspend

Port Change Detect Enable (PCE) ; enables interrupt on device connect

When software has enabled this hardware assist, it must not interfere during the transition and should not write any register in the core until it gets an interrupt from the device controller signifying that a reset interrupt has occurred or at least first verify that the core has entered device mode. HCD/DCD must not activate the core soft reset at any time since this action is performed by hardware. During the transition, the software may see an interrupt from the disconnect and/or other spurious interrupts (ie. SOF/etc.) that may or may not cascade and my be cleared by the soft reset depending on the software response time.

After the core has entered device mode by the hardware assist, the DCD must ensure that the ENDPTLISTADDR is programmed properly before the host sends a setup packet. Since the end of the reset duration, which may be initiated quickly (a few microseconds) after connect, will require at a minimum 50 ms, this is the time for which the DCD must be ready to accept setup packets after having received notification that the reset has been detected or simply that the OTG is in device mode which ever occurs first.

In the case where the A-peripheral fails to see a reset after the controller enters device mode and engages the DP-pullup, the device controller interrupt the DCD signifying that a suspend has occurred.

This assist will ensure the parameter TA_BDIS_ACON_MAX = 3ms is met.

56.4.3 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware). The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of a Periodic Schedule, Periodic Frame List, Asynchronous Schedule, Isochronous Transaction Descriptors, Split-transaction Isochronous Transfer Descriptors, Queue Heads, and Queue Element Transfer Descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) support for the host controller interface. The asynchronous list is the root for all the bulk and control transfer type support. Isochronous data streams are managed using Isochronous Transaction Descriptors. Isochronous split-transaction data streams are managed with Split-transaction Isochronous Transfer Descriptors. All Interrupt, Control, and Bulk data streams are managed via queue heads and Queue Element Transfer Descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Note that software must ensure that no interface data structure reachable by the EHCI host controller spans a 4K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writeable fields. The host controller must preserve the read-only fields on all data structure writes.

56.4.3.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the *PERIODICLISTBASE* address register and the *FRINDEX* register. The periodic schedule is based on an array of pointers called the Periodic Frame List. The *PERIODICLISTBASE* address register is combined with the *FRINDEX* register to produce a memory pointer into the frame list. The Periodic Frame List implements a *sliding window* of work over time.

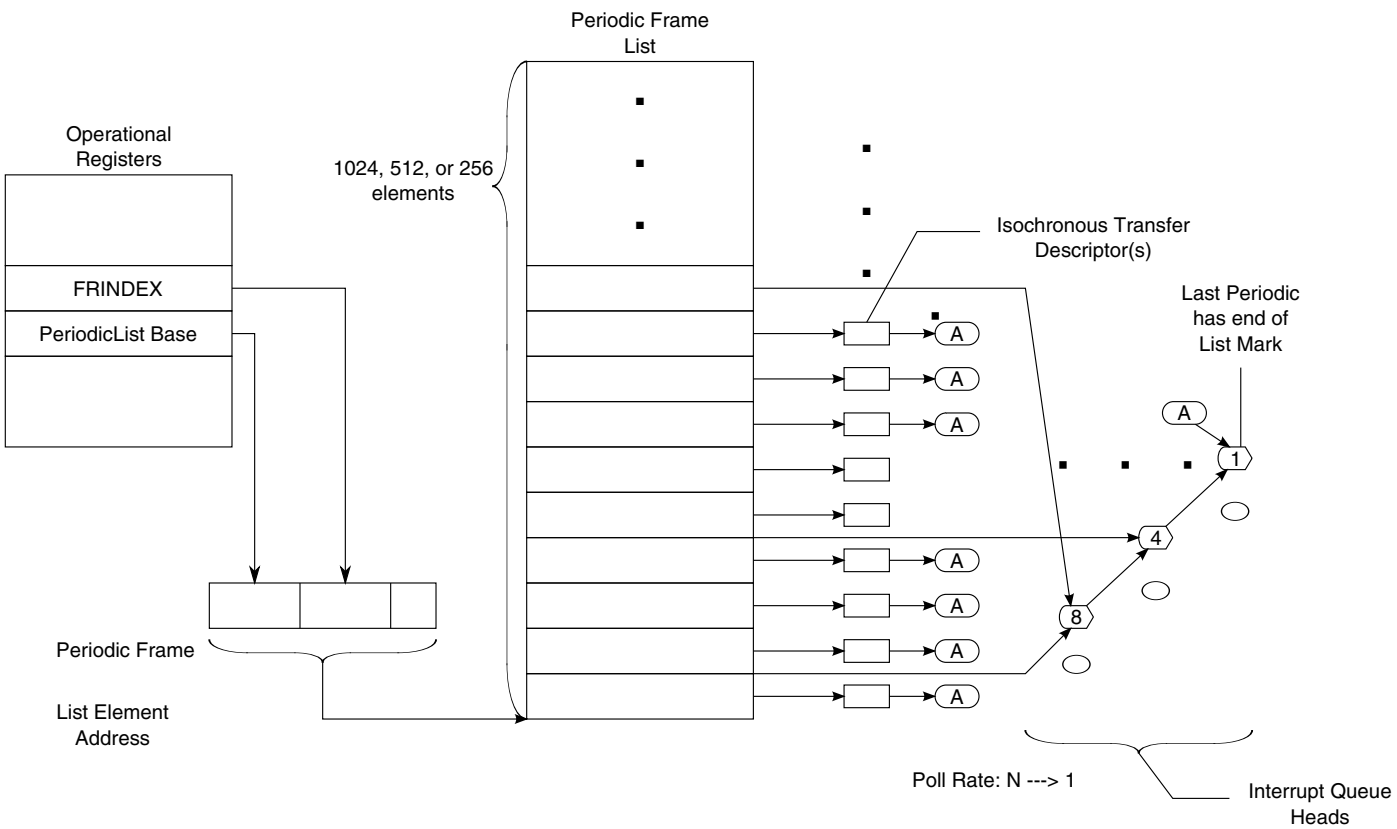


Figure 56-6. Periodic Schedule Organization

NOTE

[1] Split transaction Interrupt, Bulk and Control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4K-page aligned array of Frame List Link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1024 elements. An implementation must support all three sizes. Programming the size (i.e. the number of elements) is accomplished by system software writing the appropriate value into *Frame List Size* field in the USBCMD register.

Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro-frame. The link pointers are aligned on DWord boundaries within the Frame List.

Table 56-5. Format of Frame List Element Pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Frame List Link Pointer																											0		Typ			03-00H

Frame List Link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing.

The least significant bit is the *T-Bit* (bit 0). When this bit is set to a one, the host controller will never use the value of the frame list pointer as a physical memory pointer. The *Typ* field is used to indicate the exact type of data structure being referenced by this pointer. The value encodings are:

Table 56-6. Typ Field Value Definitions

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor.
11b	Frame Span Traversal Node.

56.4.3.2 Asynchronous List Queue Head Pointer

The Asychronous Transfer List (based at the ASYNCLISTADDR register) is where all the control and bulk transfers are managed. Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.

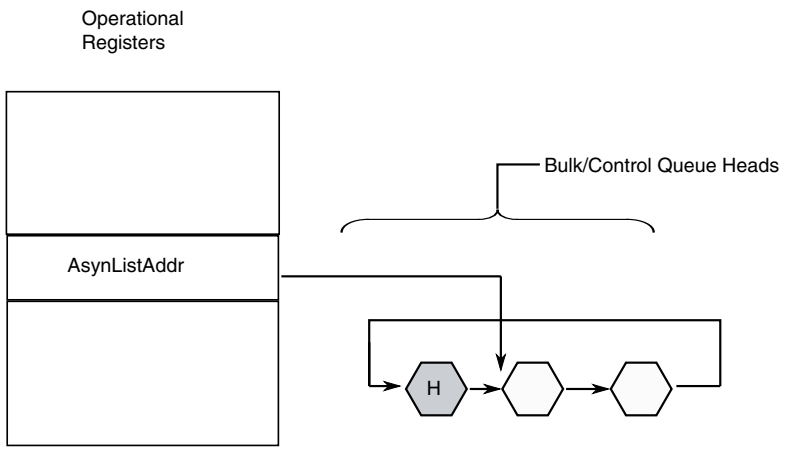


Figure 56-7. Asynchronous Schedule Organization

The Asynchronous list is a simple circular list of queue heads. The ASYNCLISTADDR register is simply pointer to the *next* queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

56.4.3.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

The format of an isochronous transfer descriptor is illustrated in Table 56-7. This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary.

Table 56-7. *Isochronous Transaction Descriptor (iTD)

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

Table continues on the next page...

Table 56-7. *Isochronous Transaction Descriptor (iTD) (continued)

Next Link Pointer				0	Typ	T	03-00H
Status	Transaction 0 Length	ioc	PG*	Transaction 0 Offset*			07-04H
Status	Transaction 1 Length	ioc	PG*	Transaction 1 Offset*			0B-08H
Status	Transaction 2 Length	ioc	PG*	Transaction 2 Offset*			0F-0CH
Status	Transaction 3 Length	ioc	PG*	Transaction 3 Offset*			13-10H
Status	Transaction 4 Length	ioc	PG*	Transaction 4 Offset*			17-14H
Status	Transaction 5 Length	ioc	PG*	Transaction 5 Offset*			1B-18H
Status	Transaction 6 Length	ioc	PG*	Transaction 6 Offset*			1F-1CH
Status	Transaction 7 Length	ioc	PG*	Transaction 7 Offset*			23-20H

Table continues on the next page...

Table 56-7. *Isochronous Transaction Descriptor (iTD) (continued)

Buffer Pointer (Page 0)	EndPt	R	Device Address	27-24H
Buffer Pointer (Page 1)	I/O	Maximum Packet Size		2B-28H
Buffer Pointer (Page 2)	Reserved		Mult	2F-2CH
Buffer Pointer (Page 3)	Reserved			33-30H
Buffer Pointer (Page 4)	Reserved			37-34H
Buffer Pointer (Page 5)	Reserved			3B-38H
Buffer Pointer (Page 6)	Reserved			3F-3CH

Shaded: Host Controller Read/Write

Non-Shaded: Host Controller Read Only.

NOTE

These fields may be modified by the host controller if the I/O field indicates an OUT.

56.4.3.3.1 iTD Next Link Pointer

The first DWord of an iTD is a pointer to the next schedule data structure.

Table 56-8. Next Schedule Element Pointer

Bit	Description
31:5	Link Pointer (LP). These bits correspond to memory address signals [31:5], respectively. This field points to another Isochronous Transaction Descriptor (iTD/siTD) or Queue Head (QH).
4:3	Reserved. These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.
2:1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1= Link Pointer field is not valid. 0= Link Pointer field is valid.

56.4.3.3.2 iTD Transaction Status and Control List

DWords 1 through 8 are eight slots of transaction control and status. Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The *PG* and *Transaction X Offset* fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three DWords of the Buffer Page Pointer list, to execute a transaction on the USB.

Table 56-9. iTD Transaction Status and Control

Bit	Description
31:28	Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding: 31 - Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule. 30 - Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary. 29 - Babble Detected. Set to one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor. 28 - Transaction Error (XactErr). Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit may only be set for isochronous IN transactions.
27:16	Transaction X Length. For an OUT, this field is the number of data bytes the host controller will send during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (e.g. 0àzero length data, 1àone byte, 2àtwo bytes, etc.). The maximum value this field may contain is 0xC00 (3072).
15	Interrupt On Complete (IOC). If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.
14:12	Page Select (PG). These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.
11:0	Transaction X Offset. This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent *PG* field to produce the starting buffer address for this transaction.

56.4.3.3.3 iTD Buffer Page Pointer List (Plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous. Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page.

Since each pointer is a 4K aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

Table 56-10. iTD Buffer Pointer Page 0 (Plus)

Bit	Description
31:12	Buffer Pointer (Page 0). This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit reserved for future use and should be initialized by software to zero.
6:0	Device Address. This field selects the specific device serving as the data source or sink.

Table 56-11. iTD Buffer Pointer Page 1 (Plus)

Bit	Description
31:12	Buffer Pointer (Page 1). This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11	Direction (I/O). 0 = OUT; 1 = IN. This field encodes whether the high-speed transaction should use an IN or OUT PID.
10:0	Maximum Packet Size. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (.e.g. per micro-frame). This field is used with the *Multi* field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results.

Table 56-12. iTD Buffer Pointer Page 2 (Plus)

Bit	Description
31:12	Buffer Pointer. This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:2	Reserved. This bit reserved for future use and should be set to zero.
1:0	Multi. This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (e.g. per micro-frame). The valid values are: Value Meaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro- frame 10b Two transactions to be issued for this endpoint per micro- frame 11b Three transactions to be issued for this endpoint per micro- frame

Table 56-13. iTD Buffer Pointer Page 3-6

Bit	Description
31:12	Buffer Pointer. This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11:0	Reserved. These bits reserved for future use and should be set to zero.

56.4.3.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All Full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.

Table 56-14. Split-transaction Isochronous Transaction Descriptor (siTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Next Link Pointer																											0		Typ		T	03-00H
I/O	Port Number							R	Hub Addr							R	EndPt			R	Device Address											07-04H
Reserved																μFrame C-mask				μFrame S-mask												0B-08H
ioc	P	Reserved						Total Bytes to Transfer								μFrame C-prog-mask				Status													0F-0CH
Buffer Pointer (Page 0)																				Current Offset												13-10H
Buffer Pointer (Page 1)																				Reserved							TP		T-count			17-14H
Back Pointer																											0				T	1B-18H

Shaded: Host Controller Read/Write

Non-Shaded: Host Controller Read Only.

56.4.3.4.1 siTD Next Link Pointer

DWord0 of a siTD is a pointer to the next schedule data structure.

Table 56-15. Next Link Pointer

Bit	Description
31:5	Next Link Pointer (LP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as zeros.
2:1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid.

56.4.3.4.2 siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

Table 56-16. Endpoint and Transaction Translator Characteristics

Bit	Description
31	Direction (I/O).0 = OUT; 1 = IN. This field encodes whether the full-speed transaction should be an IN or OUT.
30:24	Port Number. This field is the port number of the recipient Transaction Translator.
23	Reserved. Bit reserved and should be set to zero.
22:16	Hub Address. This field holds the device address of the Companion Controllers' hub.
15:12	Reserved. Field reserved and should be set to zero.
11:8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit is reserved for future use. It should be set to zero.
6:0	Device Address. This field selects the specific device serving as the data source or sink.

Table 56-17. Micro-frame Schedule Control

Bit	Description
31:16	Reserved. This field reserved for future use. It should be set to zero.

Table continues on the next page...

Table 56-17. Micro-frame Schedule Control (continued)

Bit	Description
15:8	Split Completion Mask (µFrame C-Mask). This field (along with the *Active* and *SplitX- state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the µ*Frame C-Mask* field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7:0	Split Start Mask (µFrame S-mask). This field (along with the *Active* and *SplitX-state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the µ*Frame S-mask* field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

56.4.3.4.3 siTD Transfer State

DWords 3-6 are used to manage the state of the transfer.

Table 56-18. siTD Transfer Status and Control

Bit	Description
31	Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it will assert a hardware interrupt at the next interrupt threshold.
30	Page Select (P). Used to indicate which data page pointer should be concatenated with the *CurrentOffset* field to construct a data buffer pointer (0 selects *Page 0* pointer and 1 selects *Page 1*). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
29:26	Reserved. This field reserved for future use and should be set to zero.
25:16	Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)
15:8	µFrame Complete-split Progress Mask (C-prog-Mask). This field is used by the host controller to record which split-completes has been executed.
7:0	Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:
7	Active. Set to one by software to enable the execution of an isochronous split transaction by the Host Controller.
6	ERR. Set to a one by the Host Controller when an ERR response is received from the Companion Controller.
5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the Host Controller will transmit an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.
4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.

Table continues on the next page...

Table 56-18. siTD Transfer Status and Control (continued)

Bit	Description
3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit will only be set for IN transactions.
2	Missed Micro-Frame. The host controller detected that a host-induced hold- off caused the host controller to miss a required complete-split transaction.
1	Split Transaction State (SplitXstate). The bit encodings are: Value Meaning 00b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask. 01b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.
0	Reserved. Bit reserved for future use and should be set to zero.

56.4.3.4.4 siTD Buffer Pointer List (plus)

DWords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each DWord in this section are the 4K (page) aligned buffer pointers. The least significant 12 bits of each DWord are used as additional transfer state.

Table 56-19. Buffer Page Pointer List (plus)

Bit	Description
31:12	Buffer Pointer List. Bits [31:12] of DWords 4 and 5 are 4K paged aligned, physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field *P* specifies the *current* active pointer
11:0	Page 0: Current Offset. The 12 least significant bits of the Page 0 pointer is the current byte offset for the current page pointer (as selected with the page indicator bit (*P* field)). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero). The least significant bits of Page 1 pointer is split into three sub-fields Page 1:
11:5	Reserved.

Table continues on the next page...

Table 56-19. Buffer Page Pointer List (plus) (continued)

Bit	Description
4:3	Transaction position (TP). This field is used with T-count to determine whether to send *all*, *first*, *middle*, or *last* with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are: Value Meaning 00b All. The entire full-speed transaction data payload is in this transaction (i.e. less than or equal to 188 bytes). 01b Begin. This is the first data payload for a full- speed that is greater than 188 bytes.transaction 10B Mid. This is the *middle* payload for a full-speed OUT transaction that is larger than 188 bytes. 11b End. This is the *last* payload for a full-speed OUT transaction that was larger than 188 bytes.
2:0	Transaction count (T-Count). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

56.4.3.4.5 siTD Back Link Pointer

DWord 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD. This pointer cannot reference any other schedule data structure.

Table 56-20. siTD Back Link Pointer

Bit	Description
31:5	siTD Back Pointer. This field is a physical memory pointer to a siTD.
4:1	Reserved. This field is reserved for future use. It should be set to zero.
0	Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid.

56.4.3.5 Queue Element Transfer Descriptor (qTD)

This data structure is only used with a queue head. This data structure is used for one or more USB transactions. This data structure is used to transfer up to 20480 (5*4096) bytes. The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five-element array of data buffer pointers. This structure is 32 bytes (or one 32-byte cache line). This data structure must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

Table 56-21. Queue Element Transfer Descriptor Block Diagram

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Next qTD Pointer																											0				T	03-00H
Alternate Next qTD Pointer																											0				T	07-04H
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code	Status									0B-08H
Buffer Pointer (page 0)																				Current Offset												0F-0CH
Buffer Pointer (page 0)																				Reserved												13-10H
Buffer Pointer (page 0)																				Reserved												17-14H
Buffer Pointer (page 0)																				Reserved												1B-18H
Buffer Pointer (page 0)																				Reserved												1F-1CH

Shaded: Host Controller Read/Write

Non-Shaded: Host Controller Read Only.

NOTE

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

56.4.3.5.1 Next qTD Pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

Table 56-22. qTD Next Element Transfer Pointer (DWord 0)

Bit	Description
31:5	Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.
4:1	Reserved. These bits are reserved and their value has no effect on operation.
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

56.4.3.5.2 Alternate Next qTD Pointer

The second DWord of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next client buffer on short packet. To be more explicit the host controller will always use this pointer when the current qTD is retired due to short packet.

Table 56-23. TD Alternate Next Element Transfer Pointer (DWord 1)

Bit	Description
31:5	Alternate Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4:1	Reserved. These bits are reserved and their value has no effect on operation.
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

56.4.3.5.3 qTD Token

The third DWord of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

NOTE

The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation

Table 56-24. qTD Token (DWord 2)

Bit	Description
31	Data Toggle. This is the data toggle sequence bit. The use of this bit depends on the setting of the *Data Toggle Control* bit in the queue head.
30:16	Total Bytes to Transfer. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 * 4K (5000H). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that *Total Bytes To Transfer* be an even multiple of QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction will always be less than QHD.Maximum Packet Length. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K(4000H).
15	Interrupt On Complete (IOC). If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.
14:12	Current Page (C_Page). This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0H to 4H. The host controller is not required to write this field back when the qTD is retired.
11:10	Error Counter (CERR). This field is a 2-bit down counter that keeps track of the number of consecutive Errors detected while executing this qTD. If this field is programmed with a non-zero value during set-up, the Host Controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the Host Controller marks the qTD inactive, sets the *Halted* bit to a one, and error status bit for the error that caused *CERR* to decrement to zero. An interrupt will be generated if the *USB Error Interrupt Enable* bit in the USBINTR register is set to a one. If HCD programs this field to zero during set-up, the Host Controller will not count errors for this qTD and there will be no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD. Error Decrement Counter Transaction Error Yes Data Buffer Error No[3] Stalled No[1] Babble Detected No[1] No Error No[2]
-	Below is the Error Count and description of the Decrement Counter Error Decrement Counter: 1 - Detection of Babble or Stall automatically halts the queue head. Thus, count is not decremented 2 - If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and *PIDCode* indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset *CERR* to extend the total number of errors for this transaction. For example, *CERR* should be reset with maximum value (3) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 00b. See section Split Transaction Interrupt for CERR adjustment rules when the EPS field indicates a FS or LS device and the queue head is in the Periodic Schedule. See section Asynchronous-Do Complete Split for CERR adjustment rules when the EPS field indicates a FS or LS device, the queue head is in the Asynchronous schedule and the *PIDCode* indicates a SETUP. 3 - Data buffer errors are host problems. They don't count against the device's retries.

Table continues on the next page...

Table 56-24. qTD Token (DWord 2) (continued)

Bit	Description
-	Note: Software must not program CERR to a value of zero when the EPS field is programmed with a value indicating a Full- or Low-speed device. This combination could result in undefined behavior.
9:8	PID Code. This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:
-	00b - OUT Token generates token (E1H) 01b - IN Token generates token (69H) 10b - SETUP Token generates token (2DH) (undefined if endpoint is an Interrupt the queue head is non-zero.) transfer type, e.g. µ*Frame S-mask* field in 11b - Reserved

Table continues on the next page...

Table 56-24. qTD Token (DWord 2) (continued)

Bit	Description
7:0	Status. This field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. The bit encodings are: 7 - Active. Set to one by software to enable the execution of transactions by the Host Controller. 6 - Halted. Set to a one by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero. 5 - Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the Host Controller will force a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer. 4 - Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction. In addition to setting this bit, the Host Controller also sets the *Halted* bit to a one. Since "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor. 3 - Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer. 2 - Missed Micro-Frame. This bit is ignored unless the *QH.EPS* field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer. 1 - Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the *QH.EPS* field indicates a full- or low-speed endpoint. When a Full- or Low-speed device, the host controller uses this bit to track the state of the split- transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are: Value Meaning 0b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. 1b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint. 0 - Ping State (P)/ERR. If the *QH.EPS* field indicates a High-speed device and the *PID_Code* indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are: Value Meaning 0b Do OUT. This value directs the host controller to issue an OUT PID to the endpoint. 1b Do Ping. This value directs the host controller to issue a PING PID to the endpoint. If the *QH.EPS* field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.

56.4.3.5.4 qTD Buffer Page Pointer List

The last five DWords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes *Current Offset* field to the starting offset into the current page, where current page is selected via the value in the *C_Page* field.

Table 56-25. qTD Buffer Pointer(s) (DWords 3-7)

Bit	Description
31:12	Buffer Pointer List. Each element in the list is a 4K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction via the new buffer pointer.
11:0	Current Offset (Reserved). This field is reserved in all pointers except the first one (e.g. Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the Reserved fields are initialized to zeros.

56.4.3.6 Queue Head

Table 56-26. Queue Head Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Queue Head Horizontal Link Pointer																											0		Typ		T	03-00H
RL			C	Maximum Packet Length												H	dtc	EP		EndPt			I	Device Address								07-04H
Mult		Port Number*								Hub Addr*						μFrame C-mask*								μFrame S-mask*								0B-08H
Current qTD Pointer																													0			0F-0CH

Table continues on the next page...

Table 56-26. Queue Head Structure Layout (continued)

Next qTD Pointer						0	T	13-10H
Alternate Next qTD pointer						NakCnt	T	17-14H
dt	Total Bytes to Transfer	ioc	C_Page	Cerr	PID Code	Status		1B-18H
Buffer Pointer (Page 0)				Current Offset				1F-1CH
Buffer Pointer (Page 1)				Reserved	C-prog-mask*			23-20H
Buffer Pointer (Page 2)				S-bytes*		FrameTag*		27-24H
Buffer Pointer (Page 3)				Reserved				2B-28H
Buffer Pointer (Page 4)				Reserved				2F-2CH

Transfer Overlay Transfer Results

Static Endpoint State

*These fields are used exclusively to support split transactions to USB 2.0 Hubs

Shaded: Host Controller Read/Write

Non-Shaded: Host Controller Read Only.

Queue Head Horizontal Link Pointer

The first DWord of a Queue Head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

Table 56-27. Queue Head DWord 0

Bit	Description
31:5	Queue Head Horizontal Link Pointer (QHLP). This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as zeros.
2:1	QH/(s)iTD Select (Typ). This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Last QH (pointer is invalid). 0=Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the Asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

56.4.3.6.1 Queue Head Endpoint Capabilities/Characteristics

The second and third DWords of a Queue Head specifies static information about the endpoint. This information does not change over the lifetime of the endpoint. There are three types of information in this region:

- Endpoint Characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.
- Endpoint Capabilities. These are adjustable parameters of the endpoint. They effect how the endpoint data stream is managed by the host controller.
- Split Transaction Characteristics. This data structure is used to manage full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

Table 56-28. Endpoint Characteristics: Queue Head DWord 1

Bit	Description
31:28	Nak Count Reload (RL). This field contains a value, which is used by the host controller to reload the Nak Counter field.
27	Control Endpoint Flag (C). If the *QH.EPS* field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to a zero.
26:16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). The maximum value this field may contain is 0x400 (1024).
15	Head of Reclamation List Flag (H). This bit is set by System Software to mark a queue head as being the head of the reclamation list.
14	Data Toggle Control (DTC). This bit specifies where the host controller should get the initial data toggle on an overlay transition. 0b Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1b Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.
13:12	Endpoint Speed (EPS). This is the speed of the associated endpoint. Bit combinations are: 00b - Full-Speed (12Mbs) 01b - Low-Speed (1.5Mbs) 10b - High-Speed (480 Mb/s) 11b - Reserved This field must not be modified by the host controller.
11:8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Inactivate on Next Transaction (I). This bit is used by system software to request that the host controller set the Active bit to zero. See Section Rebalancing the Periodic Schedule for full operational details. This field is only valid when the queue head is in the Periodic Schedule and the *EPS* field indicates a Full or Low-speed endpoint. Setting this bit to a one when the queue head is in the Asynchronous Schedule or the *EPS* field indicates a high-speed device yields undefined results.
6:0	Device Address. This field selects the specific device serving as the data source or sink.

Table 56-29. Endpoint Capabilities: Queue Head DWord 2

Bit	Description
31:30	High-Bandwidth Pipe Multiplier (Mult). This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). The valid values are: Value Meaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro-frame 10b Two transactions to be issued for this endpoint per micro-frame 11b Three transactions to be issued for this endpoint per micro-frame

Table continues on the next page...

Table 56-29. Endpoint Capabilities: Queue Head DWord 2 (continued)

Bit	Description
29:23	Port Number. This field is ignored by the host controller unless the *EPS* field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 Hub (for hub at device address *Hub Addr* below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22:16	Hub Addr. This field is ignored by the host controller unless the *EPS* field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 Hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.
15:8	Split Completion Mask (µFrame C-Mask). This field is ignored by the host controller unless the *EPS* field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the *Active* and *SplitX-state* fields) is used to determine during which micro-frames the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the µ*Frame C- Mask* field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7:0	Interrupt Schedule Mask (µFrame S-mask). This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the µ*Frame S-mask* field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the *EPS* field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the *PID_Code* field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the *EPS* field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

56.4.3.6.2 Queue Head Transfer Overlay

The nine DWords in this area represent a *transaction working space* for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the *Queue Head Horizontal Link Pointer* to the next queue head. The host controller will never follow the *Next Transfer Queue Element or Alternate Queue Element* pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a Queue Head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

Table 56-30. Current qTD Link Pointer

Bit	Description
31:5	Current Element Transaction Descriptor Link Pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4:0	Reserved (R). These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The DWords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a Queue Element Transfer Descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an *overlay* because when the queue is advanced to the next queue element, the source queue element is *merged* onto this area. This area serves an execution cache for the transfer.

Table 56-31. Host-Controller Rules for Bits in Overlay (DWords 5, 6, 8 and 9)

DWord	Bit	Description
5	4:1	Nak Counter (NakCnt)µRW. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from *RL* before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.
6	31	Data Toggle. The *Data Toggle Control* controls whether the host controller preserves this bit when an overlay operation is performed.
6	15	Interrupt On Complete (IOC). The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	11:10	Error Counter (C_ERR). This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0	Ping State (P)/ERR. If the *EPS* field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	7:0	Split-transaction Complete-split Progress (C-prog-mask). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	4:0	Split-transaction Frame Tag (Frame Tag). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	11:5	S-bytes. Software must ensure that the *S-bytes* field in a *qTD* is zero before activating the *qTD*. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.

56.4.3.7 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full- and Low-speed transactions that span a Host-frame boundary. See section Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous

Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use will yield undefined results.

Table 56-32. Frame Span Traversal Node Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Normal Path Link Pointer																											0		Typ		T	03-00H
Back Path Link Pointer																											0		Typ		T	07-04H

Shaded: Host Controller Read/Write

Non-Shaded: Host Controller Read Only.

56.4.3.7.1 FSTN Normal Path Pointer

The first DWord of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

Table 56-33. FSTN Normal Path Pointer Descriptions

Bit	Description
31:5	Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as 0s.
2:1	QH/(s)iTD/FSTN Select (Typ). This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (Frame Span Traversal Node)
0	Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid.

56.4.3.7.2 FSTN Back Path Link Pointer

The second DWord of an FTSN node contains a link pointer to a queue head. If the *T-bit* in this pointer is a zero, then this FSTN is a *Save-Place* indicator. Its *Typ* field must be set by software to indicate the target data structure is a queue head. If the *T-bit* in this pointer is set to a one, then this FSTN is the *Restore* indicator. When the *T-bit* is a one, the host controller ignores the *Typ* field.

Table 56-34. FSTN Back Path Link Pointer Descriptions

Bit	Description
31:5	Back Path Link Pointer (BPLP). This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.
4:3	Reserved. These bits must be written as 0s.
2:1	Typ. Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	Terminate (T). 1=Link Pointer field is not valid (i.e. the host controller must not use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Restore indicator. 0=Link Pointer is valid (i.e. the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator.

56.4.4 Host Operational Model

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software). Each significant operational feature of the EHCI host controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

56.4.4.1 Host Controller Initialization

When the system boots, the host controller is enumerated, assigned a base address for the register space and BIOS sets the FLADJ register to a system-specific value. After initial power-on or *HCReset* (hardware or via *HCReset* bit in the USBCMD register), all of the operational registers will be at their default values, as illustrated in Table 56-35. After a hardware reset, only the operational registers not contained in the Auxiliary power well will be at their default values.

Table 56-35. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USBCMD	00080000h (00080B00h if *Asynchronous Schedule Park Capability is a one*)
USBSTS	00001000h
USBINTR	00000000h
FRINDEX	00000000h
CTRLDSSEGMENT	00000000h
PERIODICLISTBASE	Undefined
ASYNCLISTADDR	Undefined
CONFIGFLAG	00000000h
PORTSC	00002000h (w/*PPC* set to one); 00003000h (w/*PPC* set to a zero)

In order to initialize the host controller, software should perform the following steps

- Program the CTRLDSSEGMENT register with 4-Gigabyte segment where all of the interface data structures are allocated.
- Write the appropriate value to the USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the PERIODICLIST BASE register. If there are no work items in the periodic schedule, all elements of the Periodic Frame List should have their *T-Bits* set to a one.
- Write the USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller *ON* via setting the *Run/Stop* bit.
- Write a 1 to CONFIGFLAG register to route all ports to the EHCI controller.

At this point, the host controller is up and running and the port registers will begin reporting device connects, etc. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled port enabled High-speed ports, but the schedules have not yet been enabled. The EHCI Host controller will not transmit SOFs to enabled Full- or Low-speed ports. In order to communicate with devices via the asynchronous schedule, system software must write the ASYNDLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing a one to the *Asynchronous Schedule Enable* bit in the USBCMD register. In order to communicate with devices via the periodic schedule, system software must enable the periodic schedule by writing a one to the *Periodic Schedule Enable* bit in the USBCMD register. Note that the schedules can be turned on before the first port is reset (and enabled).

Any time the USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

56.4.4.2 Port Routing and Control

A USB 2.0 Host controller is comprised of one high-speed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers. Companion host controllers (cHCs) may be implementations of either Universal or Open host controller specifications. This configuration is used to deliver the required full USB 2.0-defined port capability; e.g. Low-, Full-, and High-speed capability for every port. Figure 56-8 illustrates a simple block diagram of the port routing logic and its relationship to the high-speed and companion host controllers within a USB 2.0 host controller.

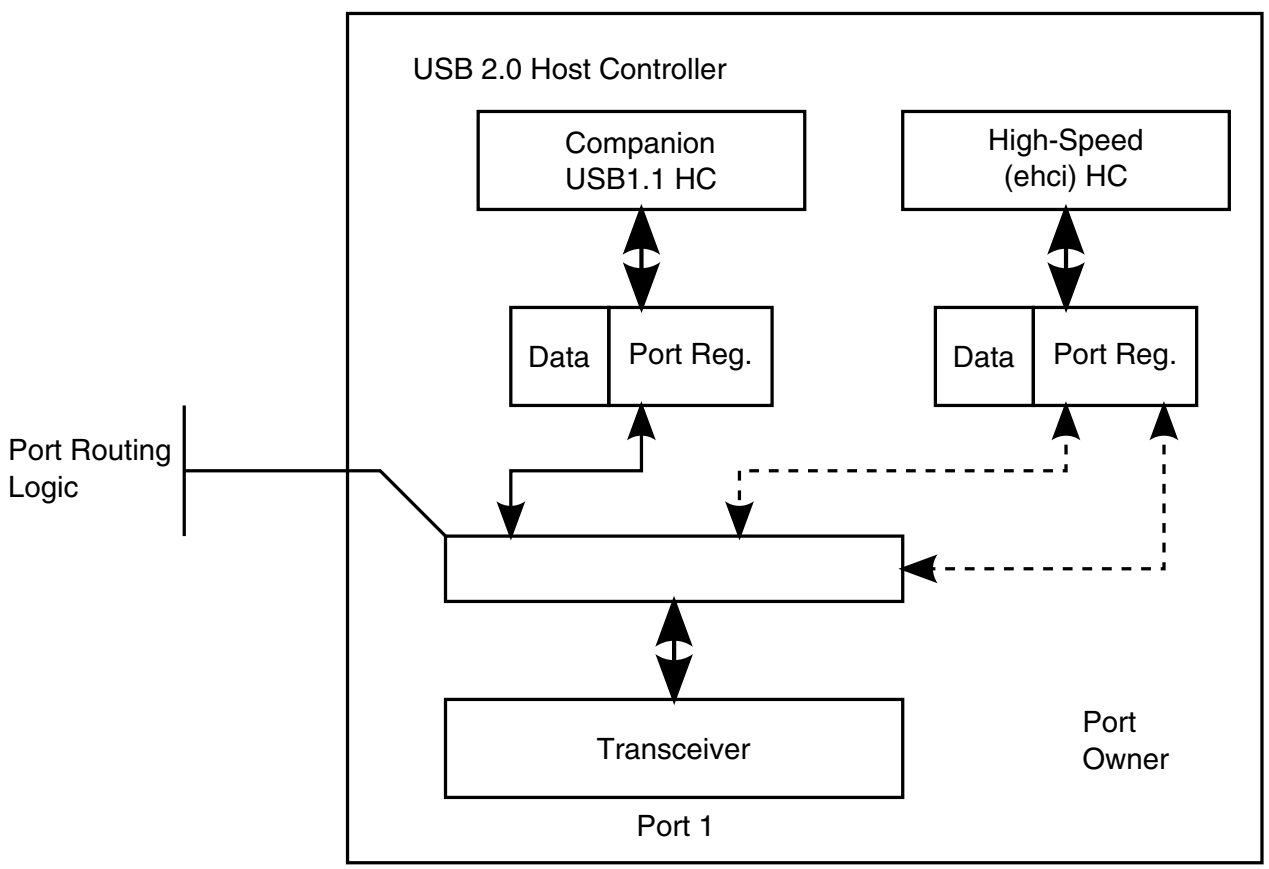


Figure 56-8. Example USB 2.0 Host Controller Port Routing Block Diagram

There exists one transceiver per physical port and each host controller module has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status

registers it is required to operate. Each transceiver can be controlled by either the EHCI host controller or one companion host controller. Routing logic lies between the transceiver and the port status and control registers. The routing logic should not be implemented in the 480 MHz clock domain of the transceiver.

The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a *global* routing policy control field and per-port *ownership* control fields. The *Configured Flag* (*CF*) bit (defined in section Programmable Burst Size (USB_BURSTSIZE)) is the global routing policy control. At power-on or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes Companion Controllers, then the ports will still work in Full- and Low-speed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multi-function PCI device if the implementation includes companion controllers. The companion host controllers' function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (i.e. no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The *N_CC* field in the Structural Parameter register (*HCSPARAMS*) indicates whether the controller implementation includes companion host controllers. When *N_CC* has a non-zero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full- or low-speed devices, the ports will always fail the high-speed chirp during reset and the ports will not be enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a root port to provide full and low-speed device connectivity.

System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers. See Section Host Controller Structural Parameters (USB_HCSPARAMS - EHCI Compliant).If an implementation includes more than one set of companion and EHCI host controllers, they are organized as groups of companion host controllers with intermixed EHCI controllers.

56.4.4.2.1 Port Routing Control via EHCI *Configured* (*CF*) Bit

Each port in the USB 2.0 host controller can be routed either to a single companion host controller or to the EHCI host controller. The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and per-port control. The *Configured Flag* (*CF*) bit (defined in Section Programmable Burst Size (USB_BURSTSIZE)), is used to globally set the policy of the routing logic. Each port register has a *Port Owner* control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the *CF bit* transitions from a zero to a one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all *Port Owner* bits go to zero). While the *CF-bit* is a one, the EHCI Driver can control individual ports' routing via the *Port Owner* control bit. Likewise, whenever the *CF bit* transitions from a one to a zero (as a result of Aux power application, *HCRESET*, or software writing a zero to *CF-bit*), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC's *CF bit* (after Aux power application or *HCRESET*) is zero. Programmable Burst Size (USB_BURSTSIZE) summarizes the default routing for all the ports, based on the value of the EHCI HC's *CF bit*.

The *view* of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

Table 56-36. Default Port Routing Depending on EHCI HC CF Bit

HS CF Bit	Default Port Ownership	Explanation
0B	Companion HCs	The companion host controllers own the ports and only Full- and Low-speed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full- and Low-speed ports in this configuration
1B	EHCI HC	The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC.The EHCI HC has access to the additional port status and control bits defined in this specification (see Section Port Status and Control (USB_PORTSC)). The EHCI HC can temporarily release control of the port to a companion HC by setting the *PortOwner* bit in the PORTSC register to a one.

56.4.4.2.2 Port Routing Control via *PortOwner* and Disconnect Event

Manipulating the port routing via the *CF-bit* is an extreme process and not intended to be used during normal operation. The normal mode of port ownership transferal is on the granularity of individual ports using the *Port Owner* bit in the EHCI HC's PORTSC

register (for hand-offs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI *CF-bit* is set to a one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a port-change interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.
- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the *LineStatus* bits in the PORTSC register. If they indicate the attached device is a full-speed device (e.g. D+ is asserted), then the EHCI Driver sets the *PortReset* control bit to a one (and sets the *PortEnable* bit to a zero) which begins the reset-process. Software times the duration of the reset, then terminates reset signaling by writing a zero to the port reset bit. The reset process is actually complete when software reads a zero in the *PortReset* bit. The EHCI Driver checks the PortOwner bit in the PORTSC register. If set to a one, the connected device is a high-speed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.
- At the time the EHCI Driver receives the port reset and enable request the *LineStatus* bits might indicate a low-speed device. Additionally, when the port reset process is complete, the *PortEnable* field may indicate that a full-speed device is attached. In either case the EHCI driver sets the PortOwner bit in the PORTSC register to a one to release port ownership to a companion host controller.
- When the EHCI Driver sets *PortOwner* bit to a one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI PORTSC register observes and reports a disconnect event via the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that port-state, the EHCI Driver responds with a reset complete change set to a one, a connect change set to a one and a connect status set to a zero. This information is derived directly from the EHCI port register. This will allow the

hub driver to assume the device was disconnected during reset. It will acknowledge the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected. The device-connect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's *CF-bit* transitions from a 1b to a 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects will be detected by the EHCI port register and the process will repeat.

56.4.4.2.3 Example Port Routing State Machine

Figure 56-9 illustrates an example of how the port ownership should be managed. The following sections describe the entry conditions to each state.

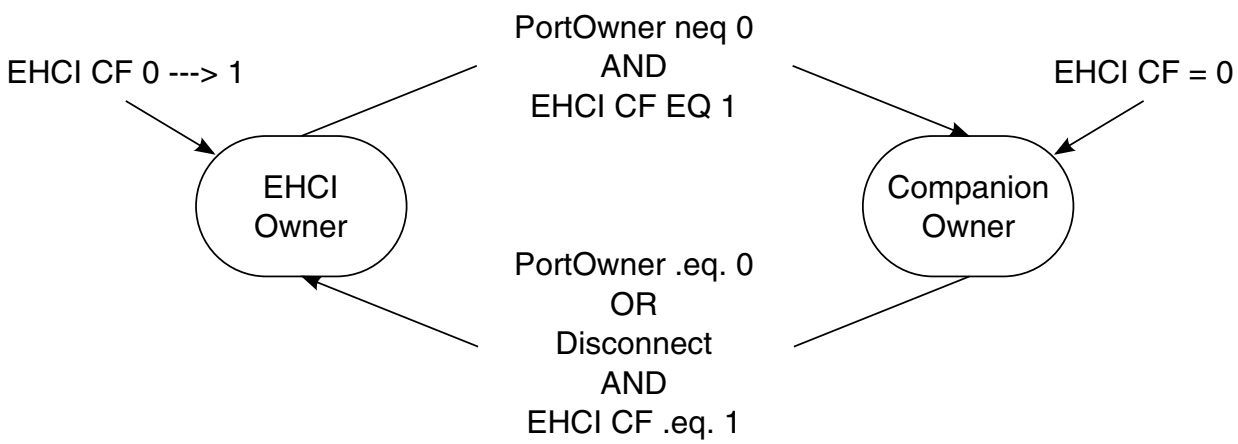


Figure 56-9. Port Owner Handoff State Machine

56.4.4.2.3.1 EHCI HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the EHCI HC's *Configure Flag* (*CF*) bit in the CONFIGFLAG register transitions from a zero to a one. This signals the fact that the system has a host controller driver for the EHCI HC and that all ports in the USB 2.0 host controller must default route to the EHCI controller.

- When the port is owned by a companion HC and the device is disconnected from the port. The EHCI port routing control logic is notified of the disconnect, and returns port routing to the EHCI controller. The connection state of the companion HC goes immediately to the disconnected state (with appropriate side effect to connect change, enable and enable change). The companion HC driver will acknowledge the disconnect by setting the connect status change bit to a zero. This allows the companion HC's driver to interact with the port completely through the disconnect process.
- When system software writes a zero to the *PortOwner* bit in the PORTSC register. This allows software to take ownership of a port from a companion host controller. When this occurs, the routing logic to the companion HC effectively signals a disconnect to the companion HC's port status and control register.

56.4.4.2.3.2 Companion HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the *Port Owner* field transitions from a zero to a one.
- When the HS-mode HC's *Configure Flag* (*CF*) is equal to zero.

On entry to this state, the routing logic allows the companion HC port register to detect a device connect. Normal port enumeration proceeds.

56.4.4.2.4 Port Power

The *Port Power Control* (*PPC*) bit in the HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (See section Host Controller Structural Parameters (USB_HCSPARAMS - EHCI Compliant)). When this bit is a zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the *PPC* bit is a one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is referred to in this discussion as *PortPowerOutputEnable* (*PPE*). *PPE* is controlled based on the state of the combination bits *PPC* bit, EHCI *Configured* (*CF*)-bit and individual *Port Power* (*PP*) bits. Table 56-37 illustrates the summary behavioral model.

Table 56-37. Port Power Enable Control Rules

CF	CHC[1] (PP)	EHC[2] (PP)	Owner	PPE[3]	Description
0	0	X	CHC	0	When the EHCI controller has not been configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off.
0	1	X	CHC	1	Similar to previous entry. When the companion HC's port power select is on, then the port power is on.
1	0	0	CHC	0	Port owner has port power turned off, the power to port is off.
1	0	0	EHC	0	Port owner has port power turned off, the power to port is off.
1	0	1	EHC	1	Port owner has port power on, so power to port is on.
1	0	1	CHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	EHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	EHC	1	Port owner has port power on, so power to port is on.

1. CHC (Companion Host Controller).
2. EHC (EHCI Host Controller).
3. PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists).

56.4.4.2.5 Port Reporting Over-Current

Host controllers are by definition power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI PORTSC register has an over-current status and over-current change bit. The functionality of these bits is specified in the USB Specification Revision 2.0.

The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an over-current condition it is made available to both the companion and EHCI ports. The effect of an over-current status on a companion host controller port is beyond the scope of this document. The over-current condition effects the following bits in the PORTSC register on the EHCI port:

- *Over-current Active* bits are set to a one. When the over-current condition goes away, the *Over-currentActive* bit will transition from a one to a zero.
- *Over-current Change* bits are set to a one. On every transition of the *Over-currentActive* bit the host controller will set the *Over-current Change* bit to a one. Software sets the *Over-current Change* bit to a zero by writing a one to this bit.
- *Port Enabled/Disabled* bit is set to a zero. When this change bit gets set to a one, then the *Port Change Detect* bit in the USBSTS register is set to a one.
- *Port Power (PP)* bits may optionally be set to a zero. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to *limit* the current and leave power applied. When the *Over-current Change* bit transitions from a zero to a one, the host controller also sets the *Port Change Detect* bit in the USBSTS register to a one. In addition, if the *Port Change Interrupt Enable* bit in the USBINTR register is a one, then the host controller will issue an interrupt to the system. Refer to Table 56-38 for summary behavior for over-current detection when the host controller is halted (suspended from a device component point of view).

56.4.4.3 Host Operational Model Suspend/Resume

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub. Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely via software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated, or software initiated resumes are called Resume Events/Actions. Bus-initiated resume events are called wake-up events. The classes of wakeup events are:

- Remote-wakeup enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the per-port control bits in the PORTSC registers.

Selective suspend is a feature supported by every PORTSC register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the *Run/Stop* bit in the USBCMD register to a zero. The EHCI module can then be placed into a

lower device state via the PCI power management interface (See Appendix A, Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com).

When a wake event occurs the system will resume operation and system software will eventually set the *Run/Stop* bit to a one and resume the suspended ports. Software must not set the *Run/Stop* bit to a one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the CPU is restarted. So, by definition, if software is running, clocks in the system are stable and the *Run/Stop* bit in the USBCMD register can be set to a one. There are also minimum system software delays defined in the PCI Power Management Specification. Refer to this specification for more information.

56.4.4.3.1 Port Suspend/Resume

System software places individual ports into suspend mode by writing a one into the appropriate PORTSC *Suspend* bit. Software must only set the *Suspend* bit when the port is in the enabled state (*Port Enabled* bit is a one) and the EHCI is the port owner (*Port Owner* bit is a zero).

The host controller may evaluate the *Suspend* bit immediately or wait until a micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the *Suspend* bit. The host controller must evaluate the *Suspend* bit at least every frame boundary.

System software can initiate a resume on a selectively suspended port by writing a one to the *Force Port Resume* bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state (see Section Port Status and Control (USB_PORTSC)). If system software sets *Force Port Resume* bit to a one when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 milliseconds after a port indicates that it is suspended (*Suspend* bit is a one) before initiating a port resume via the *Force Port Resume* bit. When *Force Port Resume* bit is a one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 milliseconds) then sets the *Force Port Resume* bit to a zero. When the host controller receives the write to transition *Force Port Resume* to zero, it completes the resume sequence as defined in the USB specification, and sets both the *Force Port Resume* and *Suspend* bits to zero. Software-initiated port resumes do not affect the *Port Change Detect* bit in the USBSTS register nor do they cause an interrupt if the *Port Change Interrupt Enable* bit in the USBINTR register is a one. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is

reflected downstream within 100 µsec. The port's *Force Port Resume* bit is set to a one and the *Port Change Detect* bit in the USBSTS register is set to a one. If the *Port Change Interrupt Enable* bit in the USBINTR register is a one the host controller will issue a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 msec), then terminates the resume sequence by writing zero to the *Force Port Resume* bit in the port. The host controller receives the write of zero to *Force Port Resume*, terminates the resume sequence and sets *Force Port Resume* and *Suspend* port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the PORTSC register and observing that the *Suspend* and *Force Port Resume* bits are zero. Software must ensure that the host controller is running (i.e. *HCHalted* bit in the USBSTS register is a zero), before terminating a resume by writing a zero to a port's *Force Port Resume* bit. If *HCHalted* is a one when *Force Port Resume* is set to a zero, then SOFs will not occur down the enabled port and the device will return to suspend mode in a maximum of 10 milliseconds.

Table 56-38 summarizes the wake-up events. Whenever a resume event is detected, the *Port Change Detect* bit in the USBSTS register is set to a one. If the *Port Change Interrupt Enable* bit is a one in the USBINTR register, the host controller will also generate an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the *Port Change Detect* status bit in the USBSTS register.

Table 56-38. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	not D0
Port disabled, resume K-State received	No Effect	N/A	N/A
Port suspended, Resume K-State received	Resume reflected downstream on signaled port. Force Port Resume status bit in PORTSC register is set to a one. Port Change Detect bit in USBSTS register set to a one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a one. A disconnect is detected.	Depending in the initial port state, the PORTSC Connected Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a zero. A disconnect is detected.	Depending on the initial port state, the PORTSC Connect and Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]

Table continues on the next page...

Table 56-38. Behavior During Wake-up Events (continued)

Port Status and Signaling Type	Signaled Port Response	Device State	
Port is not connected and the port's WKCNNT_E bit is a one. A connect is detected.	PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [2]	[2]
Port is not connected and the port's WKCNNT_E bit is a zero. A connect is detected.	PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]
Port is connected and the port's WKOC_E bit is a one. An over-current condition occurs.	PORTSC Over-current Active, Over-current Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one	[1], [2]	[2]
Port is connected and the port's WKOC_E bit is a zero. An over-current condition occurs.	PORTSC Over-current Active, Over-current Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]
[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USBINTR register is a one. [2] PME# asserted if enabled (Note: PME Status must always be set to a one). [3] PME# not asserted.			

56.4.4.4 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule. The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware / software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the *PERIODICLISTBASE* register (see Section Frame List Base Address (USB_PERIODICLISTBASE)). The *PERIODICLISTBASE* register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Host Data Structure. In each micro-frame, if the periodic schedule is enabled (see Section Periodic Scheduling Threshold) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It will only execute from the asynchronous schedule after it encounters the end of the periodic schedule. The host

controller traverses the periodic schedule by constructing an array offset reference from the *PERIODICLISTBASE* and the *FRINDEX* registers (see Figure 56-10). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a *next* link pointer of a schedule data structure having its *T-bit* set to a one. When the host controller encounters a *T-Bit* set to a one during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. Once this transition is made, the host controller executes from the asynchronous schedule until the end of the micro-frame.

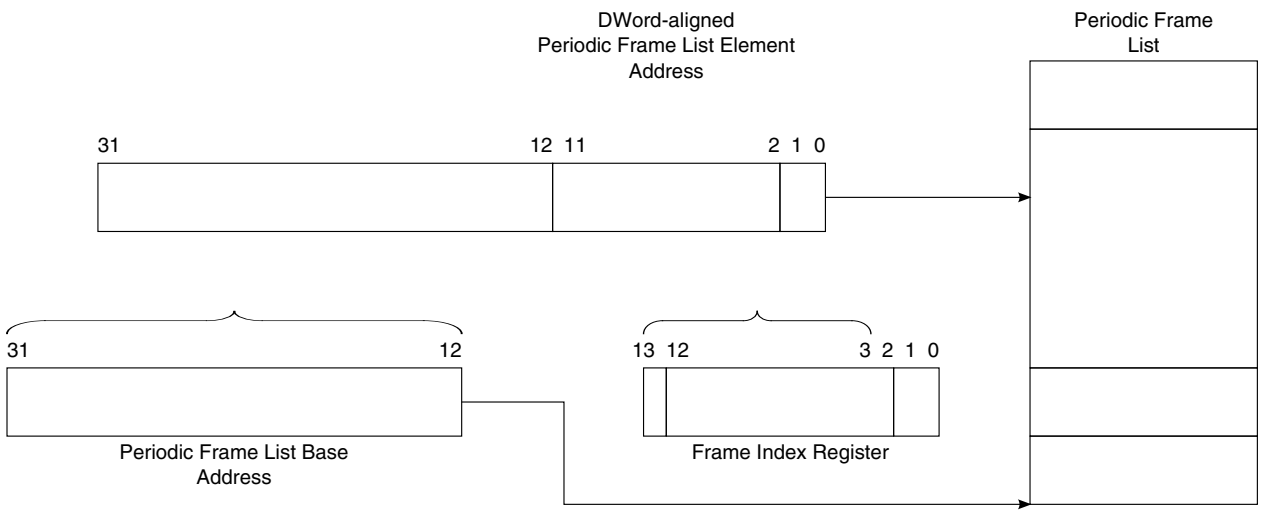


Figure 56-10. Derivation of Pointer into Frame List Array

When the host controller determines that it is time to execute from the asynchronous list, it uses the operational register *ASYNCLISTADDR* to access the asynchronous schedule, see Figure 56-11.

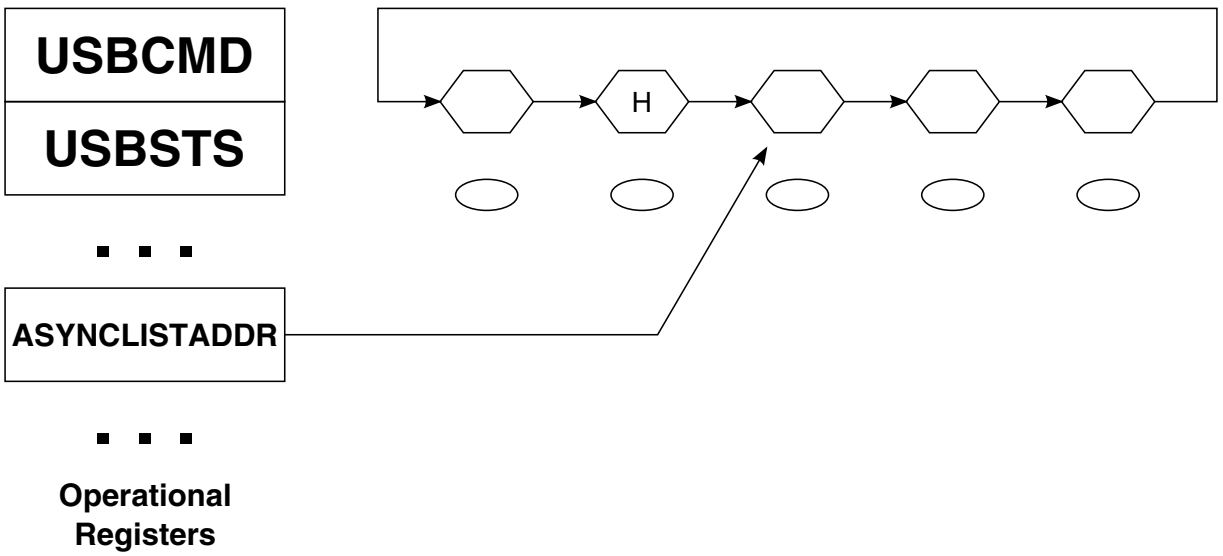


Figure 56-11. General Format of Asynchronous Schedule List

The *ASYNCLISTADDR* register contains a physical memory pointer to the *next* queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the *ASYNCLISTADDR* register. Software must set queue head *horizontal* pointer *T-bits* to a zero for queue heads in the asynchronous schedule. See Section Asynchronous Schedule for complete operational details.

56.4.4.4.1 Example - Preserving Micro-Frame Integrity

One of the requirements of a USB host controller is to maintain *Frame Integrity*. This means that the HC must preserve the micro-frame boundaries. For example: SOF packets must be generated on time (within the specified allowable jitter), and High-speed EOF1,2 thresholds must be enforced. The end of micro-frame timing points EOF1 and EOF2 are clearly defined in the USB Specification Revision 2.0. One implication of this responsibility is that the HC must ensure that it does not start transactions that will not be completed before the end of the micro-frame. More precisely, no transactions should be started by the host controller, which cannot be completed in their entirety before the EOF1 point. In order to enforce this rule, the host controller must check each transaction before it starts to ensure that it will complete before the end of the micro-frame.

So, what exactly needs to be involved in this check? Fundamentally, the transaction data payload, plus bit stuffing, plus transaction overhead must be taken into consideration. It is possible to be extremely accurate on how much time the next transaction will take. Take OUTs for an example. The host controller must fetch all of the OUT data from memory in order to send it onto the USB bus. A host controller implementation could pre-fetch *all* of the OUT data, and pre-compute the actual number of bits in the token and data

packets. In addition, the system knows the depth of the target endpoint, so it could closely estimate turnaround time for handshake. In addition, the host controller knows the size of a handshake packet. Pre-computing effects of bit stuffing and summing up the other overhead numbers could allow the host controller to know exactly whether there was enough bus time, before EOF1 to complete the OUT transaction. To accomplish this particular approach takes an inordinate amount of time and hardware complexity.

The alternative is to make a reasonable guess whether the next transaction can be started. An example approximation algorithm is described below. This example algorithm relies on the EHCI policy that periodic transactions are scheduled first in the micro-frame. It is a reasonable assumption that software will never over-commit the micro-frame to periodic transactions greater than the specification allowable 80%. In the available remaining 20% bandwidth, the host controller has some ability (in this example) to decide whether or not to execute a transaction. The result of this algorithm is that sometimes, under some circumstances a transaction will not be executed that *could* have been executed. However, under all circumstances, a transaction will never be started unless there is enough time in the frame to complete the transaction.

56.4.4.4.1.1 Transaction Fit - A Best-Fit Approximation Algorithm

A curve is calculated which represents the *latest* start time for every packet size, at which software will schedule the start of a periodic transaction. This curve is the 80% bandwidth curve. Another curve is calculated which is the absolute, latest permitted start time for every packet size. This curve represents the absolute latest time, that a transaction of each packet size can be started and completed, in the micro-frame. A plot of these two curves is illustrated in Figure 56-12. The plot Y-axis represents the number of byte-times left in a frame.

The space between the *80%* and the *Last Start* plots is bandwidth reclamation area. In this algorithm the host controller may skip transactions during this time if it is prudent.

The Best-Fit Approximation method plots a function ($f(x)$) between the 80% and *Last Start* curves. The function $f(x)$ adds a constant to every transaction's maximum packet size and the result compared with the number of bytes left in the frame. The constant represents an approximation of the effects of bit stuffing and protocol overhead. The host controller starts transactions whose results land *above* the function curve. The host controller will not start transactions whose results land below the function curve.

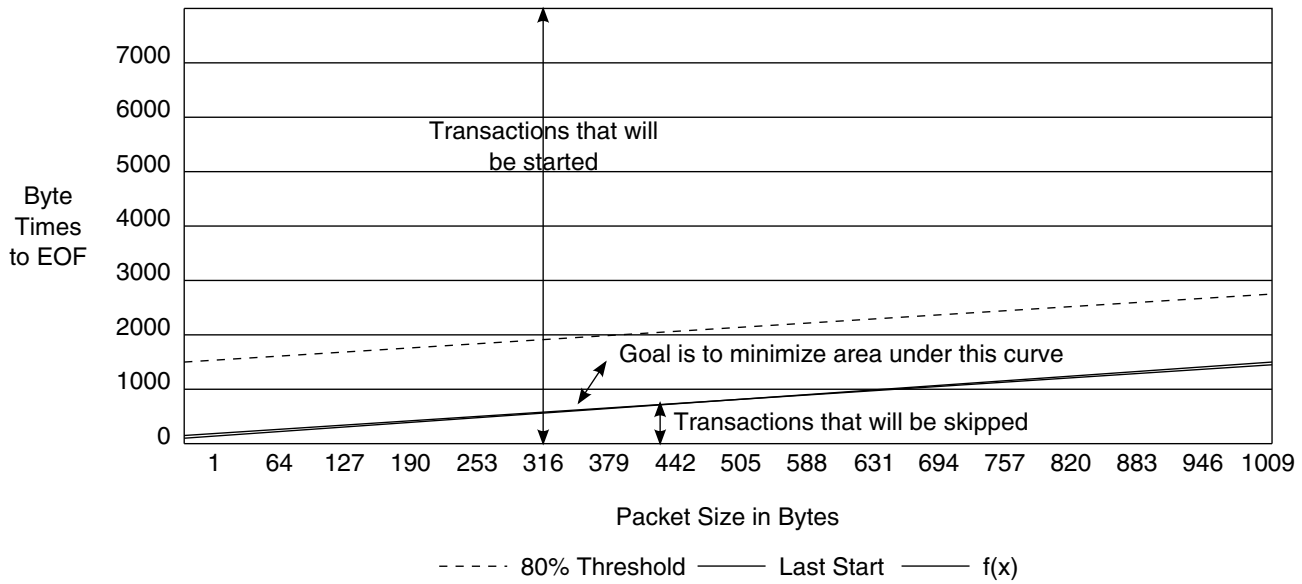


Figure 56-12. Best Fit Approximation

The *LastStart* line was calculated in this example to assume the absolute worst-case bus overhead per transaction. The particular transaction used was a start-split, zero-length OUT transaction with a handshake. Summaries of the component parts are listed in Table 56-39. The component times were derived from the protocol timings defined in the USB Specification Revision 2.0.

Table 56-39. Example Worse-case Transaction Timing Components

Component	Bit time	Byte Time	Explanation
Split Token	76	9.5	Split token as defined in USB core specification. Includes sync, token, eop, etc.
Host 2 Host IPG	88	11	Number of bit times required between consecutive host packets.
Token	67	8.375	Token as defined in USB core specification. Includessync, token, eop, etc.
Host 2 Host IPG	88	11	Same as above
Data Packet (0 data bytes)	66.7	8.34	Zero-length data packet. Includes sync, PID, crc16, eop, etc.
Turnaround time	721	90.125	Time for packet initiator (Host) to see the beginning of a response to a transmitted packet.
Handshake packet	48	6	Handshake packet as defined in USB core specification. Includes sync, PID, eop, etc.
		144	Total

The exact details of the function (f(x)) are up to the particular implementation. However, it should be obvious that the goal is to minimize the area under the curve between the approximation function and the *Last Start* curve, without dipping below the *LastStart* line, while at the same time keeping the check as simple as possible for hardware

implementation. The f(x) in Figure 56-12 was constructed using the following pseudo-code test on each transaction size data point. This algorithm assumes that the host controller keeps track of the remaining bits in the frame.

```
Alorithm CheckTransactionWillFit (MaximumPacketSize, HC_BytesLeftInFrame)
Begin
          Local Temp = MaximumPacketSize + 192
          Local rvalue = TRUE
          If MaximumPacketSize >= 128 then
                    Temp += 128
          End If
          If Temp > HC_BytesLeftInFrame then
                    Rvalue = FALSE
          End If
          Return rvalue
End
```

This algorithm takes two inputs, the current maximum packet size of the transaction and a hardware counter of the number of bytes left in the current micro-frame. It unconditionally adds a simple constant of 192 to the maximum packet size to account for a first-order effect of transaction overhead and bit stuffing. If the transaction size is greater than or equal to 128 bytes, then an additional constant of 128 is added to the running sum to account for the additional worst-case bit stuffing of payloads larger than 128. An inflection point was inserted at 128 because the *f*(*x*) plot was getting *close* to the *LastStart* line.

56.4.4.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(s) below USB 2.0 Hubs be strictly aligned. Super-imposed on this requirement is that USB 2.0 Hubs manage full- and low-speed transactions via a micro-frame pipeline (see start- (SS) and complete- (CS) splits illustrated in Figure 56-13). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

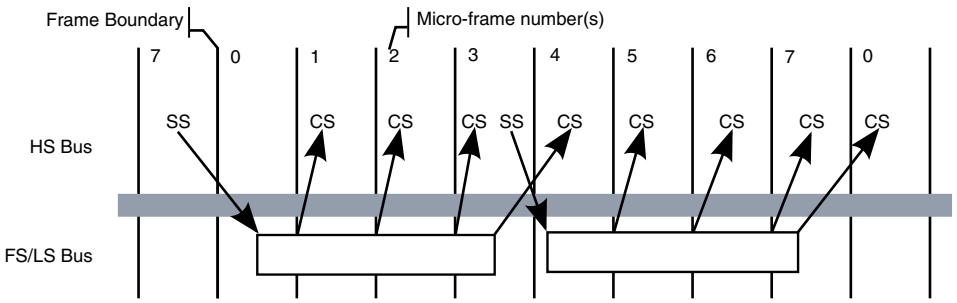


Figure 56-13. Frame Boundary Relationship between HS bus and FS/LS Bus

The simple projection, as Figure 56-13 illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement a one micro-frame phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed via the Frame List Index Register (FRINDEX) documented in Section USB Frame Index (USB_FRINDEX) and initially illustrated in Section Schedule Traversal Rules . Bits FRINDEX[2:0], represent the micro-frame number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are increment based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one micro-frame. The one micro-frame delay yields host controller periodic schedule and bus frame boundary relationship as illustrated in Figure 56-14. This adjustment allows software to trivially schedule the periodic start and complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phase-shift are beyond the scope of this specification.

Figure 56-14 illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined. The host controller's view of the 1-millisecond boundaries is called *H-Frames*. The high-speed bus's view of the 1-millisecond boundaries is called *B-Frames*.

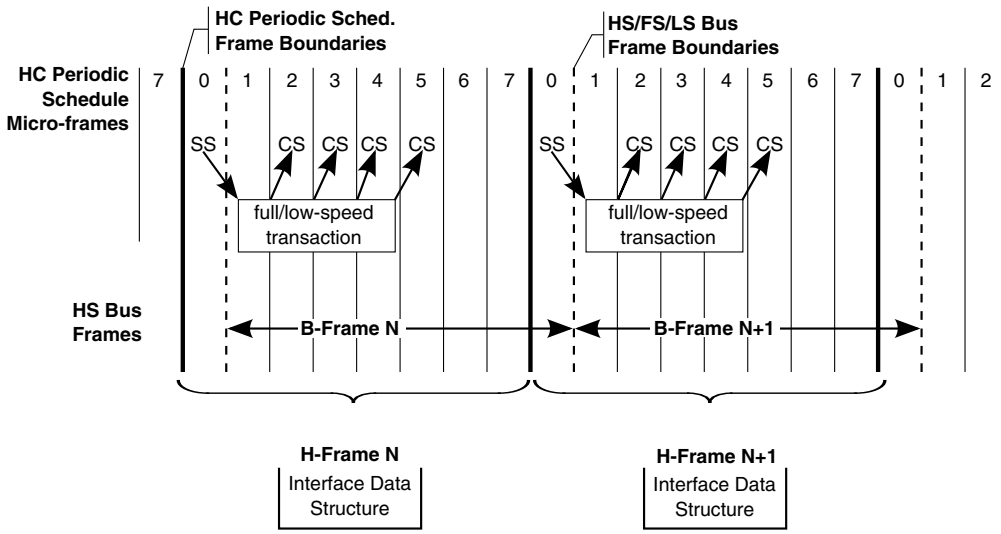


Figure 56-14. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Micro-frame numbers for the *H-Frame* are tracked by FRINDEX[2:0]. *B-Frame* boundaries are visible on the high-speed bus via changes in the SOF token's frame number. Micro-frame numbers on the high-speed bus are only derived from the SOF token's frame number (i.e. the high-speed bus will see eight SOFs with the same frame number value). *H-Frames* and *B-Frames* have the fixed relationship (i.e. *B-Frames* lag *H-Frames* by one micro-frame time) illustrated in Figure 56-14 . The host controller's periodic schedule is naturally aligned to *H-Frames*. Software schedules transactions for full- and low-speed periodic endpoints relative the *H-Frames*. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in Section USB Frame Index (USB_FRINDEX), the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one micro-frame count. Table 56-40 illustrates the required relationship between the value of FRINDEX and the value of SOFV. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carry-out on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0].

Software is allowed to write to FRINDEX. Section USB Frame Index (USB_FRINDEX) provides the requirements that software should adhere when writing a new value in FRINDEX.

Table 56-40. Operation of FRINDEX and SOFV (SOF Value Register)

	Current			Next	
FRINDEX[F]	SOFV	FRINDEX[mF]	FRINDEX[F]	SOFV	FRINDEX[mF]
N	N	111b	N+1	N	000b
N+1	N	000b	N+1	N+1	001b
N+1	N+1	001b	N+1	N+1	010b
N+1	N+1	010b	N+1	N+1	011b
N+1	N+1	011b	N+1	N+1	100b
N+1	N+1	100b	N+1	N+1	101b
N+1	N+1	101b	N+1	N+1	110b
N+1	N+1	110b	N+1	N+1	111b

NOTE

Where [F] = [13:3]; [μF] = [2:0]

56.4.4.6 Periodic Schedule

The periodic schedule traversal is enabled or disabled via the *Periodic Schedule Enable* bit in the USBCMD register. If the *Periodic Schedule Enable* bit is set to a zero, then the host controller simply does not try to access the periodic frame list via the *PERIODICLISTBASE* register. Likewise, when the *Periodic Schedule Enable* bit is a one, then the host controller does use the *PERIODICLISTBASE* register to traverse the periodic schedule. The host controller will not react to modifications to the *Periodic Schedule Enable* immediately. In order to eliminate conflicts with split transactions, the host controller evaluates the *Periodic Schedule Enable* bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b micro-frame. These work items must be removed from the schedule before the *Periodic Schedule Enable* bit is written to a zero. The *Periodic Schedule Status* bit in the USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing a one (or zero) to the *Periodic Schedule Enable* bit in the USBCMD register. Software then can poll the *Periodic Schedule Status* bit to determine when the periodic schedule has made the desired transition. Software must not modify the *Periodic Schedule Enable* bit unless the value of the *Periodic Schedule Enable* bit equals that of the *PeriodicSchedule Status* bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the USB. Figure 56-15

illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (e.g. closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

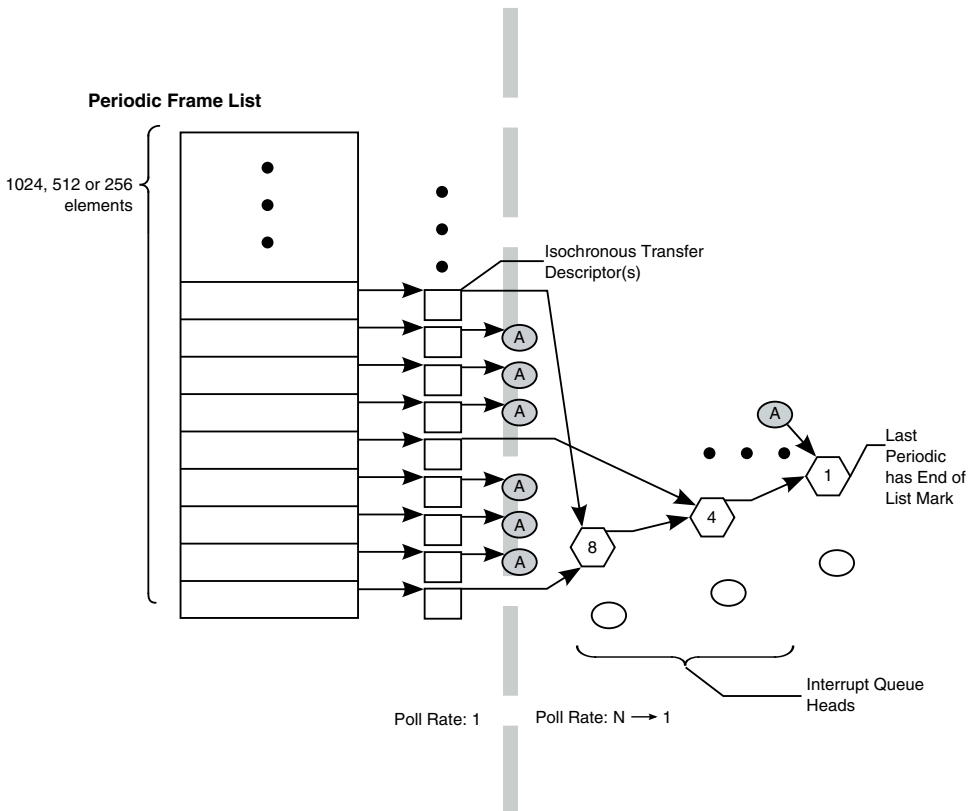


Figure 56-15. Example Periodic Schedule

56.4.4.7 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (High-Speed) Transfer Descriptor (iTD). There are four distinct sections to an iTD:

- The first field is the *Next Link Pointer*. This field is for schedule linkage purposes only;
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.

- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

56.4.4.7.1 Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Each iTD can span 8 micro-frames worth of transactions. When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array. If the *active* bit in the *Status* field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the *Next* pointer to the next schedule data structure.

When the indexed *active* bit is a one the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, etc.). It also uses the *Page Select (PG)* field to index the *buffer pointer* array, storing the selected *buffer pointer* and the next sequential *buffer pointer*. For example, if *PG* field is a 0, then the host controller will store Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's *PG* field) and the transaction description's *Transaction Offset* field. The host controller uses the endpoint addressing information and *I/O-bit* to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the *Status* field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet's offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (example: page 0 pointer) selected by the active transaction descriptions' *PG* (example value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer will cross a page boundary. When this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (example: page 1 pointer) and continues to move data. The

size of each bus transaction is determined by the value in the *Maximum Packet Size* field. An iTD supports high-bandwidth pipes via the *Mult* (multiplier) field. When the *Mult* field is 1, 2, or 3, the host controller executes the specified number of *Maximum Packet* sized bus transactions for the endpoint in the current micro-frame. In other words, the *Mult* field represents a transaction count for the endpoint in the current micro-frame. If the *Mult* field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the *Mult* field.

For OUT transfers, the value of the *Transaction X Length* field represents the total bytes to be sent during the micro-frame. The *Mult* field must be set by software to be consistent with *Transaction X Length* and *Maximum Packet Sixe.* The host controller will send the bytes in *Maximum Packet Size*'d portions. After each transaction, the host controller decrements it's local copy of *Transaction X Length* by *Maximum Packet Size*. The number of bytes the host controller sends is always *Maximum Packet Size* or *Transaction X Length*, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues *Mult* transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use *Maximum Packet Size* to detect packet babble errors. The host controller keeps the sum of bytes received in the *Transaction X Length* field. After all transactions for the endpoint have completed for the micro-frame, *Transaction X Length* contains the total bytes received. If the final value of *Transaction X Length* is less than the value of *Maximum Packet Size*, then less data than was allowed for was received from the associated endpoint. This *short packet* condition does not set the *USBINT* bit in the USBSTS register to a one. The host controller will not detect this condition. If the device sends more than *Transaction X Length* or *Maximum Packet Size* bytes (whichever is less), then the host controller will set the *Babble Detected* bit to a one and set the *Active* bit to a zero. Note, that the host controller is not required to update the iTD field *Transaction X Length* in this error scenario. If the *Mult* field is greater than one, then the host controller will automatically execute the value of *Mult* transactions. The host controller will not execute all *Mult* transactions if:

- The endpoint is an OUT and *Transaction X Length* goes to zero before all the *Mult* transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before *Mult* transactions have been executed. The end of micro-frame may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next micro-frame. Refer to Appendix D for a table

summary of the host controller required behavior for all the high-bandwidth transaction cases.

56.4.4.7.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

Figure 56-16 illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (i.e. the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left is the host controller's frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

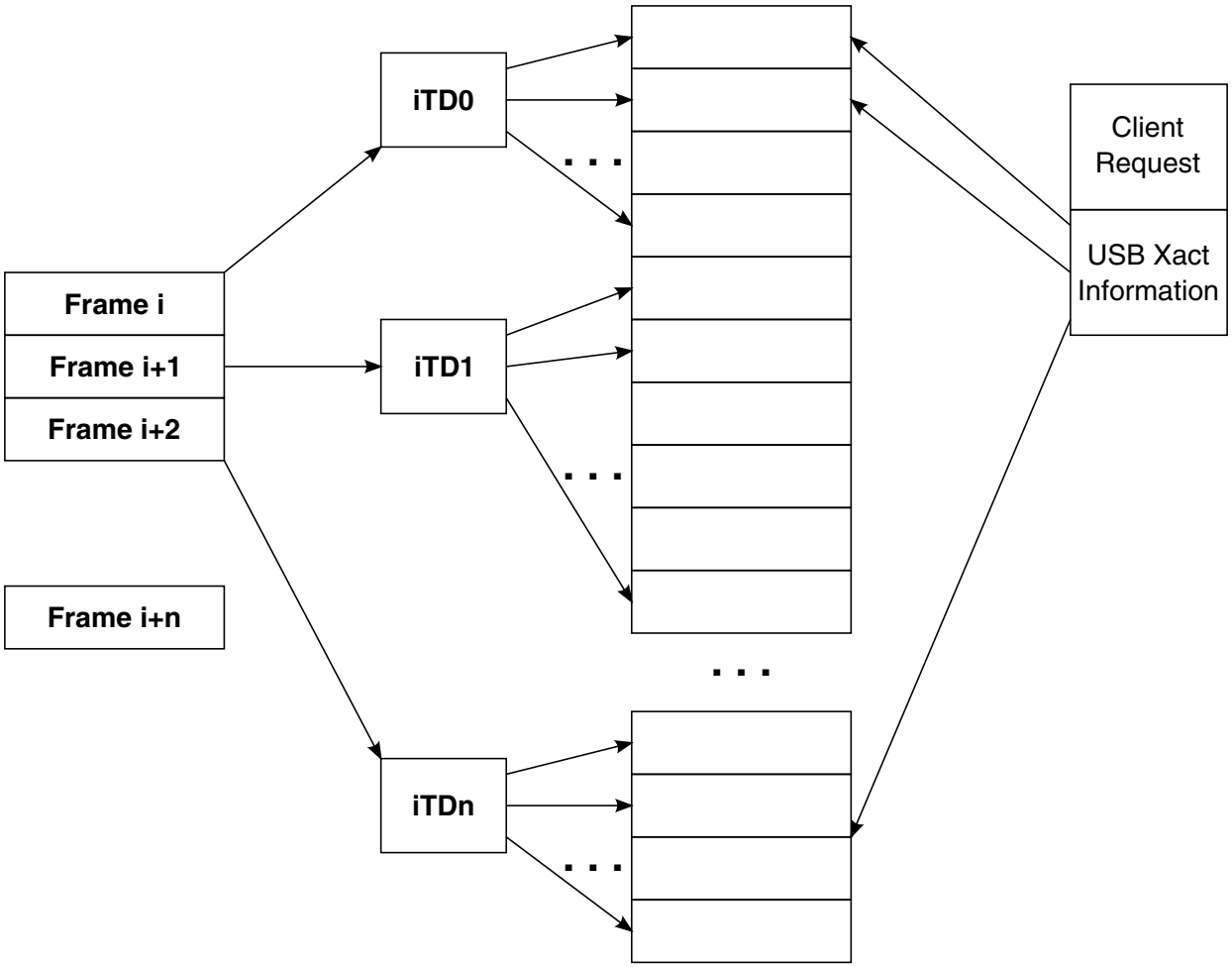


Figure 56-16. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the *PG* field is set to index the correct physical buffer page pointer and the *Transaction Offset* field is set relative to the correct buffer pointer page (e.g. the same one referenced by the *PG* field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current *Page Buffer Pointer* (as selected by the *PG* field) and concatenates to the *transaction offset* field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available *Page Buffer Pointer*. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer will wrap a page boundary. Doing so will yield undefined behavior. The host controller hardware is not required to 'alias' the page selector to page zero. USB 2.0

isochronous endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to a one.

56.4.4.7.2.1 Periodic Scheduling Threshold

The *Isochronous Scheduling Threshold* field in the HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures. It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 micro-frames worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. There are three basic caching models that account for the fact the isochronous data structures span 8 micro-frames. The three caching models are: no caching, micro-frame caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Of course, there is no information about where in the micro-frame the host controller is, so a constant uncertainty-factor of one micro-frame has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the *Isochronous Scheduling Threshold* field. The host controller may pre-fetch data structures during a periodic schedule traversal (per micro-frame) but will always dump any accumulated schedule state at the end of the micro-frame. At the appropriate time relative to the beginning of every micro-frame, the host controller always begins schedule traversal from the frame list. Software can use the value of the FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 micro-frames in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the *Isochronous Scheduling Threshold* field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 micro-frames). Software uses the value of the FRINDEX register (plus the constant 1 uncertainty) to determine the *current* micro-frame/frame (assume modulo 8 arithmetic in

adding the constant 1 to the micro-frame number). For any current frame N, if the current micro-frame is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current micro-frame is 7, then software can add isochronous transactions to Frame N + 2.

Micro-frame caching is indicated with a non-zero value in the least-significant 3 bits of the *Isochronous Scheduling Threshold* field. System software assumes the host controller caches one or more periodic data structures for the number of micro-frames indicated in the *Isochronous Scheduling Threshold* field. For example, if the count value were 2, then the host controller keeps a *window* of 2 micro-frames worth of state (current micro-frame, plus the next) on-chip. On each micro-frame boundary, the host controller releases the current micro-frame state and begins accumulating the next micro-frame state.

56.4.4.8 Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled via the *Asynchronous Schedule Enable* bit in the USBCMD register. If the *Asynchronous Schedule Enable* bit is set to a zero, then the host controller simply does not try to access the asynchronous schedule via the *ASYNCLISTADDR* register. Likewise, when the *Asynchronous Schedule Enable* bit is a one, then the host controller does use the *ASYNCLISTADDR* register to traverse the asynchronous schedule. Modifications to the *Asynchronous Schedule Enable* bit are not necessarily immediate. Rather the new value of the bit will only be taken into consideration the next time the host controller needs to use the value of the *ASYNCLISTADDR* register to get the next queue head.

The *Asynchronous Schedule Status* bit in the USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to the *Asynchronous Schedule Enable* bit in the USBCMD register. Software then can poll the *Asynchronous Schedule Status* bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the *Asynchronous Schedule Enable* bit unless the value of the *Asynchronous Schedule Enable* bit equals that of the *AsynchronousSchedule Status* bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the *ASYNCLISTADDR* register. The default value of the *ASYNCLISTADDR* register after reset is undefined and the schedule is disabled when the *Asynchronous Schedule Enable* bit is a zero.

Software may only write this register with defined results when the schedule is disabled. e.g. *Asynchronous Schedule Enable* bit in the USBCMD and the *Asynchronous Schedule Status* bit in the USBSTS register are zero. System software enables execution from the

asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asychronous schedule by setting the *Asynchronous Schedule Enable* bit is set to one. The asynchronous schedule is actually enabled when the *Asynchronous Schedule Status* bit is a one.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the *ASYNCLISTADDR* register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller "completes" processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the *ASYNCLISTADDR* register. Next time the asynchronous schedule is accessed, this is the first data structure that will be serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller "completes" processing the asynchronous schedule when one of the following events occur:

- The end of a micro-frame occurs.
- The host controller detects an empty list condition (i.e. see Section Empty Asynchronous Schedule Detection)
- The schedule has been disabled via the *Asynchronous Schedule Enable* bit in the USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 56-11. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

56.4.4.8.1 Adding Queue Heads to Asynchronous Schedule

This is a software requirement section. There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the *ASYNCLISTADDR* register, then enables the list by setting the *Asynchronous Schedule Enable* bit in the USBCMD register to a one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example qTD pointers have *T-Bit*s set to a one or reference valid qTDs and the *Horizontal Pointer* references a valid queue head data structure. The following algorithm represents the functional requirements:

```
          InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
           --
           -- Requirement: all inputs must be properly initialized.
           --
           -- pQHeadCurrent is a pointer to a queue head that is
           -- already in the active list
           -- pQHeadNew is a pointer to the queue head to be added
           --
           -- This algorithm links a new queue head into a existing
           -- list
           --
           pQueueHeadNew.HorizontalPointer  =  pQueueHeadCurrent.HorizontalPointer
           pQueueHeadCurrent.HorizontalPointer =  physicalAddressOf(pQueueHeadNew)
End InsertQueueHead
```

56.4.4.8.2 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section. There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list. Software deactivates the asynchronous schedule by setting the *Asynchronous Schedule Enable* bit in the USBCMD register to a zero. Software can determine when the list is idle when the *Asynchronous Schedule Status* bit in the USBSTS register is a zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list via the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
           --
           -- Requirement: all inputs must be properly initialized.
           --
           -- pQHeadPrevious is a pointer to a queue head that
           -- references the queue head to remove
           -- pQHeadToUnlink is a pointer to the queue head to be
           -- removed
           -- pQheadNext is a pointer to a queue head still in the
           -- schedule. Software provides this pointer with the
           -- following strict rules:
           --          if the host software is one queue head, then
           --          pQHeadNext must be the same as
           --          QueueheadToUnlink.HorizontalPointer. If the host
           --          software is unlinking a consecutive series of
           --          queue heads, QHeadNext must be set by software to
```

```
          --          the  queue head remaining in the schedule.
          --
          -- This algorithm unlinks a queue head from a circular list
          --
          pQueueHeadPrevious.HorizontalPointer =  pQueueHeadToUnlink.HorizontalPointer
                                  pQueueHeadToUnlink.HorizontalPointer = pQHeadNext
End UnlinkQueueHead
```

If software removes the queue head with the *H-bit* set to a one, it must select another queue head still linked into the schedule and set its *H-bit* to a one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its *H-bit* set to a one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers, etc.). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (*Interrupt on Async Advance Doorbell* bit in the USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (*Interrupt on Async Advance* bit in the USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to a one, it also sets the command bit to a zero. The third bit is an interrupt enable (*Interrupt on Async Advance* bit in the USBINTR register) that is matched with the status bit. If the status bit is a one and the interrupt enable bit is a one, then the host controller will assert a hardware interrupt.

Figure 56-17 illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that will remain in the asynchronous schedule.

When the host controller observes that doorbell bit being set to a one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A & B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (i.e. traversed beyond queue head (B) in this example).

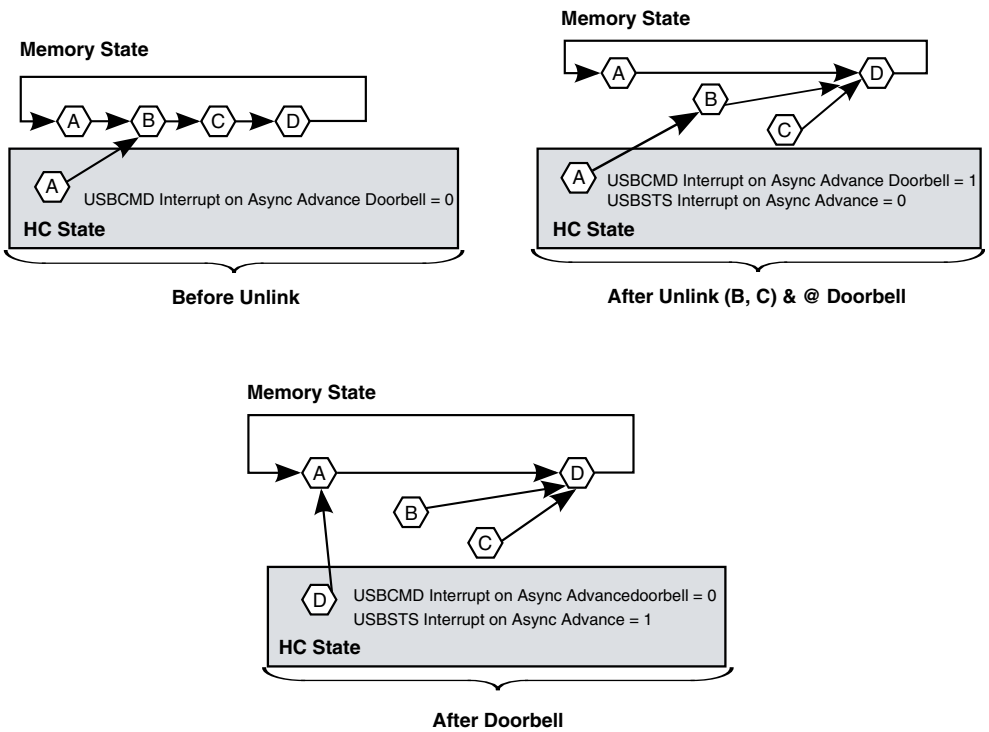


Figure 56-17. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (e.g. observed the head of the queue (twice)) before setting the *Advance on Async* status bit to a one.

Software may re-use the memory associated with the removed queue heads after it observes the *Interrupt on Async Advance* status bit is set to a one, following assertion of the doorbell. Software should acknowledge the *Interrupt on Async Advance* status as indicated in the USBSTS register, before using the doorbell handshake again.

56.4.4.8.3 Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty. The queue head data structure (see Table 56-26) defines an *H-bit* in the queue head, which allows software to mark a queue head as being the *head* of the reclaim list. The Enhanced Host Controller Interface also keeps a 1-bit flag in the USBSTS register (*Reclamation*) that is set to a zero when the Enhanced Interface Host

Controller observes a queue head with the H-bit set to a one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Section Asynchronous Schedule Traversal : *Start Event*).

If the Enhanced Host Controller Interface ever encounters an *H-bit* of one and a *Reclamation* bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule.

An example illustrating the H-bit in a schedule is illustrated in Figure 56-18

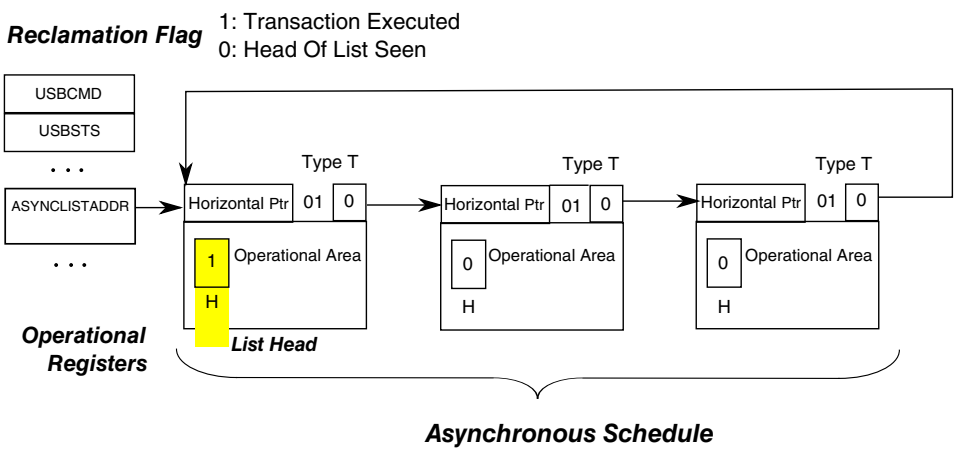


Figure 56-18. Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the *H-bit* set to a one, and that it is always coherent with respect to the schedule.

56.4.4.8.4 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list *long* before the end of the micro-frame. It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the micro-frame. Given the EHCI simple traversal rules, the complete-split for that transaction may Nak/Nyet out very quickly. If it is the only item in the schedule, then the host controller will cease traversal of the Asynchronous schedule very early in the micro-frame. In order to provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current micro-frame, instead of waiting until the next micro-frame. When the reason for host controller idling asynchronous

schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

56.4.4.8.4.1 Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. The question is: *how longshould the host controller stay idle before restarting*?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller will stay idle before restarting traversal. In this example, the manifest constant is called *AsyncSchedSleepTime*, and has a value of 10µsec. The value is derived based on the analysis in Section Example Derivation for *AsyncSchedSleepTime*, The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-micro-Frame event occurs, or an empty list is detected. If the event is an End-of-micro-Frame, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a *schedule sleep* state.
- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to *schedule active* state. The start context allows the HC to reload *Nakcnt* fields, etc. so the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each micro-frame. Figure 56-19 illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is specifically a special state where the host controller is just waiting for a period of time before resuming execution of the Asynchronous schedule.

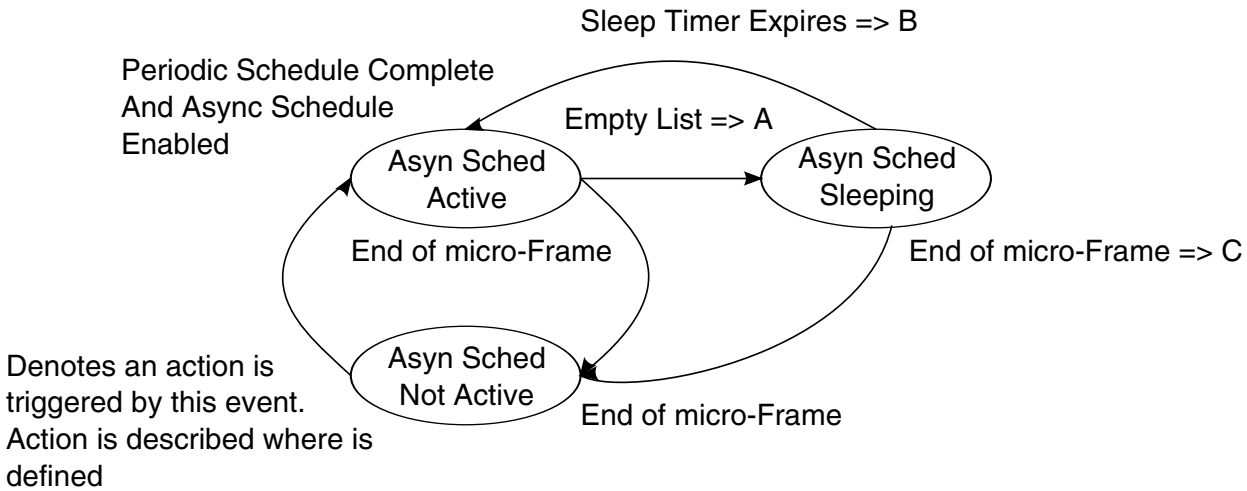


Figure 56-19. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in Figure 56-19 are defined in Table 56-41.

Table 56-41. Asynchronous Schedule SM Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USBSTS register to a one and moves the Nak Counter reload state machine to WaitForListHead (see Section Nak Count Reload Control).
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

56.4.4.8.4.2 Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine will not leave this state when the *Asynchronous Schedule Enable* bit in the USBCMD register is a zero.

This state is entered from **Async Sched Active** or **Async Sched Sleeping** states when the end-of-micro-frame event is detected.

56.4.4.8.4.3 Async Sched Active

This state is entered from the **Async Sched Not Active** state when the periodic schedule is not active. It is also entered from the **Async Sched Sleeping** states when the *AsyncrhonousTraversalSleepTimer* expires. On every transition into this state, the host controller sets the *Reclamation* bit in the USBSTS register to a one.

While in this state, the host controller will continually traverse the asynchronous schedule until either the end of micro-frame or an empty list condition is detected.

56.4.4.8.4.4 Async Sched Sleeping

The state is entered from the **Async Sched Active** state when a schedule empty condition is detected. On entry to this state, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.

56.4.4.8.4.5 Example Derivation for *AsyncSchedSleepTime*

The derivation is based on analysis of what work the host controller could be doing next. It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests. Table 56-42 summarizes some of the typical 'next transactions' that could be in the schedule, and the amount of time (e.g. *footprint*, or *wall clock*) the transaction will take to complete.

Table 56-42. Typical Low-/Full-speed Transaction Times

Transaction Attributes		Footprint (time)	Description
Speed	HS	11.9 ms	Maximum foot print for a worst-case, full-sized bulk data transaction.
Size	512	9.45 ms	Maximum footprint for an approximate best-case, full-sized bulk data transaction.
Type	Bulk	-	-
Speed	FS	~50 ms	Approximate typical for full-sized bulk data. An 8-byte low-speed is about 2x, or between 90 and 100 ms.
Size	64	-	-
Type	Bulk	-	-
Speed	FS	~12 ms	Approximate typical for 8-byte bulk/control (i.e. setup)
Size	8	-	-
Type	Cntrl	-	-

A *AsyncSchedSleepTime* value of 10 µs provides a reasonable relaxation of the system memory load and still provides a good level of service for the various transfer types and payload sizes. For example, say we detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller will get the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it nak'd, the 10µs sleep period would allow the host controller to get the NAK results on the first complete-split.

56.4.4.8.5 Asynchronous Schedule Traversal : *Start Event*

Once the HC has *idled* itself via the empty schedule detection (Section 0), it will naturally *activate* and begin processing from the Periodic Schedule at the beginning of each micro-frame. In addition, it may have idled itself early in a micro-frame. When this occurs (idles early in the micro-frame) the HC must occasionally *re-activate* during the micro-frame and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Section Restarting Asynchronous Schedule Before EOF . Asynchronous schedule *Start Event*s are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the micro-frame is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Section Restarting Asynchronous Schedule Before EOF).

56.4.4.8.6 Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (Section Empty Asynchronous Schedule Detection) depends on the proper management of the *Reclamation* bit in the USBSTS register. The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (See Section Fetch Queue Head).

It is required that the host controller sets the *Reclamation* bit to a one whenever an asynchronous schedule traversal *Start Event*, as documented in Section Asynchronous Schedule Traversal : *Start Event*, occurs. The *Reclamation* bit is also set to a one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Section Execute Transaction). The host controller sets the *Reclamation* bit to a zero whenever it finds a queue head with its *H-bit* set to a one. Software should only set a queue head's *H-bit* if the queue head is in the asynchronous schedule. If software sets the *H-bit* in an interrupt queue head to a one, the resulting behavior is undefined. The host controller may set the *Reclamation* bit to a zero when executing from the periodic schedule.

56.4.4.9 Operational Model for Nak Counter

This section describes the operational model for the *NakCnt* field defined in a queue head (see Section Queue Head Initialization). Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control via the Nak response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally Nak or Nyet the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced via the same asynchronous schedule. The time between the *Start-split* transaction and the first *Complete-split* transaction could be very short (i.e. like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively Naks) the *Complete-split* transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which will not complete until after the transaction on the classic bus completes.

There are two component fields in a queue head to support the throttling feature: a counter field (*NakCnt*), and a counter reload field (*RL*). *NakCnt* is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. There are two operational modes associated with this counter:

- Not Used. This mode is set when the *RL* field is zero. The host controller ignores the *NakCnt* field for any execution of transactions through a queue head with an *RL* field of zero. Software must use this selection for interrupt endpoints.
- Nak Throttle Mode. This mode is selected when the *RL* field is non-zero. In this mode, the value in the *NakCnt* field represents the maximum number of Nak or Nyet responses the host controller will tolerate on each endpoint. In this mode, the HC will decrement the *NakCnt* field based on the token/handshake criteria listed in Table 56-43 The host controller must reload *NakCnt* when the endpoint successfully moves data (e.g. policy to reward device for moving data).

Table 56-43. NakCnt Field Adjustment Rules

Token	Handshake	
	Handshake NAK	NYET
IN/PING	decrement *NakCnt*	N/A (protocol error)
OUT	decrement *NakCnt*	No Action[1, 1] Start
Split	decrement *NakCnt*	N/A (protocol error)
Complete Split	No Action	Decrement *NakCnt*

1. Recommended behavior on this response is to reload *NakCnt*

In summary, system software enables the counter by setting the reload field (*RL*) to a non-zero value. The host controller may execute a transaction if *NakCnt* is non-zero. The host controller will not execute a transaction if *NakCnt* is zero. The reload mechanism is described in detail in Section Nak Count Reload Control .

Note that when all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller will detect an empty Asynchronous Schedule and idle schedule traversal (see Section Empty Asynchronous Schedule Detection).

Any time the host controller begins a new traversal of the Asynchronous Schedule, a *Start Event* is assumed, see Section Asynchronous Schedule Traversal : *Start Event*. Every time a Start-Event occurs, the Nak Count reload procedure is enabled.

56.4.4.9.1 Nak Count Reload Control

When the host controller reaches the *Execute Transaction* state for a queue head (meaning that it has an active operational state), it checks to determine whether the *NakCnt* field should be reloaded from *RL* (see Section Execute Transaction). If the answer is yes, then *RL* is copied into *NakCnt*. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload nak counters (*NakCnt* see Table 56-26) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Section Asynchronous Schedule Traversal : *Start Event* for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 56-18). Figure 56-20 illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the nak counter during the queue head traversal state: Execute Transaction (Figure 56-20). The host controller does not perform the nak counter reload operation if the RL field (see Table 56-26) is set to zero.

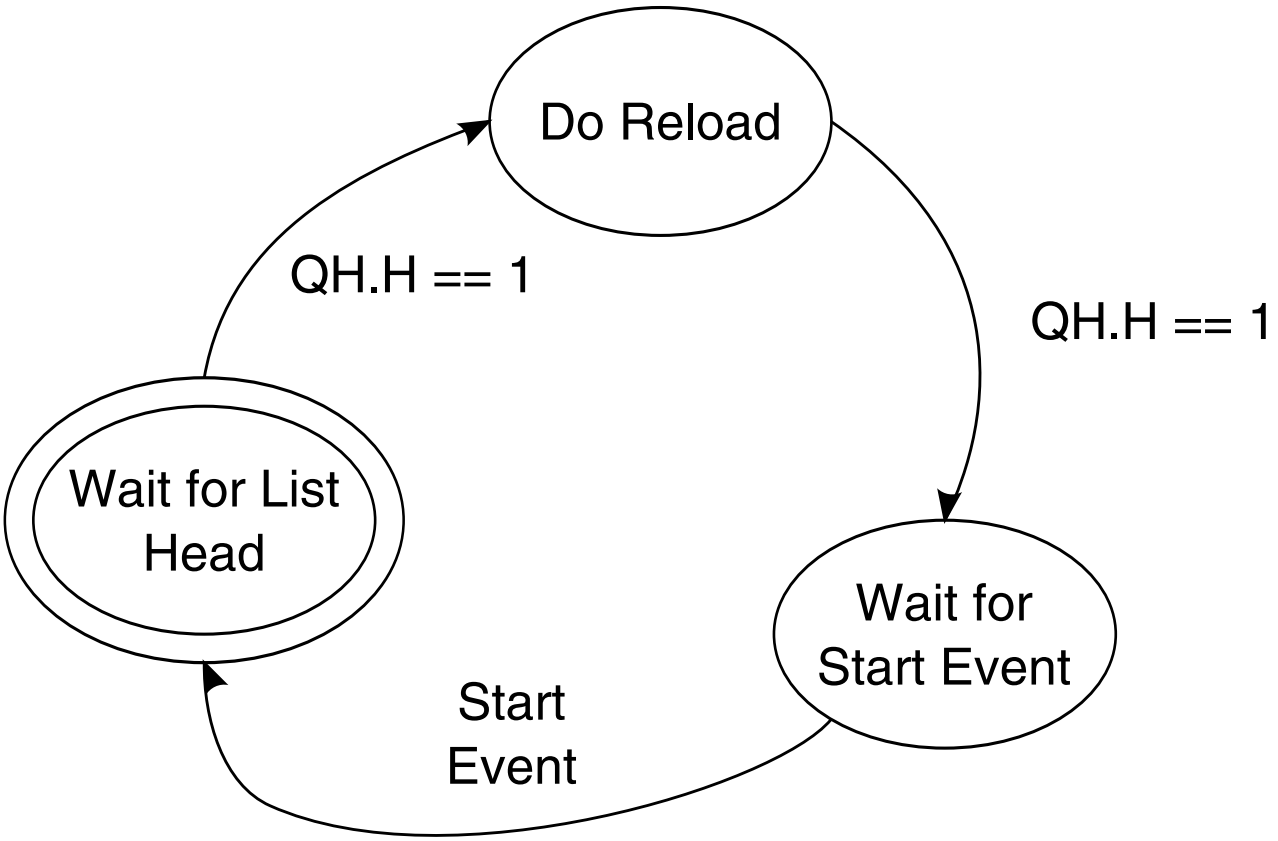


Figure 56-20. Example HC State Machine for Controlling Nak Counter Reloads

56.4.4.9.1.1 Wait for List Head

This is the initial state. The state machine enters this state from Wait for Start Event when a start event as defined in Section Asynchronous Schedule Traversal : *Start Event* occurs. The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule. This occurs when the host controller fetches a queue head whose *H-bit* is set to a one.

56.4.4.9.1.2 Do Reload

This state is entered from the Wait for List Head state when the host controller fetches a queue head with the *H-bit* set to a one. While in this state, the host controller will perform nak counter reloads for every queue head visited that has a non-zero nak reload value (*RL*) field.

56.4.4.9.1.3 Wait for Start Event

This state is entered from the *Do Reload* state when a queue head with the *H-bit* set to a one is fetched. While in this state, the host controller will not perform nak counter reloads.

56.4.4.10 Managing Control/Bulk/Interrupt Transfers via Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure. One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in Table 56-26). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a *transfer*.

The general processing model for the host controller's use of a queue head is simple:

- read a queue head,
- execute a transaction from the overlay area,
- write back the results of the transaction to the overlay area
- move to the next queue head.

If the host controller encounters errors during a transaction, the host controller will set one (or more) of the error reporting bits in the queue head's *Status* field. The *Status* field accumulates all errors encountered during the execution of a qTD (e.g. the error bits in the queue head *Status* field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (e.g. buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions will occur for the endpoint and the host controller will not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in Figure 56-21. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the *Fetch QH* state in order to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and *activate* them, then the host controller will always *find* them if/when they are reachable.

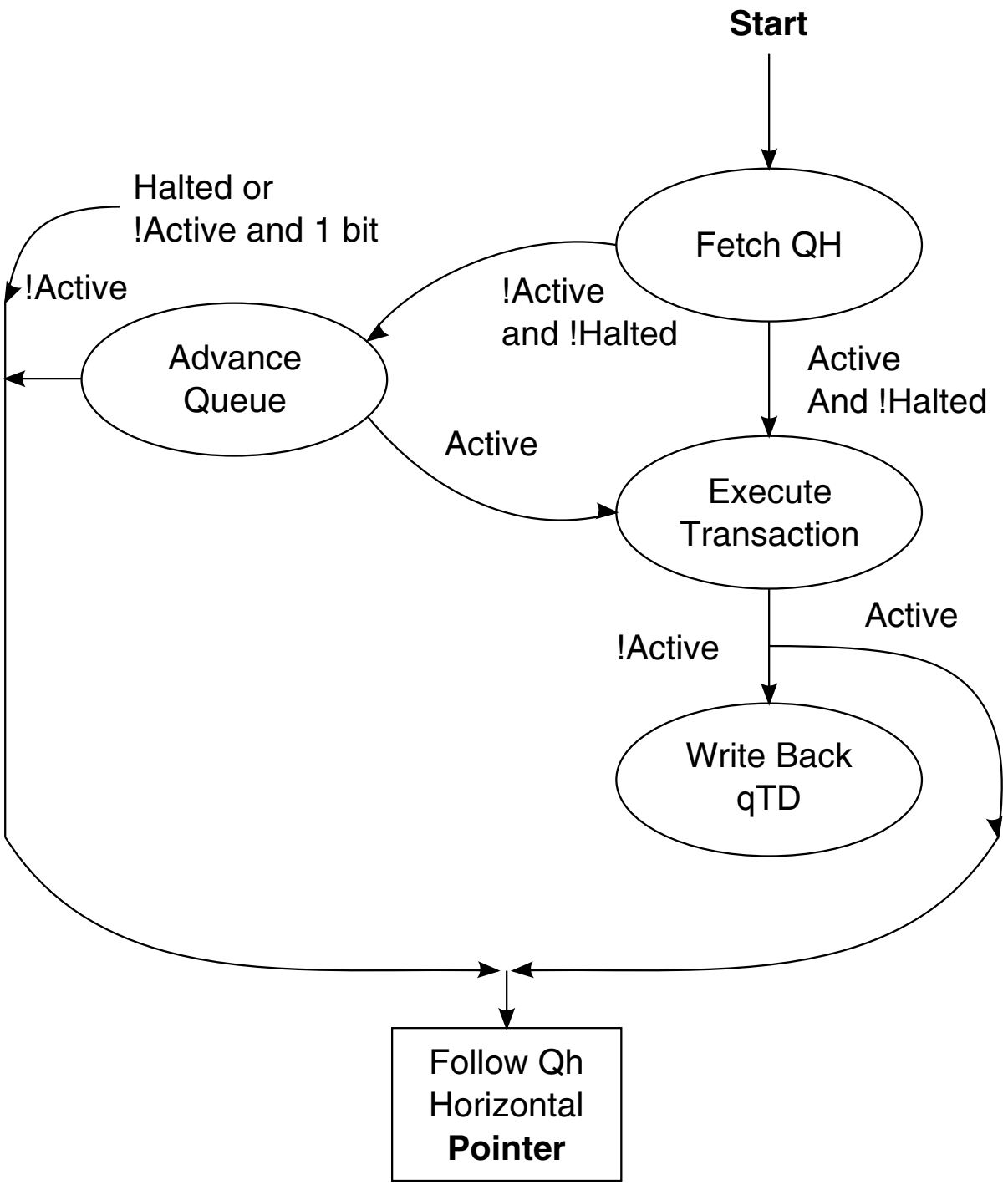


Figure 56-21. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The Execute Transaction state (Section Execute Transaction) describes the basic requirements for all endpoints. Sections Split

Transactions for Asynchronous Transfers and Split Transaction Interrupt describe details of the required extensions to the Execute Transaction state for endpoints requiring split transactions.

Note: Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see SectionQueue Head for layout of a queue head):

- Valid static endpoint state
- For the very first use of a queue head, software may zero-out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

56.4.4.10.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (Section Next Asynch. Address (USB_ASYNCLISTADDR)). Additionally, it may be referenced from the *Next LinkPointer* field of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the *Typ* field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Table 56-26.

While in this state, the host controller performs operations to implement empty schedule detection (Section Empty Asynchronous Schedule Detection) and Nak Counter reloads (Section Operational Model for Nak Counter). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (i.e. *S-mask* is a zero), and
- The *H-bit* is a one, and
- The *Reclamation* bit in the USBSTS register is a zero.

When these criteria are met, the host controller will stop traversing the asynchronous list (as described in Section Empty Asynchronous Schedule Detection). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the *H-bit* is a one and the *Reclamation* bit is a one, then the host controller sets the *Reclamation* bit in the USBSTS register to a zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Section Operational Model for Nak Counter.

This state is complete when the queue head has been read on-chip.

56.4.4.10.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the **FetchQHD** state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay. Note that if the *I-bit* is a one and the *Active* bit is a zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field *Bytes to Transfer* is not zero and the *T-bit* in the *Alternate Next qTD Pointer* is set to zero, then the host controller uses the *Alternate Next qTD Pointer*. Otherwise, the host controller uses the *NextqTD Pointer*. If *NextqTD Pointer*'s *T-bit* is set to a one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has it's *Active* bit set to a one, the host controller moves the pointer value used to reach the qTD (*Next* or *Alternate Next*) to the *Current qTD Pointer* field, then performs the overlay. If the fetched qTD has its *Active* bit set to a zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure. The host controller performs the overlay based on the following rules:

- The value of the data toggle (*dt*) field in the overlay area depends on the value of the *data toggle control* (*dtc*) bit (see Table 56-28).
- If the *EPS* field indicates the endpoint is a high-speed endpoint, the *Ping* state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- *C-prog-mask* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *Frame Tag* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *NakCnt* field in the overlay area is loaded from the *RL* field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

56.4.4.10.3 Execute Transaction

The host controller enters this state from the **Fetch Queue Head state** only if the *Active* bit in *Status* field of the queue head is set to a one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's *S-mask* field contains a non-zero value. It is the responsibility of software to ensure the *S-mask* field is appropriately initialized based on the transfer type. There are other criteria that must be met if the *EPS* field indicates that the endpoint is a low- or full-speed endpoint, see Sections Split Transactions for Asynchronous Transfers and Split Transaction Interrupt .

56.4.4.10.3.1 Interrupt Transfer Pre-condition Criteria

If the queue head is for an interrupt endpoint (e.g. non-zero *S-mask* field), then the FRINDEX[2:0] field must identify a bit in the *S-mask* field that has a one in it. For example, an *S-mask* value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction.

56.4.4.10.3.2 Asynchronous Transfer Pre-operations and Pre-condition Criteria

If the queue head is not for an interrupt endpoint (e.g. a zero *S-mask* field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria: The pre-operation is:

Checks the Nak counter reload state (Section Operational Model for Nak Counter). It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

The pre-condition evaluated is:

- Whether or not the *NakCnt* field has been reloaded, the host controller checks the value of the *NakCnt* field in the queue head. If *NakCnt* is non-zero, or if the *Reload Nak Counter* field is zero, then the host controller considers this queue head for a transaction.

56.4.4.10.3.3 Transfer Type Independent Pre-operations

Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is:

- A host controller internal transaction (down) counter *qHTransactionCounter* is loaded from the queue head's *Mult* field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt

queue heads. Software should ensure that the *Mult* field is set appropriately for the transfer type.

The pre-conditions evaluated are:

- The host controller determines whether there is enough time in the micro-frame to complete this transaction (see Section Transaction Fit - A Best-Fit Approximation Algorithm for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.
- If the value of *qHTransactionCounter* for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the *Reclamation* bit in the USBSTS register to a one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates *qHTransactionCounter* times in this state executing transactions. After each transaction is executed, *qHTransactionCounter* is decremented by one. The host controller will exit this state when one of the following events occurs:

- The *qHTransactionCounter* decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,[4] or
- The transaction experiences a transaction error, or
- The *Active* bit in the queue head goes to a zero, or
- There is not enough time in the micro-frame left to execute the next transaction (see Section Transaction Fit - A Best-Fit Approximation Algorithm for example method for implementing the frame boundary test).

NOTE

[4] Note that for a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer state, the *Total Bytes to Transfer* field is decremented by the number of bytes moved in the transaction, the data toggle bit (*dt*) is toggled, the current page offset is advanced to the next appropriate value (e.g. advanced by the number of bytes successfully moved), and the *C_Page* field is updated to the appropriate value (if necessary). See Section Buffer Pointer List Use for Data Streaming with qTDs .

Note that the *Total Bytes To Transfer* field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller will execute a zero-length transaction to the endpoint. If the *PID_Code* field indicates an IN transaction and

the device delivers data, the host controller will detect a packet babble condition, set the *babble* and *halted* bits in the *Status* field, set the *Active* bit to a zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (e.g. not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (e.g. decrement *qHTransactionCounter*). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a Nak (or Nyet) and *RL* is non-zero, *NakCnt* is decremented by one. If *RL* is zero, then no write-back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of *CErr* did not change). Software should set the *RL* field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly *NakCnt*, the host controller writes back the results of the transaction to the queue head's overlay area in main memory.

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is *MaximumPacket Size*. The number of bytes moved during an OUT transaction is either *Maximum Packet Length* bytes or *Total Bytes to Transfer*, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The *CErr* field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Section Transaction Error .

The following events will cause the host controller to clear the *Active* bit in the queue head's overlay status field. When the *Active* bit transitions from a one to a zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the *Active* bit) determines the next state.

- *CErr* field decrements to zero. When this occurs the *Halted* bit is set to a one and *Active* is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the *Halted* bit is set to a one and the *Active* bit is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.

- The *Total Bytes to Transfer* field is zero after the transaction completes. Note that for a zero length transaction, it was zero before the transaction was started. When this condition occurs, the *Active* bit is set to zero.
- The PID code is an IN, and the number of bytes moved during the transaction is less than the *Maximum Packet Length*. When this occurs, the *Active* bit is set to zero and a short packet condition exists. The short-packet condition is detected during the Advance Queuestate. Refer to Section Split Transactions for additional rules for managing low- and full-speed transactions.

With the exception of a NAK response (when *RL* field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields: The *PID Code* field indicates an IN and the device sends more than the expected number of bytes (e.g. *Maximum Packet Length* or *Total Bytes to Transfer* bytes, whichever is less) (e.g. a packet babble). This results in the host controller setting the *Halted* bit to a one.

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

56.4.4.10.3.4 Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed via queue heads (control, bulk and interrupt). The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USBSTS register to a one. To halt the queue head, the *Active* bit is set to a zero and the *Halted* bit is set to a one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller will not advance the transfer state on a transaction that results in a *Halt* condition (e.g. no updates

necessary for *Total Bytes to Transfer*, *C_Page*, *Current Offset,* and *dt*). The host controller must update *CErr* as appropriate. When a queue head is halted, the *USB Error Interrupt* bit in the USBSTS register is set to a one. If the *USB Error Interrupt Enable* bit in the USBINTR register is set to a one, a hardware interrupt is generated at the next interrupt threshold.

56.4.4.10.3.5 Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule. This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the *Mult* feature that is used in the Periodic schedule. Where-as the *Mult* feature is a characteristic that is tunable for each endpoint; park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Section Execute Transaction apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the *Asynchronous Schedule Park Capability* bit in the HCCPARAMs register to a one. This information keys system software that the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register are modifiable. System software enables the feature by writing a one to the *Asynchronous Schedule Park Mode Enable* bit.

When park-mode is not enabled (e.g. *Asynchronous Schedule Park Mode Enable* bit in the USBCMD register is a zero), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose *EPS* field indicates a Low/Full-speed device (i.e. only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose *EPS* field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose *EPS* field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the

Asynchronous Schedule Park Mode Count field in the USBCMD register. Software must not set *Asynchronous Schedule Park Mode Enable* bit to a one and also set *Asynchronous Schedule Park Mode Count* field to a zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Section Execute Transaction . It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the *Mult* field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter *PM-Count* from *Asynchronous Schedule Park Mode Count* when *Asyncrhonous Schedule Park Mode Enable* bit is a one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, *PM-Count* is decremented. The host controller may continue to execute bus transactions from the current queue head until *PM-Count* goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake. Table 56-44 summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

Table 56-44. Actions for Park Mode, based on Endpoint Response and Residual Transfer State

PID	Endpoint Response	Transfer State after Transaction		Action
IN	DATA[0,1] w/Maximum Packet sized data	Not zero	Not Zero	Allowed to perform another bus transaction.[1,, 2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't care	Move to next QH.
	DATA[0,1] w/short packet	Don't care	Don't care	Retire qTD and move to next QH.
	NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH.
OUT	ACK	Not zero	Not Zero	Allowed to perform another bus transaction. [2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't' care	Move to next QH.
	NYET, NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH
PING	ACK	Not Zero	Not Zero	Allowed to perform another bus transaction. [2]
	NAK	Don't care	Don't care	Move to next QH
	STALL, XactErr	Don't care	Don't care	Move to next QH

1. Note, the host controller may continue to execute bus transactions from the current high-speed queue head (if *PM-Count* is not equal to zero), if a PID mismatch is detected (e.g. expected DATA1 and received DATA0, or visa-versa).,
2. Note, this specification does not *require* that the host controller execute another bus transaction when *PM-Count* is non-zero. Implementations are encouraged to make appropriate complexity and performance trade-offs.

56.4.4.10.4 Write Back qTD

This state is entered from the Execute Transaction state when the *Active* bit is set to a zero. The source data for the write-back is the transfer results area of the queue head overlay areaTable 56-44. The host controller uses the *Current qTD Pointer* field as the target address for the qTD. The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include *Total Bytes to Transfer*, *Cerr*, and *Status*.

The duration of this state depends on when the qTD write-back has been committed.

56.4.4.10.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the *Active* bit is a one on exit from the **Execute Transaction** state, or
- When the host controller exits the **Write Back qTD state**, or
- If the **Advance Queue** state fails to advance the queue because the target qTD is not active, or
- If the *Halted* bit is a one on exit from the **Fetch QH state**.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

56.4.4.10.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be *virtually contiguous*. This means: if the buffer spans more than one physical page, it must obey the following rules (Figure 56-22 illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4K chunk beyond the first page, each buffer portion matches to a full 4K

page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16Kbyte buffer with any starting buffer alignment.

The host controller uses the field *C_Page* field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction will span a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing *C_Page* and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the *Bytes to Transfer* field.

Figure 56-22 illustrates a nominal example of how System software would initialize the buffer pointers list and the *C_Page* field for a transfer size of 16383 bytes. *C_Page* is set to zero. The upper 20-bits of Page 0 references the start of the physical page. *Current Offset* (the lower 12-bits of queue head Dword 7) holds the offset in the page e.g. 2049 (e.g. 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4K page.

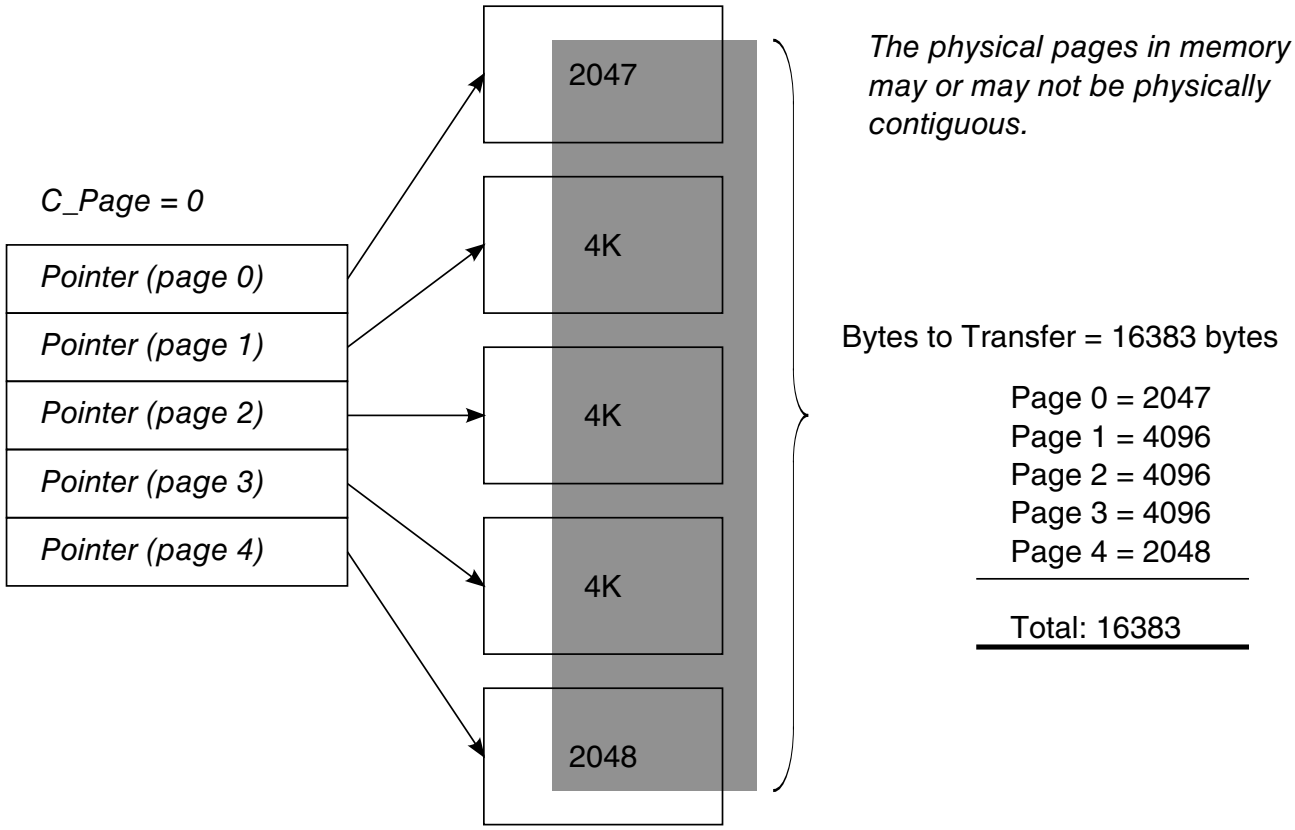


Figure 56-22. Example Mapping of qTD Buffer Pointers to Buffer Pages

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because *C_Page* is set to zero) and concatenates the *Current Offset* field. The 512 bytes are moved during the transaction, the *Current Offset* and *Total Bytes to Transfer* are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller will increment *C_Page* (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and *Current Offset* has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (i.e. *C_Page*) when necessary. There are three conditions for how the host controller handles *C_Page*.

- The current transaction does not span a page boundary. The value of *C_Page* is not adjusted by the host controller.

- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (i.e. the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment *C_Page* before writing back status for the transaction.

Note that the only valid adjustment the host controller may make to *C_Page* is to increment by one.

56.4.4.10.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate. System software sets a bit in a queue head's *S-Mask* to indicate which micro-frame with-in a 1 millisecond period a transaction should be executed for the queue head. Software must ensure that all queue heads in the periodic schedule have *S-Mask* set to a non-zero value. An *S-mask* with a zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and *S-Mask* values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in Table 56-45.

Table 56-45. Example Periodic Reference Patterns for Interrupt Transfers with 2ms Poll Rate

Frame # Reference Sequence	Description
0, 2, 4, 6, 8, etc. *S-Mask* = 01h	A queue head for the *bInterval* of 2 milliseconds (16 micro-frames) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the *S-Mask* field in the queue head is set to 01h, indicating that the transaction for the endpoint should be executed on the bus during micro-frame 0 of the frame.
0, 2, 4, 6, 8, etc. *S-Mask* = 02h	Another example of a queue head with a *bInterval* of 2 milliseconds is linked into the periodic frame list at exactly the same interval as the previous example. However, the *S-Mask* is set to 02h indicating that the transaction for the endpoint should be executed on the bus during micro-frame 1 of the frame.

56.4.4.10.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller will set an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an *Interrupt on Complete* (*IOC*) bit set to a one, or whenever a transfer (qTD) completes with a short packet. If system software needs multiple qTDs to complete a client request (i.e. like a control transfer) the intermediate

qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

56.4.4.11 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints. Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The *Status* field has a *Ping State* bit, which the host controller uses to determine the *next* actual PID it will use in the next transaction to the endpoint (see Table 56-46). The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- *EPS* field equals High-Speed and
- *PIDCode* field equals OUT

Table 56-46 illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

Table 56-46. Ping Control State Transition Table

	Event		
Current	Host	Device	Next
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr[1]	Do Ping
Do Ping	PING	Stall	N/C[2] Do
OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping
Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr	Do Ping
Do OUT	OUT	Stall	N/C[2]

1. Transaction Error (XactErr) is any time the host misses the handshake.
2. No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (e.g. Active set to zero and Halt set to a one). Software intervention is required to restart queue. 3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to Do Ping. The Ping State bit has the following encoding:

Value	Meaning
0B	Do OUT The host controller will use an OUT PID during the next bus transaction to this endpoint.
1B	Do Ping The host controller will use a PING PID during the next bus transaction to this endpoint.

The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be *imprecise* on the initialization of the ping protocol (i.e. start in *Do OUT* when we don't know whether there is space on the device or not). The host controller manages the *Ping State* bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the *Ping State* bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the *Ping State* bit is preserved.

56.4.4.12 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs. This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below USB 2.0 hub, utilizing the split transaction protocol. Refer to USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and Low-speed devices are enumerated identically as high-speed devices, but the transactions to the Full- and Low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction protocol is an encapsulation of (or wrapper around) the Full- or Low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full- or Low-speed device is attached.

The EHCI interface uses dedicated data structures for managing full-speed isochronous data streams (see Section Split Transaction Isochronous Transfer Descriptor (siTD)). Control, Bulk and Interrupt are managed using the queuing data structures (see Sections Queue Head). The interface data structures need to be programmed with the device address and the Transaction Translator number of the USB 2.0 Hub operating as the Low-/Full-speed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

56.4.4.12.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an *EPS* field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head. All full-speed bulk and full-, low-speed control are managed via queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (*SplitXState*) to **Do-Start-Split**. Finally, if the endpoint is a control endpoint, then system software must set the *Control Transfer Type* (*C*) bit in the queue head to a one. If this is not a control transfer type endpoint, the *C* bit must be initialized by software to be a zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the *C* bit is a zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the *C* bit is a one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of USB Specification Revision 2.0 for details.

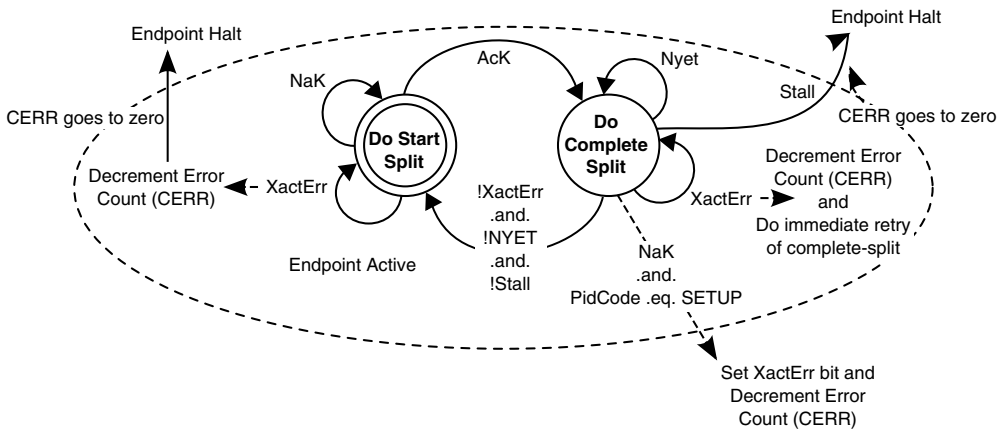


Figure 56-23. Host Controller Asynchronous Schedule Split-Transaction State Machine

56.4.4.12.1.1 Asynchronous-Do Start Split

This is the state which software must initialize a full- or low-speed asynchronous queue head. This state is entered from the **>Do Complete Split** state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller will execute a start-split transaction to the appropriate transaction translator. If the bus transaction completes without an error and *PidCode* indicates an IN or OUT transaction, then the host controller will reload the error counter (*CErr*). If it is a successful bus transaction and the *PidCode* indicates a SETUP, the host controller will not reload the error counter.If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements *Cerr* and proceeds to the next queue head in the asynchronous schedule.

56.4.4.12.1.2 Asynchronous-Do Complete Split

This state is entered from the **Do Start Split state** only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller will execute a complete-split transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's *PidCode* indicates an IN or OUT, the host controller will reload the error counter (*CErr*). When a Nyet handshake is received for a complete-split bus transaction where the queue head's *PidCode* indicates a SETUP, the host controller must not adjust the value of *CErr*.

Independent of *PIDCode*, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, etc. The error counter (*Cerr*) is decremented by one and the complete split transaction is *immediately* retried (if possible). If there is not enough time in the micro-frame to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to accomplish this behavior is to not advance the asynchronous schedule. When the host controller returns to the asynchronous schedule in the next micro-frame, the first transaction from the schedule will be the retry for this endpoint. If *Cerr* went to zero, the host controller must halt the queue.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the *PidCode* is a SETUP, then the Nak response is a protocol error. The *XactErr* status bit is set to a one and the *CErr* field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the *halt* bit in the status byte, retires the qTD but does not attempt to advance the queue.
- If the *PidCode* indicates an IN, then any of following responses are expected:
- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is *good*, the host controller will

advance the state of the transfer, e.g. move the data pointer by the number of bytes received, decrement *BytesToTransfer* field by the number of bytes received, and toggle the *dt* bit. The host controller will then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

- If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the *PidCode* indicates an OUT/SETUP, then any of following responses are expected:
- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount and the data toggle bit (*dt*) is toggled. The host controller will then exit this state.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers via Queue Heads).

56.4.4.12.2 Split Transaction Interrupt

Split-transaction Interrupt-IN/OUT endpoints are managed via the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule. Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, for a high-speed endpoint, the host controller will visit a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the *execution* phase are different (i.e. takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, etc. occurs as defined in Section Managing Control/Bulk/Interrupt Transfers via Queue Heads, but only occurs on the completion of a split transaction.

56.4.4.12.2.1 Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an *EPS* field indicating full- or low-speed and have a non-zero *S-mask* field. The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint will occur. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames, or the data or response information in the pipeline is lost. Figure 56-24 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The S and ^{C}X labels indicate micro-frames where software can schedule start-splits and complete splits (respectively).

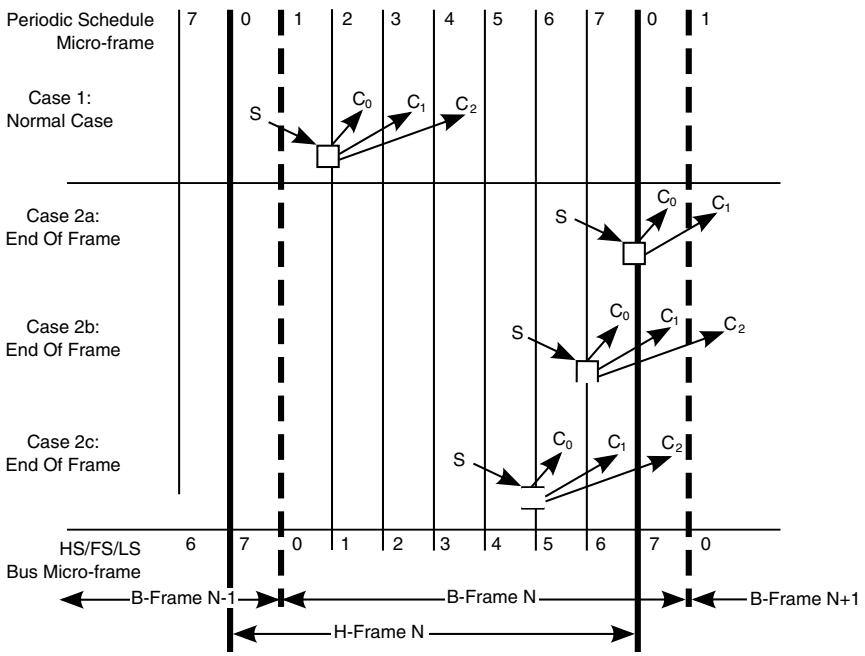


Figure 56-24. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (*H-Frame* in this case).
- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the *H-Frame* boundary when the start-split is in micro-frame 4 or later. When this occurs, the *H-Frame* to *B-Frame* alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs. Figure 56-25 illustrates the general layout of the periodic schedule.

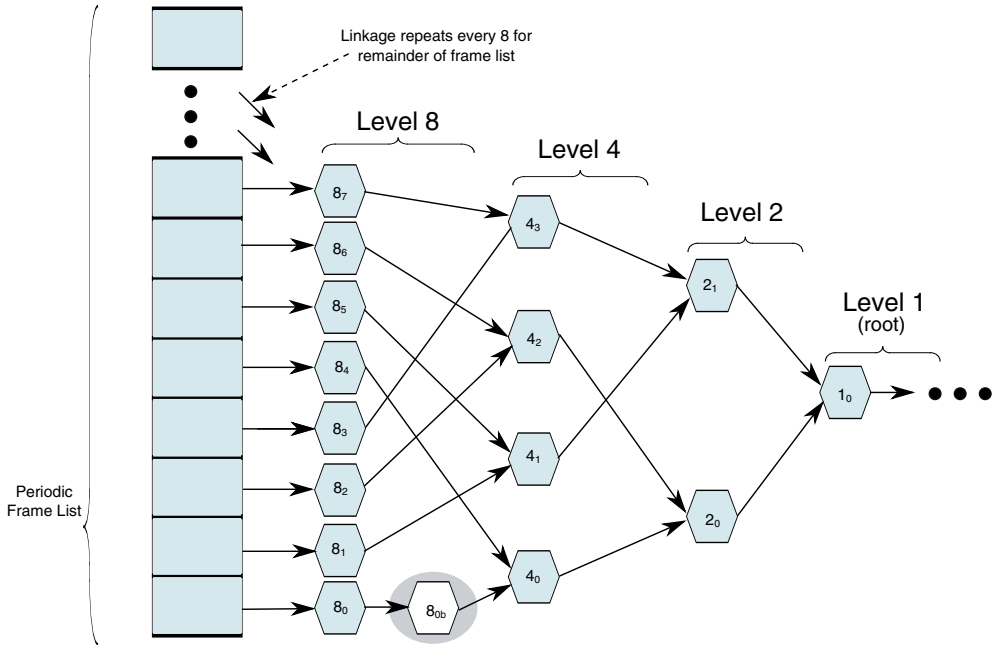


Figure 56-25. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by *spreading* interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} where such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Section Host Controller Operational Model for FSTNs defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of a queue head. Figure 56-24 This bit is used to track the current state of the split transaction.

- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Table 56-24, case one, the *S-mask* would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates **Do_Start**, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 56-24, case one, the *C-mask* would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates **Do_Complete**, and the current micro-frame as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between *H-Frames* and *B-Frames* is correctly performed when setting bits in *S-mask* and *C-mask*

56.4.4.12.2.2 Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (i.e. boundary cases 2a through 2c). An FSTN is essentially a *back pointer*, similar in intent to the back pointer field in the siTD data structure (see Section siTD Back Link Pointer).

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

There are four components to the use of FSTNs:

- FSTN data structure.
- A *Save Place* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to zero.
- A *Restore* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to a one.
- Host controller FSTN traversal rules.

When the host controller encounters an FSTN during micro-frames 2 through 7 it simply follows the node's *Normal Path Link Pointer* to access the next schedule data structure. Note that the FSTN's *Normal Path Link Pointer.T-bit* may set to a one, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a *Save-Place* FSTN in micro-frames 0 or 1, it will save the value of the *Normal Path Link Pointer* and set an internal flag indicating that it is executing in *Recovery Path* mode. *Recovery Path* mode modifies the host controller's rules for how it traverses the schedule and limits which data structures will be considered for execution of bus transactions. The host controller continues executing in *Recovery Path* mode until it encounters a *Restore* FSTN or it determines that it has reached the end of the micro-frame (see details in the list below).

The rules for schedule traversal and limited execution while in *Recovery Path* mode are:

- Always follow the *Normal Path Link Pointer* when it encounters an FSTN that is a *Save-Place* indicator. The host controller must not recursively follow *Save-Place* FSTNs. Therefore, while executing in *Recovery Path* mode, it must never follow an FSTN's *Back Path Link Pointer*.
- Do not process an siTD or, iTD data structure. Simply follow its *Next Link Pointer*.
- Do not process a QH (Queue Head) whose *EPS* field indicates a high-speed device. Simply follow its *Horizontal Link Pointer*.
- When a QH's *EPS* field indicates a Full/Low-speed device, the host controller will only consider it for execution if its *SplitXState* is DoComplete (note: this applies whether the *PID Code* indicates an IN or an OUT). See Sections Execute Transaction and Tracking Split Transaction Progress for Interrupt Transfers for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction. Note that the host controller must not execute a Start-split transaction while executing in *Recovery Path* mode. See Section Periodic Isochronous-Do Complete Split for special handling when in *Recovery Path* mode.
- Stop traversing the *recovery path* when it encounters an FSTN that is a *Restore* indicator. The host controller unconditionally uses the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* when returning to the normal path traversal. The host controller must clear the context of executing a *Recovery Path* when it restores schedule traversal to the *Save-Place* FSTN's *Normal Path Link Pointer*.
- If the host controller determines that there is not enough time left in the micro-frame to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive micro-frame, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in Figure 56-26.

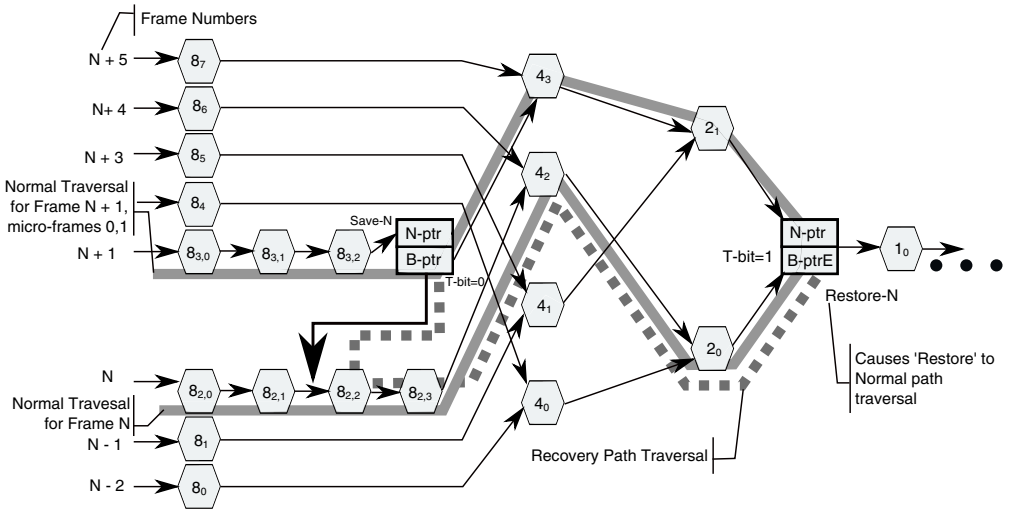


Figure 56-26. Example Host Controller Traversal of Recovery Path via FSTNs

In frame N+1 (micro-frames 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N.Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in *Recovery Path* mode (the recovery path is annotated in Figure 56-26 with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set to a one (definition of a Restore indicator), so the host controller exits *Recovery Path* mode by clearing the internal *Recovery Path* mode flag and commences (restores) schedule traversal using the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* (e.g. Save-N.Normal Path Link Pointer). The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 4_3, 2_1, Restore-N, 1_0 }. The nodes on the recovery-path are bolded.In frame N (micro-frames 0-7), for this example, the host controller will traverse all of the schedule data structures utilizing the *Normal Path Link Pointers* in any FSTNs it encounters. This is because the host controller has not yet encountered a *Save-Place* FSTN so it not executing in *Recovery Path* mode. When it encounters the *Restore* FSTN, (Restore-N), during micro-frames 0 and 1, it uses Restore-N.Normal Path Link Pointer to traverse to the next data structure (i.e. normal schedule traversal). This is because the host controller must use a Restore FSTN's *Normal Path Link Pointer* when not executing in a *Recovery-Path* mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0 }.

In frame N+1 (micro-frames 2-7), when the host controller encounters Save-Path FSTN Save-N, it will unconditionally follow Save-N.Normal Path Link Pointer. The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 }.

56.4.4.12.2.3 Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse. When using FSTNs, system software must adhere to the following rules:

- Each *Save-Place* indicator requires a matching *Restore* indicator.
  - The *Save-Place* indicator is an FSTN with a valid *Back Path Link Pointer and T-bit* equal to zero. Note that *Back Path Link Pointer.Typ* field must be set to indicate the referenced data structure is a queue head. The *Restore* indicator is an FSTN with its *Back Path Link Pointer.T-bit* set to a one.
  - A *Restore* FSTN may be matched to one or more *Save-Place* FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a *Restore* FSTN at the beginning of this list in order to match all possible *Save-Place* FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more *Save-Place* FSTNs are used, then System Software must ensure the *Restore* FSTN's *Normal Path Link Pointer*'s *T-bit* is set to a one, as this will be use to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a *Restore* FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that *Recovery Path* mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A *Save-Place* FSTN's *Back Path Link Pointer* must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list location. In other words, if the *Save-Place* FSTN is reachable from frame list offset N, then the FSTN's *Back Path Link Pointer* must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one *Save-Place* FSTN reachable in any single frame. Note there will be times when two (or more, depending on the implementation) could exist as full/low-speed footprints change with bandwidth adjustments. This could occur, for example when a bandwidth rebalance causes system software to move the *Save-Place* FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

56.4.4.12.2.4 Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost. For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline. When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 Hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets one of the *C-prog*-mask bits for each complete-split executed. The bit position is determined by the micro-frame number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- *FrameTag*. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (*H-Frame* number) when the next complete split must be executed.
- *S-bytes*. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The *S-bytes* field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

56.4.4.12.2.5 Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence. Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- *cMicroFrameBit*. This is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the *FRINDEX* register (i.e. *cMicroFrameBit* = (1 shifted-left(*FRINDEX*[2:0]))). The *cMicroFrameBit* has at most one bit asserted, which always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then *cMicroFrameBit* will equal 00000001b. The variable *cMicroFrameBit* is used to compare against the *S-mask* and *C-mask* fields to determine whether the queue head is marked for a start- or complete-splt transaction for the current micro-frame.

Figure 56-27 illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the *SplitXState* is at **Do_Start** and the single bit in *cMicroFrameBit* has a corresponding bit active in *QH.S-mask*. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to **Do_Complete.** Due to the available jitter in the transaction translator pipeline, there will be more than one complete-split transaction scheduled by software for the **Do_Complete state**. This translates simply to the fact that there are multiple bits set to a one in the *QH.C-mask* field.

The host controller keeps the queue head in the **Do_Complete state** until the split transaction is complete (see definition below), or an error condition triggers the *three-strikes-rule* (e.g. after the host tries the same transaction three times, and each encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

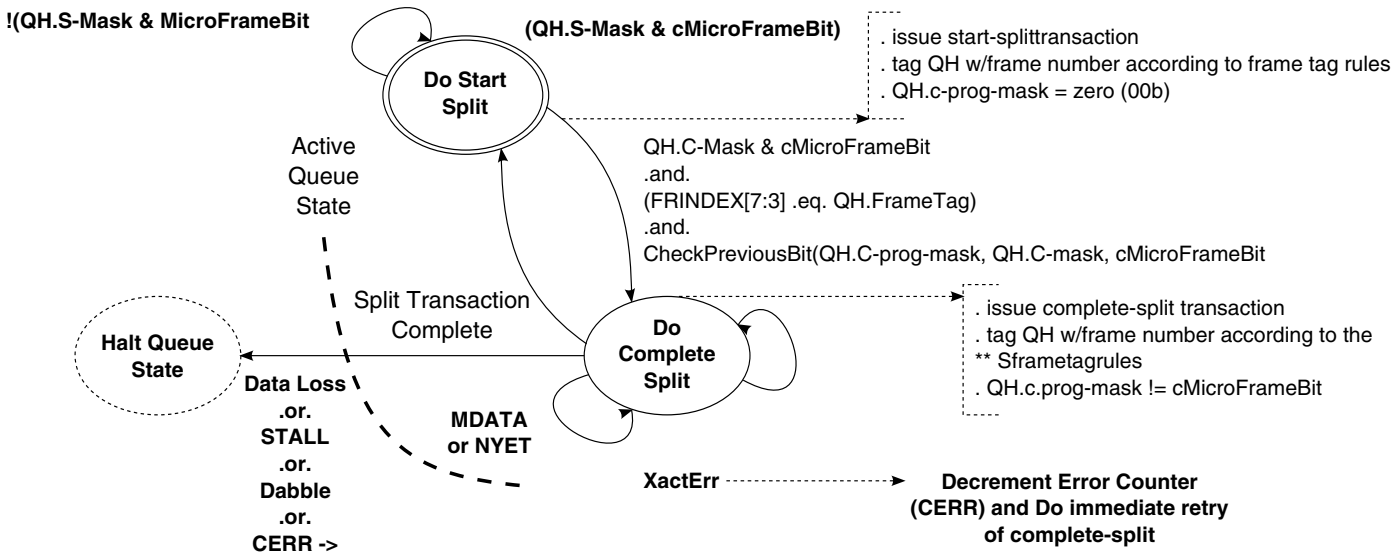


Figure 56-27. Split Transaction State Machine for Interrupt

**See Previous Section for the frame tag management rules.

56.4.4.12.2.6 Periodic Interrupt-Do Start Split

This is the state software must initialize a full- or low-speed interrupt queue head *StartXState* bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (e.g. timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section Periodic Isochronous-Do Complete Split for the definition of 'Last'.

Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it performs the following test to determine whether to execute a start-split.

- *QH.S-mask* is bit-wise anded with *cMicroFrameBit*.

If the result is non-zero, then the host controller will issue a start-split transaction. If the *PIDCode* field indicates an IN transaction, the host controller must zero-out the *QH.S-bytes* field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into *QH.FrameTag* field, set *C-prog-mask* to zero (00h), and exits this state. Note that the host controller must not adjust the value of *CErr* as a result of completion of a start-split transaction.

56.4.4.12.2.7 Periodic Interrupt- Do Complete Split

This state is entered unconditionally from the Do Start Split state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- Test A. *cMicroFrameBit* is bit-wise anded with *QH.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame.
- Test B. *QH.FrameTag* is compared with the current contents of *FRINDEX*[7:3]. An equal indicates a match.
- Test C. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

```
        Algorithm Boolean CheckPreviousBi
        t(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)
Begin
-- Return values:
-- TRUE - no error
-- FALSE - error
--
Boolean rvalue = TRUE;
previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous micro-frame. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                              If not(previousBit bitAND QH.C-prog-mask) then
                    rvalue = FALSE;
          End if
End If
-- If the C-prog-mask already has a one in this bit position,
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
```

```
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
 rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- Test D. Check to see if a start-split should be executed in this micro-frame. Note this is the same test performed in the Do Start Split state (see Section Periodic Isochronous-Do Start Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a *Recovery Path* mode, it means a start-split should occur in this micro-frame. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets *QH.FrameTag* to the expected *H-Frame* number. The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (note that any responses that result in decrementing of the *CErr* will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.
- The test for whether this is the Last complete split can be performed by XOR *QH.C-mask* with *QH.C-prog-mask*. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the *XactErr* status bit is set to a one and the *CErr* field is decremented.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask* and *FrameTag*) and stay in this state. The host controller must not adjust *CErr* on this response.
- Transaction Error (XactErr). Timeout, data CRC failure, etc. The *CErr* field is decremented and the *XactErr* bit in the *Status* field is set to a one. The complete split transaction is *immediately* retried (if *Cerr* is non-zero).If there is not enough time in the micro-frame to complete the retry and the endpoint is an IN, or *CErr* is

decremented to a zero from a one, the queue is halted. If there is not enough time in the micro-frame to complete the retry and the endpoint is an OUT and *CErr* is not zero, then this state is exited (i.e. return to Do Start Split). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section full- and low-speed Interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.

- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount. And the data toggle bit (*dt*) is toggled. The host controller will then exit this state for this queue head. The host controller must reload *CErr* with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers via Queue Heads).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in *QH.S-bytes*. The host controller must not adjust *CErr* on this response.
- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in *QH.S-bytes*. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers via Queue Heads).
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload *CErr* with maximum value on this response.
- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set to a one, *Cerr* is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the Execute Transaction state). The status field bits: *Active* bit is set to zero and the *Halted* bit is set to a one and the qTD is retired. Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The

other possible combinations of tests A, B, C, and D may indicate that data or response was lost. Table 56-47 lists the possible combinations and the appropriate action.

Table 56-47. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current micro-frame.Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	*QH.FrameTag* test failed. This means that exactly one or more *H-Frames* have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one. Note: When executing in the context of a *Recovery Path* mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a *Recovery Path* mode.

56.4.4.12.2.8 Managing QH.FrameTag Field

The *QH.FrameTag* field in a queue head is completely managed by the host controller. The rules for setting *QH.FrameTag* are simple:

- **Rule 1:** If transitioning from **Do Start Split to Do Complete Split** and the current value of *FRINDEX*[2:0] is 6 *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different *H-Frames* (case 2a, see Figure 56-24).
- **Rule 2:** If the current value of *FRINDEX*[2:0] is 7, *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates staying in **Do Complete Split** for cases 2a, 2b, and 2c (Figure 56-24).
- **Rule 3:** If transitioning from **Do_Start Split to Do Complete Split** and the current value of *FRINDEX*[2:0] is not 6, or currently in **Do Complete Split** and the current value of (*FRINDEX*[2:0]) is not 7, *FrameTag* is set to *FRINDEX*[7:3]. This accommodates all other cases (Figure 56-24).

56.4.4.12.2.9 Rebalancing the Periodic Schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation. This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (i.e. new S-mask and C-mask values).

It is imperative that System software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the *Inactivate-on-next-Transaction* (*I*) bit to a one to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software will then wait for the host controller to observe the *I-bit* is a one and transition the *Active* bit to a zero. The rules for how and when the host controller sets the *Active* bit to zero are enumerated below:

- If the *Active* bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the *I-bit* is a one.
- If the *Active* bit is a one and the *SplitXState* is DoStart (regardless of the value of *S-mask*), the host controller will simply set *Active* bit to a zero. The host controller is not required to write the transfer state back to the *current* qTD. Note that if the *S-mask* indicates that a start-split is scheduled for the current micro-frame, the host controller must not issue the start-split bus transaction. It must set the *Active* bit to zero.

System software must save transfer state before setting the *I-bit* to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the *I-bit* was set to a one and the host controller executed it's final bus-transaction and set *Active* to a zero.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Since the *Active* bit and the *I-bit* cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped via the *I-bit*.

1. Set the *Halted* bit to a one, then
2. Set the *I-bit* to a zero, then
3. Set the *Active* bit to a one and the *Halted* bit to a zero in the same write.

Setting the *Halted* bit to a one inhibits the host controller from attempting to advance the queue between the time the *I-bit* goes to a zero and the *Active* bit goes to a one.

56.4.4.12.3 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous split-transactions. This data structure uses the scheduling model of isochronous TDs (iTD, Section Isochronous (High-Speed) Transfer Descriptor (iTD)) (see Section Managing Isochronous Transfers Using iTDs for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

56.4.4.12.3.1 Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur. The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. Figure 56-28 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The ^{S}X and ^{C}X labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The *H-Frame* boundaries are marked with a large, solid bold vertical line. The *B-Frame* boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the *H-Frame*.

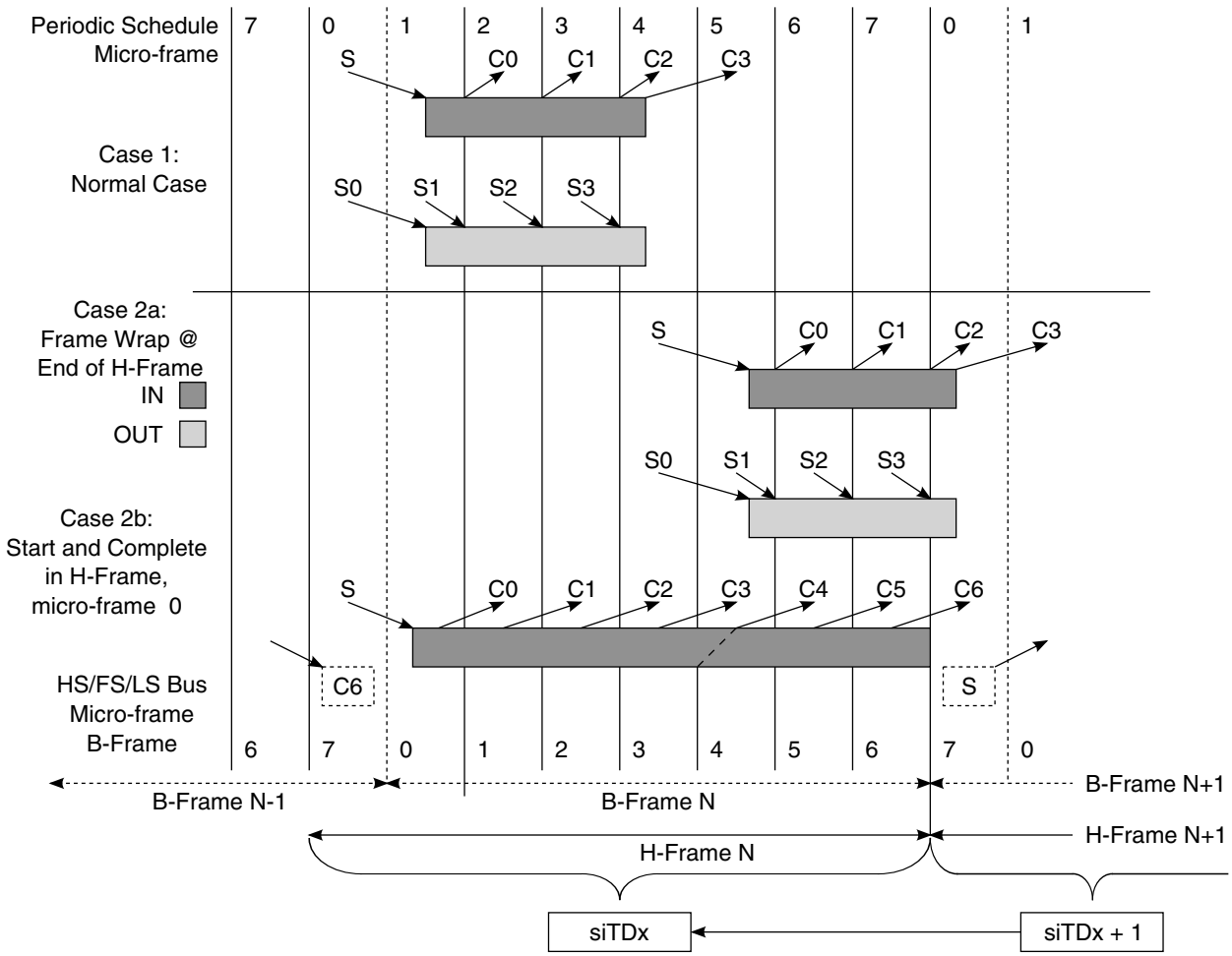


Figure 56-28. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to *N* complete-splits. The scheduling boundary cases are:

- *Case 1*: The entire split transaction is completely bounded by an *H-Frame*. For example: the start-splits and complete-splits are all scheduled to occur in the same *H-Frame*.
- *Case 2a*: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an *H-Frame* boundary. This can only occur when the split transaction has the possibility of moving data in *B-Frame,* micro-frames 6 or 7 (*H-Frame* micro-frame 7 or 0). When an *H-Frame* boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(e.g. it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction).
- Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one

data pointer. For this reason, siTDs contain a back pointer, the use of which is described below.
- Software must never schedule full-speed isochronous OUTs across an *H-Frame* boundary.
- *Case 2b*: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of an siTD. This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section Split Transaction Execution State Machine for Interrupt .
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 56-28, case one, the *S-mask* would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do Start Split, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 56-28, case one, the *C-mask* would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do Complete Split, and the current micro-frame as indicated by *FRINDEX*[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.
- *Back Pointer*. This field in a siTD is used to complete an IN split-transaction using the previous *H-Frame*'s siTD. This is only used when the scheduling of the complete-splits span an *H-Frame* boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data

buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the *H-Frame* boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. Figure 56-29 illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the *B-Frames* (HS/FS/LS Bus) and the *H-Frames*. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each *H-Frame* corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

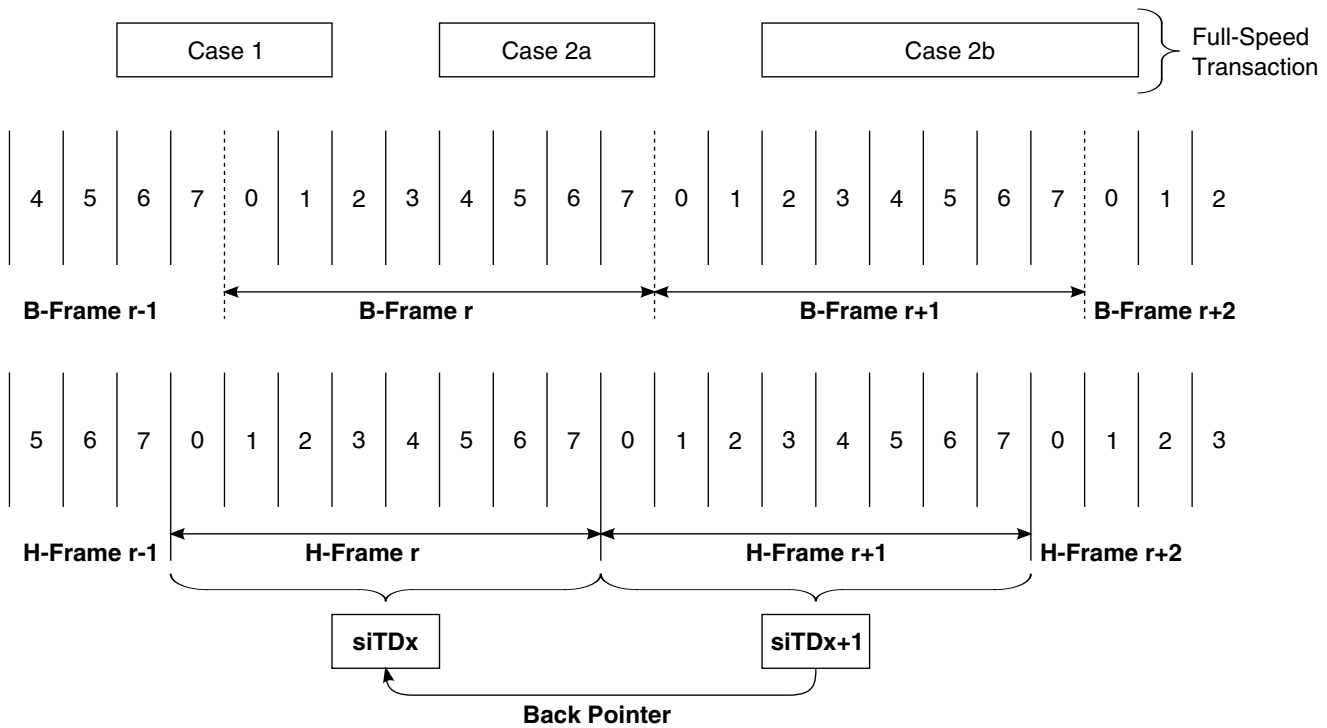


Figure 56-29. siTD Scheduling Boundary Examples

Each case is described below:

- *Case 1*: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single *H-Frame*.
- *Case 2a, 2b*: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction siTDX is used to always issue the start-split and the first *N* complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during

micro-frame 7 of *H-Frame*$_{Y+1}$, or micro-frame 0 of *H-Frame*$_{Y+2.}$ The complete splits are scheduled using siTD$_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from siTD$_{X+1}$. The only way for the host controller to reach siTD$_{X+1}$ from *H-Frame*$_{Y+2}$ is to use siTD$_{X+2}$'s back pointer. The host controller rules for when to use the back pointer are described is Section Periodic Isochronous-Do Complete Split .

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the *B-Frame*.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in *H-Frame, micro-frame 1*. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in micro-frame 1 of *H-Frame* N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

56.4.4.12.3.2 Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped micro-frames), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable via the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD via the fields *Transaction Position* (*TP*) and *Transaction Count* (*T-*

count). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section Periodic Isochronous-Do Start Split for a description on how these fields are used during a sequence of start-split transactions.

The fields *siTD.T-Count* and *siTD.TP* are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction isochronous endpoint is established, *S-mask*, *T-Count*, and *TP* initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the micro-frame (FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's *Active* bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. Refer to Section Asynchronous-Do Complete Split for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired based solely on the responses from the Transaction Translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD field *Total Bytes to Transfer* to decrement to zero.

This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (e.g. high-speed data streams through queue heads), the transition of *Total Bytes to Transfer* to zero signals the end of the transfer and results in setting of the *Active* bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a micro-frame boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and give the remaining bytes to the high-speed pipeline stage. At the micro-frame boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next micro-frame, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement it's *Total Bytes to Transfer* field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the *C-prog-mask* is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (e.g. a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (e.g. state not advanced) and report the appropriate error to the client driver.

56.4.4.12.3.3 Split Transaction Execution State Machine for Isochronous

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

If the *Active* bit in the *Status* byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in Section Tracking Split Transaction Progress for Interrupt Transfers , plus the variable

cMicroFrameBit defined in Section Split Transaction Execution State Machine for Interrupt to track the progress of an isochronous split transaction. Figure 56-30 illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the *Active* bit in the *Status* field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

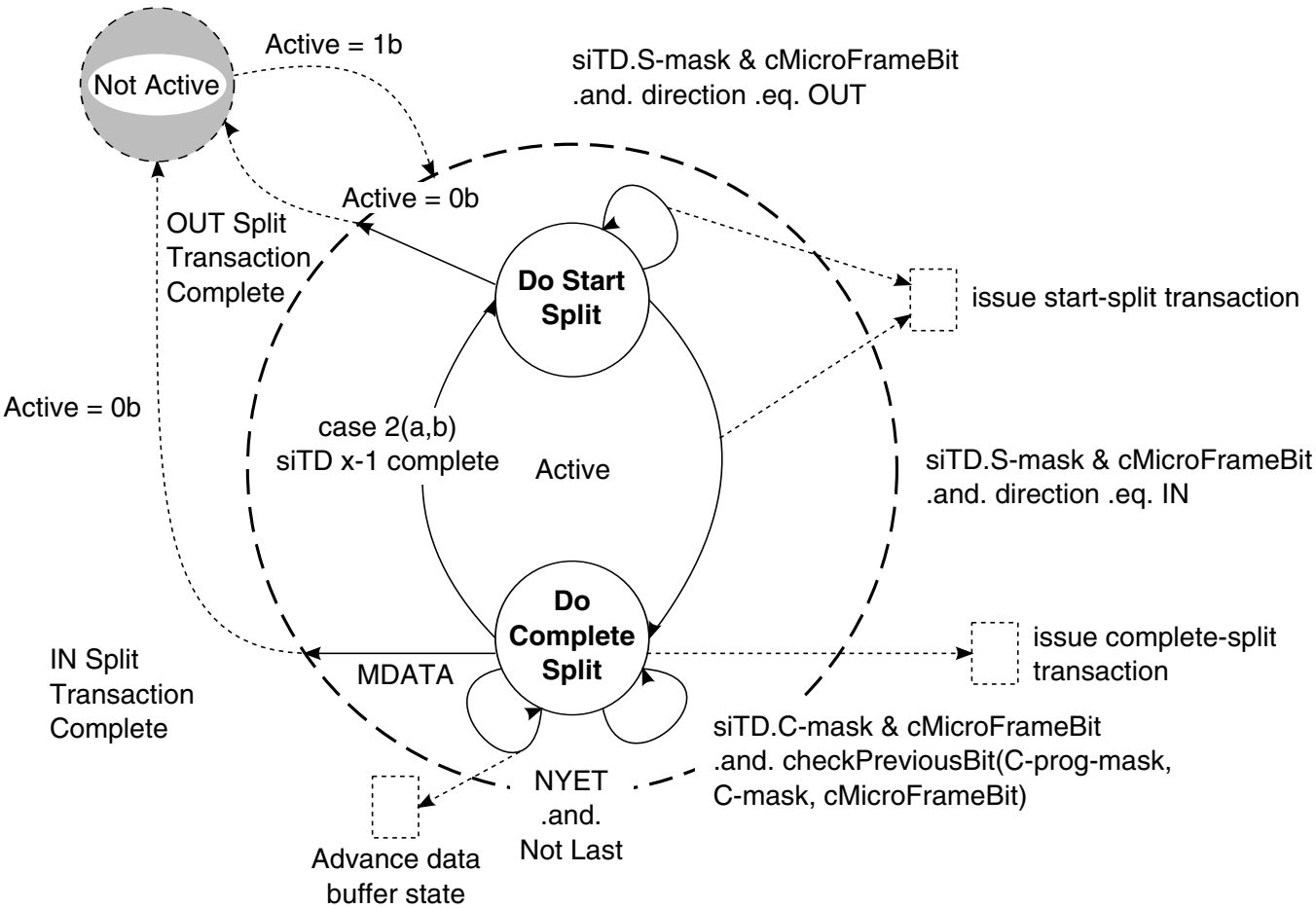


Figure 56-30. Split Transaction State Machine for Isochronous

56.4.4.12.3.4 Periodic Isochronous-Do Start Split

Isochronous split transaction OUTs use only this state. An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from Do Complete Split when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the *siTD.S-mask* against *cMicroFrameBit*. If there is a one in the appropriate position, the siTD will execute a start-split transaction. By definition, the host controller cannot *reach* an siTD at the wrong time. If the *I/O* field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the *siTD.Total Bytes To Transfer* field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (i.e. the *I/O* field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory buffer address for the data payload is constructed by concatenating *siTD.Current Offset* with the page pointer indicated by the page selector field (*siTD.P*). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the *siTD.P*-bit from a zero to a one, and begin using the *siTD.Page 1* with *siTD.Current Offset* as the memory address pointer. The field *siTD.TP* is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in *siTD.TP* to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. *TP* is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 56-29) is used to determine the initial value of *TP*. The initial cases are summarized in Table 56-48.

Table 56-48. Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates *T-Count* and *TP* appropriately so that the next start-split is correctly annotated. Table 56-49 illustrates all of the *TP* and *T-count* transitions, which must be accomplished by the host controller.

.

Table 56-49. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when *T-count* starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when *T-count* starts at greater than 2.
MID	!=1	MID	*TP* stays at MID while *T-count* is not equal to 1 (e.g. greater than 1). This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The *siTD.Total Bytes To Transfer* and the *siTD.Current Offset* fields are adjusted to reflect the number of bytes transferred.
- The *siTD.P* (page selector) bit is updated appropriately.
- The *siTD.TP* and *siTD.T-count* fields are updated appropriately.

These fields are then written back to the memory based siTD. The *S-mask* is fixed for the life of the current budget. As mentioned above, *TP* and *T-count* are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of *S-mask*, the actual number of start-split transactions depends on *T-count* (or equivalently, *Total Bytes to Transfer*). The host controller must set the *Active* bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when *T-Count* decrements to zero as a result of a start-split bus transaction. Equivalently, the host controller can detect when *Total Bytes to Transfer* decrements to zero. Either implementation must ensure that if the initial condition is

Total Bytes to Transfer equal to zero and *T-count* is equal to a one, then the host controller will issue a single start-split, with a zero-length data payload. Software must ensure that *TP*, *T-count* and *Total Bytes to Transfer* are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the start-splits for the OUT endpoint (i.e. the transaction position annotation will be incorrect as received by the transaction translator).

Example scenarios are described in Section Split Transaction for Isochronous - Processing Examples .

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the *siTD.C-prog-mask* as it executes each scheduled start-split. The *checkPreviousBit*() algorithm defined in Section Periodic Isochronous-Do Complete Split can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed micro-frames. It can then set the siTD's *Active* bit to zero and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

56.4.4.12.3.5 Periodic Isochronous-Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint. This state is entered unconditionally from the Do Start State after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied depends on which micro-frame the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- **Test A.***cMicroFrameBit* is bit-wise anded with *siTD.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame. This test is always applied to a newly fetched siTD that is in this state.
- **Test B.** The *siTD.C-prog-mask* bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Section Periodic Isochronous-Do Complete Split). The sequence in which this test is applied depends on the current value of FRINDEX[2:0]. If FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

```
Algorithm Boolean CheckPreviousBit(siTD.C-prog-mask, siTD.C-mask, cMicroFrameBit)
```

```
Begin
          Boolean rvalue = TRUE;
          previousBit = cMicroFrameBit rotate-right(1)
          -- Bit-wise anding previousBit with C-mask indicates whether there was an intent
          -- to send a complete split in the previous micro-frame. So, if the
          -- 'previous bit' is set in C-mask, check C-prog-mask to make sure it
          -- happened.
          if previousBit bitAND siTD.C-mask then
                    if not (previousBit bitAND siTD.C-prog-mask) then
                              rvalue = FALSE
                    End if
          End if
          Return rvalue
End Algorithm
```

If Test A is true and FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (see Figure 56-28). See Section Periodic Isochronous-Do Complete Split for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from *siTD.Total Bytes To Transfer,*
- Adjust *siTD.Current Offset* by the number of bytes received,
- Adjust *siTD.P* (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the *siTD.Status* field, depending on the results of the transaction.

Note that if the host controller encounters a condition where *siTD.Total Bytes To Transfer* is zero, and it receives more data, the host controller must not write the additional data to memory. The *siTD.Status.Active* bit must be set to zero and the *siTD.Status.Babble Detected* bit must be set to a one. The fields *siTD.Total Bytes To Transfer*, *siTD.Current Offset*, and *siTD.P* (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of *siTD.Total Bytes To Transfer*) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set *siTD.Status Active* to a zero and *siTD.Status.Babble Detected* to a one when it receives and MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the *ERR* bit in the *siTD.Status* field and sets the *Active* bit to a zero.

- Transaction Error (XactErr). The complete-split transaction encounters a Timeout, CRC16 failure, etc. The *siTD.Status* field *XactErr* field is set to a one and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the micro-frame occurs, the *Active* bit is set to zero.
- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the *Active* bit is set to a zero. If the *Bytes To Transfer* field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This *short packet* event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.
- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in Section Periodic Isochronous-Do Complete Split . If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the *Active* bit is set to a zero. No bits are set in the *Status* field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.
- MDATA (and Last). See above description for testing for Last. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the *S-mask* and/or *C-mask*s incorrectly. The host controller must set *XactErr* bit to a one and the *Active* bit is set to a zero.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask*) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from micro-frame X to X+1 and during micro-frame X, the transaction translator will respond with an MDATA and the data accumulated up to the end of micro-frame X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the *Missed Micro-Frame* status bit and sets the *Active* bit to a zero.

56.4.4.12.3.6 Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 56-28) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. Table 56-50 enumerates the transaction state fields.

Table 56-50. Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

NOTE

TP and *T-count* are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the *siTD.Back Pointer* field to reference a valid siTD and will have the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field set to a zero. Otherwise, software must set the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field to a one. The host controller's rules for interpreting when to use the *siTD.Back Pointer* field are listed below. These rules apply only when the siTD's *Active* bit is a one and the *SplitXState* is Do Complete Split.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.T-bit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.S-mask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from $siTD_{X-1}$.

In order to access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$.*Back Pointer*.

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of *siTD.Back Pointers*.

If *siTD*$_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is **Do Complete Split**), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state of si*TD*$_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of *siTD*$_{X-1}$'s *Active* bit is a one, then the host controller returns to the context of *siTD*$_X$, and follows its next pointer to the next schedule item. No updates to *siTD*$_X$ are necessary.

If *siTD*$_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is **Do Start Split**), then the host controller must set *Active* bit to a zero and *Missed Micro-Frame* status bit to a one and the resultant status written back to memory.

If *siTD*$_{X-1}$'s *Active* bit is a zero, (because it was zero when the host controller first visited *siTD*$_{X-1}$ via *siTD*$_X$'s back pointer, it transitioned to zero as a result of a detected error, or the results of *siTD*$_{X-1}$'s complete-split transaction transitioned it to zero), then the host controller returns to the context of *siTD*$_X$ and transitions its *SplitXState* to **Do Start Split**. The host controller then determines whether the case 2b start split boundary condition exists (i.e. if *cMicroframeBit* is a 1b and *siTD*$_X$.*S-mask*[0] is a 1b). If this criterion is met the host controller immediately executes a start-split transaction and appropriately advances the transaction state of *siTD*$_X$, then follows *siTD*$_X$.*Next Pointer* to the next schedule item. If the criterion is not met, the host controller simply follows *siTD*$_X$.*Next Pointer* to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of *siTD*$_{X-1}$ will have its *Active* bit set to zero when the host controller returns to the context of *siTD*$_X$. Also, note that software should not initialize an siTD with *C-mask* bits 0 and 1 set to a one and an *S-mask* with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

56.4.4.12.3.7 Split Transaction for Isochronous - Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines. The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced via the **Execute Transaction** queue head traversal state machine (see Figure 56-21).

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, Table 56-51 illustrates a couple of frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 56-51. Example Case 2a - Software Scheduling siTDs for an IN Endpoint

siTDX		Micro-Frames								Initial
#	Masks	0	1	2	3	4	5	6	7	SplitXState
X	S-Mask					1				Do Start Split
	C-Mask	1	1					1	1	
X+1	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X+2	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X+3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Since this is the case-2a frame-wrap case, *S-mask*s of all siTDs for this endpoint have a value of 10h (a one bit in micro-frame 4) and *C-mask* value of C3h (one-bits in micro-frames 0,1, 6 and 7). Additionally, sofware ensures that the *Back Pointer* field of each siTD references the appropriate siTD data structure (and the *Back PointerT-bits* are set to zero).

The initial *SplitXState* of the first siTD is **Do Start Split** . The host controller will visit the first siTD eight times during frame X. The C-mask bits in micro-frames 0 and 1 are ignored because the state is **Do Start Split**. During micro-frame 4, the host controller determines that it can run a start-split (and does) and changes *SplitXState* to **Do Complete Split**. During micro-frames 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has it's *SplitXState* initialized to **Do Complete Split**. As the host controller continues to traverse the schedule during *H-Frame* X+1, it will visit the second siTD eight times. During micro-frames 0 and 1 it will detect that it must execute complete-splits.

During *H-Frame* X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's *Active* bit is set to zero and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the *SplitXState* in the $siTD_{X+1}$ to **Do Start Split**. At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches micro-frame 4. If the split-transaction completes

early (transaction-complete is defined in Section Periodic Isochronous-Do Complete Split), i.e. before all the scheduled complete-splits have been executed, the host controller will transition *siTD*$_X$.*SplitXState* to **Do Start Split** early and naturally skip the remaining scheduled complete-split transactions. For this example, siTD$_{X+1}$ does not receive a DATA0 response until *H-Frame* X+2, micro-frame 1.

During *H-Frame X+2,* micro-frame 0, the host controller detects that siTD$_{X+2}$'s *Back Pointer.T-bit* is a zero, saves the state of siTD$_{X+2}$ and fetches siTD$_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the *Active* bit. The host controller returns to the context of siTD$_{X+2}$, and traverses it's next pointer without any state change updates to siTD$_{X+2}$. S

During *H-Frame X+2*, micro-frame 1, the host controller detects siTD$_{X+2}$'s *S-mask[0]* is a zero, saves the state of siTD$_{X+2}$ and fetches siTD$_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and sets the *Active* bit to a zero. It returns to the state of siTD$_{X+2}$ and changes its *SplitXState* to Do Start Split. At this point, the host controller is prepared to execute start-splits for siTD$_{X+2}$ when it reaches micro-frame 4. <TBD… describe how software detects that there was missing micro-frames (don't think we care about missing out micro-frames. There is enough residual state to identify than not all transactions were executed.).

56.4.4.13 Host Controller Pause

When the host controller's *HCHalted* bit in the USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports. When the schedules are enabled, the EHCI host controller will access the schedules in main memory each micro-frame. This constant pinging of main memory is known to create CPU power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the CPU, based on recent history usage. In the more aggressive power saving modes, the CPU can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the CPU power management software can detect this activity over time and inhibit the transition of the CPU into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the CPU power management software from placing the CPU into its lowest power savings state.

USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the CPU power management to get the CPU into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the CPU to reach it's low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the CPU will be required to process the data streams.

It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing it's shared memory based data structures (schedule lists or otherwise).

56.4.4.14 Port Test Modes

EHCI host controllers must implement the port test modes **Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SE0_NAK** as described in the USB Specification Revision 2.0. The system is only allowed to test ports that are owned by the EHCI controller (e.g. *CF-bit* is a one and *PortOwner* bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the *CF-bit* in the CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the *Asynchronous Schedule Enable* and *Periodic Schedule Enable* bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the *Suspend* bit in each appropriate PORTSC register to a one.

- Set the *Run/Stop* bit in the USBCMD register to a zero and wait for the *HCHalted* bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the *Run/Stop* bit set to a one. However, all host controllers must support port testing with *Run/Stop* set to a zero and *HCHalted* set to a one.
- Set the *Port Test Control* field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is **Test_Force_Enable**, then the *Run/Stop* bit in the USBCMD register must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (*HCHalted* bit is a one) then it terminates and exits test mode by setting *HCReset* to a one.

56.4.4.15 EHCI Interrupts

The EHCI Host Controller hardware provides interrupt capability based on a number of sources. There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, and so on), and
- Host Controller error events

All transaction-based sources are maskable through the Host Controller's Interrupt Enable register (USBINTR, see Section Interrupt Enable Register (USB_USBINTR)). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the *Interrupt Threshold Control* field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight micro-frames. This means that the host controller will not generate interrupts any more frequently than once every eight micro-frames.

Section Host System Error details effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, CPU control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

Note: the host controller is not required to de-assert a currently active interrupt condition when software sets the interrupt enables (in the USBINR register to a zero. The only reliable method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register from a one to a zero.

56.4.4.15.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

56.4.4.15.1.1 Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully. Table 56-52 lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the *XactErr* status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the *XactErr* status bit in the queue head is set and the *CErr* field is decremented. When the *PIDCode* indicates a SETUP, the following responses are protocol errors and result in *XactErr* bit being set to a one and the *CErr* field being decremented.

- *EPS* field indicates a high-speed device and it returns a Nak handshake to a SETUP.

- *EPS* field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- *EPS* field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Table 56-52. Summary of Transaction Errors

Event / Result	Queue Head/qTD/iTD/siTD Side-effects		USB Status Register (USBSTS)
	Cerr	Status Field	USBERRINT
CRC	-1	XactErr set to a one.	1[1]
Timeout	-1	XactErr set to a one.	1[1]
Bad PID[2]	-1	XactErr set to a one.	1[1]
Babble	N/A	Section Serial Bus Babble	1
Buffer Error	N/A	Section Data Buffer Error	

[1] If occurs in a queue head, then *USBERRINT* is asserted only when *CErr* counts down from a one to a zero. In addition the queue is halted, see Section Halting a Queue Head .

[2] The host controller received a response from the device, but it could not recognize the PID as a valid PID.

56.4.4.15.1.2 Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a *Packet Babble*. When a device sends more data than the *Maximum Length* number of bytes, the host controller sets the *Babble Detected* bit to a one and halts the endpoint if it is using a queue head (see Halting a Queue Head). *Maximum Length* is defined as the minimum of *Total Bytes to Transfer* and *Maximum Packet Size*. The *CErr* field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The *USBERRINT* bit in the USBSTS register is set to a one and if the *USB Error Interrupt Enable* bit in the USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that will babble across a micro-frame EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on *Maximum Packet Size*. The USB core specification defines

the requirements on a data receiver when it receives a data PID mismatch (e.g. expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence. The EHCI interface allows System software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device will re-send its maximum packet size data packet, with the original data PID, in response to the next IN token. In order to properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

56.4.4.15.1.3 Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction. This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the *Data Buffer Error* bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

56.4.4.15.1.4 USB Interrupt (Interrupt on Completion (IOC))

Transfer Descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USBSTS register to be set to a one. In addition, if a short packet is encountered on an IN transaction associated

with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USBINTR register is set to a one, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the *USBERRINT* bit in the USBSTS register is also set to a one.

56.4.4.15.1.5 Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the *USBINT* bit in the USBSTS register is set to a one. If the *USB Interrupt Enable* bit is set in the USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

56.4.4.15.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Interrupt on Async Advance).

56.4.4.15.2.1 Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the *Port Change Detect* bit in the USBSTS register to a one. If the *Port Change Interrupt Enable* bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

56.4.4.15.2.2 Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs. If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, etc. When a frame list rollover is detected, the host controller sets the *Frame List Rollover* bit in the USBSTS register to a one. If the *Frame List Rollover Enable* bit in the USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

56.4.4.15.2.3 Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the *Interrupt on Async Advance Doorbell* bit in the USBCMD register. If it is a one, it sets the *Interrupt on Async Advance* bit in the USBSTS register to a one. If the *Interrupton Async Advance Enable* bit in the USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section Removing Queue Heads from Asynchronous Schedule .

56.4.4.15.2.4 Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors. The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion. In the presence of non-catastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Host-based error must result in the following actions:

- The *Run/Stop* bit in the USBCMD register is set to a zero.
- The following bits in the USBSTS register are set:
  - *Host System Error* bit is to a one.
  - *HCHalted* bit is set to a one.
- If the *Host System Error Enable* bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. Table 56-53 summarizes the required actions taken on the various host errors.

Table 56-53. Summary Behavior of EHCI Host Controller on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal [o]
siTD fetch (read)	Fatal	Fatal	Fatal [o]
siTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
iTD fetch (read)	Fatal	Fatal	Fatal [o]
iTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
qTD fetch (read)	Fatal	Fatal	Fatal [o]
qHD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
Data write	Fatal [o]	Fatal [o]	Fatal [o]
Data read	Fatal	Fatal	Fatal [o]

[o] Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

NOTE

After a *Host System Error*, Software must reset the host controller via *HCReset* in the USBCMD register before re-initializing and restarting the host controller.

56.4.5 EHCI Deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specificationEnhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation & On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary. The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator-Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation-In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface-This core does not a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.
- On-The-Go Operation-This design includes an On-The-Go controller for Port #1.

56.4.5.1 Embedded Transaction Translator Function

The ARC USB-HS OTG High-Speed USB On-The-Go OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction trasnslator. Although there is no separate Tranaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator fuction is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

56.4.5.1.1 Capability Registers

The following additions have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT added to HCSPARAMS-Host Control Structural Parameters
- N_PTT added toHCSPARAMS-Host Control Structural Parameters

56.4.5.1.2 Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- is a new register.
- Addition of two-bit Port Speed (PSPD) to the Port Status and Control (USB_PORTSC) register.

56.4.5.1.3 Discovery of FS or LS

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation. The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (that is, Chirp completes successfully).

Since this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in Table 56-54

Table 56-54. Summary of EHCI

Standard EHCI	EHCI with embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using PORTSCx.

Table continues on the next page...

Table 56-54. Summary of EHCI (continued)

Standard EHCI	EHCI with embedded Transaction Translator
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. **[where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (ie. Split target hub)]**	FS and LS device can be either downstream from a HS hub with HubAddr = **X [HubAddr > 0]** or directly attached **[where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (ie. Split target hub is the root hub)]**

56.4.5.1.4 Data Structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator. Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS) - Async. (Bulk/Control Endpoints) Periodic (Interrupt)
    - Hub Address = 0
    - Transactions to direct attached device/hub.
        - QH.EPS = Port Speed
    - Transactions to a device downstream from direct attached FS hub.
        - QH.EPS = Downstream Device Speed
            - Maximum Packet Size must be less than or equal 64 or undefined behaviour may result.
2. siTD (for direct attach FS) - Periodic (ISO Endpoint)
    - All FS ISO transactions:
        - Hub Address = 0
        - siTD.EPS = 00 (full speed)
            - Maximum Packet Size must less than or equal to 1023 or undefined behaviour may result.

NOTE

When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

56.4.5.1.5 Operational Model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specificationUniversal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Since the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator

operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

56.4.5.1.5.1 Micro- frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

56.4.5.1.5.2 Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator. Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. Table 56-55 summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 56-55. Summary of the conditons of handshakes

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK
Start-Split: Start Periodic Transaction.	No Handshake (Ok)

Table continues on the next page...

Table 56-55. Summary of the conditons of handshakes (continued)

Condition	Emulate TT Response
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

NOTE

The un-shaded cells represent Start-Splits and the shaded cells represent Complete-Splits.

56.4.5.1.5.3 Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0 - 11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

USB 2.0 - 11.17.4

- Transaction tracking for 2 data pipes.

USB 2.0 - 11.17.5

- Clear_TT_Buffer capability provided though the use of the register.

56.4.5.1.5.4 Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0 - 11.18.6.[1-2]

- Abort of pending start-splits
  - EOF (and not started in micro-frames 6)
  - Idle for more than 4 micro-frames
- Abort of pending complete-splits
  - EOF
  - Idle for more than 4 micro-frames

USB 2.0 - 11.18.[7-8]

- Transaction tracking for up to 16 data pipes.

- Some applications may not require transaction tracking up to a maximum of 16 periodic data pipes. The option to limit the tracking to only 4 periodic data pipes exists in the by changing the configuration constant VUSB_HS_TT_PERIODIC_CONTEXTS to 4. The result is a significant gate count savings to the core given the limitations implied.
- Complete-split transaction searching.

NOTE

Limiting the number of tracking pipes in the EMBedded -TT to four (4) will impose the restriction that no more than 4 periodic transactions (INTERRUPT/ISOCHRONOUS) can be scheduled through the embedded-tt per frame. the number 16 was chosen in the USB specification because it is sufficient to ensure that the high-speed to full-speed periodic pipeline can remain full. keeping the pipeline full puts no constraint on the number of periodic transactions that can be scheduled in a frame and the only limit becomes the flight time of the packets on the bus.

NOTE

There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

56.4.5.1.5.5 Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the Host Controller Structural Parameters (USB_HCSPARAMS - EHCI Compliant) register.

56.4.5.2 Device Operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

56.4.5.3 USBMODE Register

Given that the dual-role controller is initialized in neither host nor device mode, the USB Device Mode (USB_USBMODE) register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

56.4.5.3.1 Non-Zero Fields the Register File

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode .

56.4.5.3.2 SOF Interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode. EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USB Status Register (USB_USBSTS) and Interrupt Enable Register (USB_USBINTR) registers.

56.4.5.4 Embedded Design Interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

56.4.5.4.1 Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. i.e. 60 Mhz transceiver clock for 8-bit physical interfaces & full-speed serial interfaces or 30 Mhz transceiver clock for 16-bit physical interfaces.

56.4.5.5 Miscellaneous Variations from EHCI

56.4.5.5.1 Programmable Physical Interface Behaviour

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the

board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the Port Status and Control (USB_PORTSC) register providing a capability that is not defined by EHCI.

56.4.5.5.2 Discovery

56.4.5.5.2.1 Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the Port Status and Control (USB_PORTSC) register for a duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to the reset the device.
- Software shall write a '0' to the reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

56.4.5.5.2.2 Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed - *This information is redundant with the 2-bit Port Speed indicator above.*

56.4.5.5.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI since the release of revision 3.2.1. In earlier product revisions, the test packet mode was not EHCI compatible. An alternate host controller driver procedure is no longer necessary or supported.

56.4.6 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller. The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

NOTE

Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writeable fields. The device controller must preserve the read-only fields on all data structure writes.

The ARC USB-HS OTG High-Speed USB On-The-Go core includes DCD Software called the USB 2.0 Device API. The Device API provides an easy to use Application Program Interface for developing device (peripheral) applications using the ARC USB-HS OTG High-Speed USB On-The-Go core. The Device API incorporates and abstracts for the application developer all of the elements of the program interface.

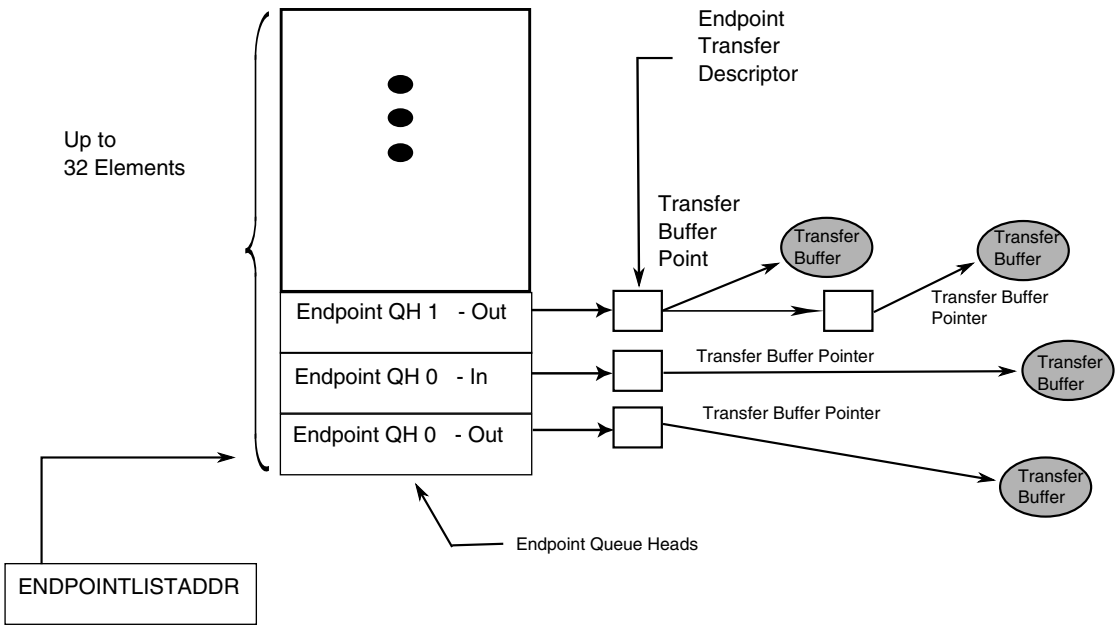


Figure 56-31. End Point Queue Head Organization

Device queue heads are arranged in an array in a continuous area of memory pointed to by the *ENDPOINTLISTADDR* pointer. The even -numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

The Endpoint Queue Head List must be aligned to a 2k boundary.

56.4.6.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries. During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the

dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

Table 56-56. Endpoint Queue Head (dQH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult		zlt	0		Maximum Packet Length											ios	0														
Current dTD Pointer																											0				
Next dTD Pointer																											0				T
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				Reserved											
Buffer Pointer (Page 2)																				Reserved											
Buffer Pointer (Page 3)																				Reserved											
Buffer Pointer (Page 4)																				Reserved											
Reserved																															
Set-up Buffer Bytes 3…0																															
Set-up Buffer Bytes 7…4																															

Shaded: Device Controller Read/Write

Non-Shaded: Device Controller Read Only.

56.4.6.1.1 dQH Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 56-57. Endpoint Capabilities/Characteristics

Bit	Description
31:30	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00 - Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 - Execute 1 Transaction. 10 - Execute 2 Transactions. 11 - Execute 3 Transactions. Note: Non-ISO endpoints must set Mult="00". Note: ISO endpoints must set Mult="01", "10", or "11" as needed.
29	Zero Length Termination Select. This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple . This bit is not relevant for Isochronous 0 - Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 - Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28:27	Reserved. These bit reserved for future use and should be set to zero.
26:16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14:0	Reserved. Bits reserved for future use and should be set to zero.

56.4.6.1.2 dQH Transfer Overlay

The seven DWords in the overlay area represent a transaction working space for the device controller. The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

56.4.6.1.3 dQH Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

Table 56-58. Next dTD Pointer

Bit	Description
31:5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4:0	Reserved. Bit reserved for future use and should be set to zero.

56.4.6.1.4 dQH Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

Table 56-59. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31:0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31:0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

56.4.6.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/ received for given transfer. The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing Transfers with Transfer Descriptors.

Table 56-60. Endpoint Transfer Descriptor (dTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Next Link Pointer																											0				T
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				0	Frame Number										

Table continues on the next page...

Table 56-60. Endpoint Transfer Descriptor (dTD) (continued)

Buffer Pointer (Page 2)	Reserved
Buffer Pointer (Page 3)	Reserved
Buffer Pointer (Page 4)	Reserved

Shaded: Device Controller Read/Write

Not-Shaded: Device Controller Read Only

Table 56-61. Next dTD Pointer

Bit	Description
31:5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4:1	Reserved. Bits reserved for future use and should be set to zero.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

Table 56-62. dTD Token

Bit	Description
31	Reserved. Bit reserved for future use and should be set to zero.
30:16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K(5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K(4000H). If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14:12	Reserved. Bits reserved for future use and should be set to zero.

Table continues on the next page...

Table 56-62. dTD Token (continued)

Bit	Description
11:10	Multiplier Override (MultO). This field can be used for transmit ISO's (ie. ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example: if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. **NOTE:** Non-ISO and Non-TX endpoints must set MultO="00".
9:8	Reserved. Bits reserved for future use and should be set to zero.
7:0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4,2,0 Reserved.

Table 56-63. dTD Buffer Page Pointer List

Bit	Description
31:12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0;11:0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1;10:0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

56.4.7 Device Operational Model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

The ARC USB-HS OTG High-Speed USB On-The-Go is shipped with a DCD called the ARC USB-HS OTG High-Speed USB On-The-Go Device API. The ARC USB-HS OTG High-Speed USB On-The-Go Device API provides an easy to use application interface for developing USB device (peripheral) applications. The ARC USB-HS OTG High-Speed USB On-The-Go Device API incorporates and abstracts for the application

developer all of the information contained in the device operational model. For more information on the ARC USB-HS OTG High-Speed USB On-The-Go Device API, refer to the "Software Design document for the Precise USB 2.0 Device API".

56.4.7.1 Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

1. Set Controller Mode in the USBMODE register to device mode.
2. Allocate and Initialize device queue heads in system memory
   - Minimum: Initialize device queue heads 0 Tx & 0 Rx.
   - For information on device queue heads, refer to section Device Data Structures.
3. Configure ENDPOINTLISTADDR Pointer.
   - For additional information on ENDPOINTLISTADDR, refer to the register table.
4. Enable the microprocessor interrupt associated with the ARC USB-HS OTG High-Speed USB On-The-Go core.
   - Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
   - For a list of available interrupts refer to the Interrupt Enable Register (USB_USBINTR) and the USB Status Register (USB_USBSTS) register tables.
5. Set Run/Stop bit to Run Mode.
   - After the Run bit is set, a device reset will occur. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the following Port State and Control section below.

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.

NOTE

All device queue heads must be initialized for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

NOTE

Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

56.4.7.2 Port State and Control

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a '1'. After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0. The following state diagram depicts the state of a USB 2.0 device.

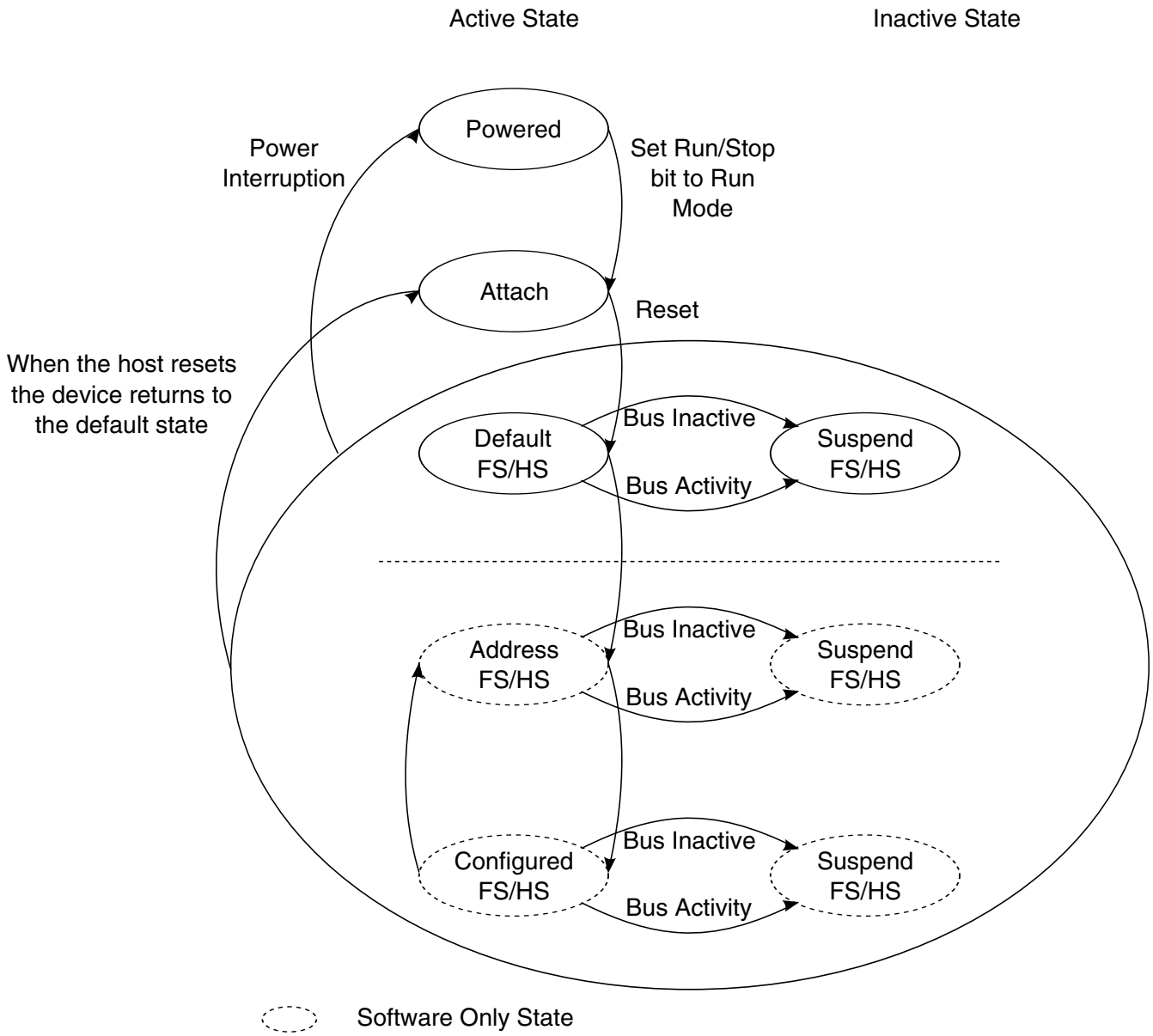


Figure 56-32. Device State Diagram

States *powered, attach, defaultFS/HS, suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

Table 56-64. Device Controller State Information Bits

Bit	Register
DCSuspend	USB Status Register (USB_USBSTS)
USB Reset Received	USB Status Register (USB_USBSTS)
Port Change Detect	USB Status Register (USB_USBSTS)
High-Speed Port	Port Status and Control (USB_PORTSC)

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the ENDPTCTRLx registers and initializing the associated queue heads.

56.4.7.2.1 Bus Reset

A bus reset is used by the host to initialize downstream devices. When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the Endpoint Setup Status (USB_ENDPTSETUPSTAT) register and writing the same value back to the Endpoint Setup Status (USB_ENDPTSETUPSTAT) register.

Clear all the endpoint complete status bits by reading the Endpoint Complete (USB_ENDPTCOMPLETE) register and writing the same value back to the Endpoint Complete (USB_ENDPTCOMPLETE) register.

Cancel all primed status by waiting until all bits in the Endpoint Initialization (USB_ENDPTPRIME) are 0 and then writing 0xFFFFFFFF to Endpoint De-Initialize (USB_ENDPTFLUSH).

Read the reset bit in Port Status and Control (USB_PORTSC) and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the Port Status and Control (USB_PORTSC) to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9 - Device Framework.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it may not be possible to enable one or more pipes while in FS mode. *Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.*

56.4.7.2.2 Port State and Control Suspend/Resume

Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming *DC Suspend Interrupt* is enabled). When the *DCSuspend* bit in the Port Status and Control (USB_PORTSC) is set to a '1', the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the Port Status and Control (USB_PORTSC) while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

Port Test Modes

Contact ARC International for port test mode capabilities.

56.4.7.2.3 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device. The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and *isochronous*. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The ARC USB-HS OTG High-Speed USB On-The-Go device controller hardware supports up to the USB 2.0 maximum of 32 endpoint specified numbers. Each additional endpoint beyond the required endpoint position adds additional hardware logic. The maximum number of endpoint numbers available to the DCD is configured at hardware synthesis timer. After synthesis, the DCD can enable, disable and configure endpoint type up to the maximum selected during synthesis.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. If the maximum of 16 endpoint numbers, one for each endpoint direction are being used by the device controller, then 32 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

56.4.7.2.4 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the ENDPTCTRLx register. Each 32-bit ENDPTCTRLx is split into an upper and lower half. The lower half of ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization

Table 56-65. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	'1'
Data Toggle Inhibit	'0'
Endpoint Type	"00" - Control "01" - Isochronous "10" - Bulk "11" - Interrupt
Endpoint Stall	'0'

A protocol stall, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the ENDPTCTRLx register during operational mode must preserve the endpoint type field (i.e. perform a read-modify-write).

Table 56-66. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit	Effect on STALL Bit	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'1'	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'0'	None.	ACK/ NAK/ NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	'1'	None	STALL
IN/OUT/PING packet received by a control endpoint.	'0'	None.	ACK/ NAK/ NYET

56.4.7.2.5 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe. For more information on data toggle, refer to the USB 2.0 specification.

Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the ENDPTCTRLx register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

Data Toggle Inhibit

Note: This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

56.4.7.3 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification. At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as **"priming"** the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term **"flushing"** is used to describe the action of clearing a packet that was queued for execution.

Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH). After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Since only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section .

Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

56.4.7.3.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical. All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the

following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 56-67. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	-
512	256	3	256	256	0
512	512	2	512	0	-

Table 56-68. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	-
512	256	2	256	256	-
512	512	1	512	-	-

The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD.

In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

Interrupt/Bulk Endpoint Bus Response Matrix

Table 56-69. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

NOTE

BS Error = Force Bit Stuff Error

NOTE

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

NOTE

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

56.4.7.3.2 Control Endpoint Operation Model

Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

In hardware versions 2.3 and later, the setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

Setup Packet Handling (Pre-2.3 hardware)

After receiving an interrupt and inspecting USB Device Mode (USB_USBMODE) to determine that a setup packet was received on a particular pipe:

1. Duplicate contents of dQH.SsetupBuffer into local software byte array.
2. Write '1' to clear corresponding Endpoint Setup Status (USB_ENDPTSETUPSTAT) bit and thereby disabling Setup Lockout. (i.e. the Setup Lockout activates as soon as a setup arrives. By writing to the Endpoint Setup Status (USB_ENDPTSETUPSTAT), the device controller will accept new setup packets.)
3. Process setup packet using local software byte array copy and execute status/ handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

To limit the exposure of setup packets to the setup lockout mechanism (if used), the DCD should designate the priority of responding to setup packets above responding to other packet completions.

Setup Packet Handling (2.3 hardware and later)

Disable Setup Lockout by writing '1' to Setup Lockout Mode (SLOM) in USB Device Mode (USB_USBMODE). (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

After receiving an interrupt and inspecting Endpoint Setup Status (USB_ENDPTSETUPSTAT) to determine that a setup packet was received on a particular pipe:

1. Write '1' to clear corresponding bit Endpoint Setup Status (USB_ENDPTSETUPSTAT).
2. Write '1' to Setup Tripwire (SUTW) in USB Command Register (USB_USBCMD) register.
3. Duplicate contents of dQH.SetupBuffer into local software byte array.
4. Read Setup TripWire (SUTW) in USB Command Register (USB_USBCMD) register. (if set - continue; if cleared - goto 2)
5. Write '0' to clear Setup Tripwire (SUTW) in USB Command Register (USB_USBCMD) register.
6. Process setup packet using local software byte array copy and execute status/ handshake phases.

Leaving the Setup Lockout Mode As '0' will result in pre-2.3 hardware behavior.

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the Endpoint Initialization (USB_ENDPTPRIME) register is zero and the associated bit in the Endpoint Status (USB_ENDPTSTAT) register is a one. If a prime fails, ie. The Endpoint Initialization (USB_ENDPTPRIME) bit goes to zero and the Endpoint Status (USB_ENDPTSTAT) bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (Endpoint Status (USB_ENDPTSTAT)) to enforce data coherency with the setup packet.

NOTE

The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

NOTE

Error handling of data phase packets is the same as bulk packets described previously.

Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase. The DCD must also perform the same checks of the ENDPTSETUPSTAT as described above in the data phase.

NOTE

The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

NOTE

Error handling of data phase packets is the same as bulk packets described previously.

Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

Table 56-70. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	
In	STALL	NAK	Transmit	BS Error	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

56.4.7.3.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes. Real time delivery by the device controller will is accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.

- NAK responses are not used. Instead, zero length packets and sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
- MULT counter reaches zero.
- Fulfillment Error [*Transaction Error* bit is set]
- # Packets Occurred > 0 AND # Packets Occurred < MULT
- RX Packet Retired:
- MULT counter reaches zero.

- Non-MDATA Data PID is received**
- ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
- Overflow Error:
- Packet received is > maximum packet length. [*Buffer Error* bit is set]
- Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]
- Fulfillment Error [*Transaction Error* bit is set]
- # Packets Occurred > 0 AND # Packets Occurred < MULT
- CRC Error [*Transaction Error* bit is set

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at Isochronous Pipe Synchronization

When it is necessary to synchronize an isochroous data pipe to the host, the (micro)frame number (FRINDEX register) can be used as a marker. To cause a packet transfer to occur at a specific (micro)frame number [N], the DCD should interrupt on SOF during frame N-1. When the FRINDEX=N-1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro)frame N-1 so that the device controller will execute delivery during (micro)frame N.

NOTE

Priming an endpoint towards the end of (micro)frame N-1 will not guarantee delivery in (micro)frame N. The delivery may actually occur in (micro)frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

Isochronous Endpoint Bus Response Matrix

Table 56-71. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A

Table continues on the next page...

Table 56-71. Isochronous Endpoint Bus Response Matrix (continued)

In	NULL Packet[1, 1]	NULL Packet	Transmit	BS Error[2, 2]	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. NULL Packet = Zero Length Packet
2. BS Error = Force Bit Stuff Error

56.4.7.4 Managing Queue Heads

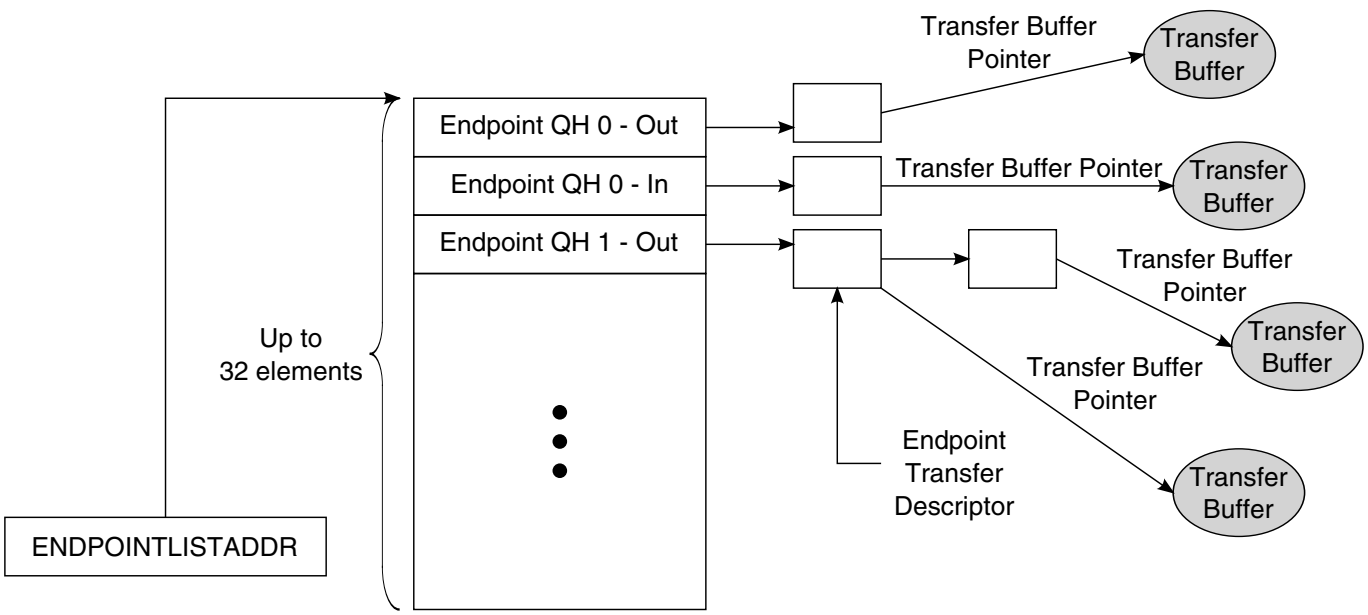


Figure 56-33. End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 56-33. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multipler, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

56.4.7.4.1 Queue Head Initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth an in conjuction with the USB Chapter 9 protocol. *Note: In FS mode, the multiplier field can only be 1 for ISO endpoints.*
- Write the next dTD Terminate bit field to "1".
- Write the Active bit in the status field to "0".
- Write the Halt bit in the status field to "0".

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

56.4.7.4.2 Operational Model For Setup Transfers

As discussed in Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

NOTE

The acknowledge must occur before continuing to process the setup packet.

NOTE

After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH - RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in Flushing/De-priming an Endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

56.4.7.5 Managing Transfers with Transfer Descriptors

56.4.7.5.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

.

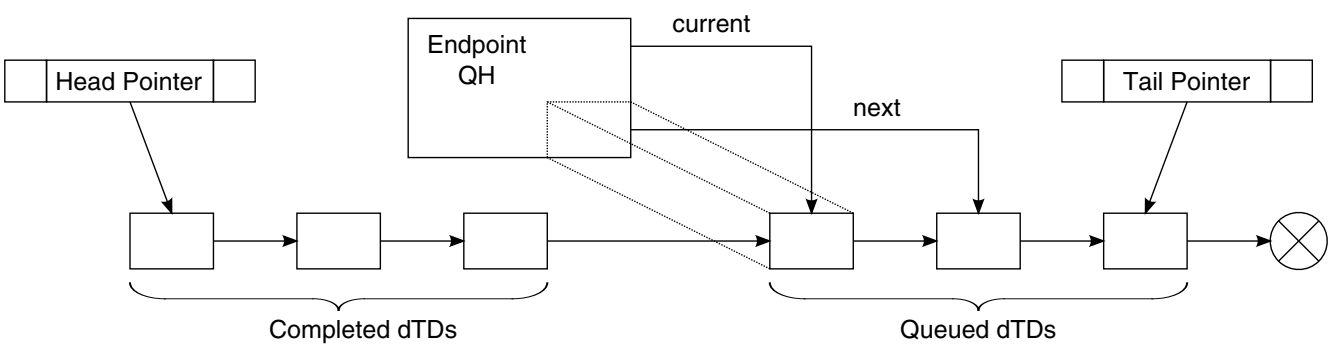


Figure 56-34. Software Link Pointers

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.

56.4.7.5.2 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer. Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to "00000"

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to "1".
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to "1" and all remaining status bits set to "0".
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

56.4.7.5.3 Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty:

Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding)

Case 1: Link list is empty

1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
2. Clear active & halt bit in dQH (in case set from a previous error).
3. Prime endpoint by writing '1' to correct bit position in Endpoint Initialization (USB_ENDPTPRIME).

Case 2: Link list is not empty

1. Add dTD to end of linked list.
2. Read correct prime bit in Endpoint Initialization (USB_ENDPTPRIME)- if '1' DONE.
3. Set ATDTW bit in USBCMD register to '1'.

4. Read correct tatus bit in Endpoint Status (USB_ENDPTSTAT). (store in tmp. variable for later)
5. Read ATDTW bit in USBCMD register.
   - If '0' go to 3.
   - If '1' continue to 6.
6. Write ATDTW bit in USBCMD register to '0'.
7. If status bit read in (3) is '1' DONE.
8. If status bit read in (3) is '0' then Goto Case 1: Step 1.

56.4.7.5.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

NOTE

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

56.4.7.5.5 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer. There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in Endpoint De-Initialize (USB_ENDPTFLUSH).
2. Wait until all bits in Endpoint De-Initialize (USB_ENDPTFLUSH) are '0'.
3. Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
4. Read Endpoint Status (USB_ENDPTSTAT) to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:
5. Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using Endpoint De-Initialize (USB_ENDPTFLUSH). A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

56.4.7.5.6 Device Error Matrix

Table 56-72 summarizes packet errors that are not automatically handled by the Device Controller.

Table 56-72. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated in Table 56-73

Table 56-73. Error Descriptions

Overflow	Number of bytes received exceeded max. packet size or total buffer length.

Table continues on the next page...

Table 56-73. Error Descriptions (continued)

	** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Hst failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the "dead" (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

56.4.7.6 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

56.4.7.6.1 High-Frequency Interrupts

High frequency interrupts in particular should be handed in the order shown in Table 56-74. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 56-74. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrup[1, 1] - ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet . Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt[1] - ENDPTCOMPLETE	Handle completion of dTD as indicated in section .
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

1. It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

56.4.7.6.2 Low-Frequency Interrupts

The low frequency events include the following interrupts. These interrupt can be handled in any order since they don't occur often in comparison to the high-frequency interrupts.

Table 56-75. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.

Table continues on the next page...

Table 56-75. Low Frequency Interrupt Events (continued)

Interrupt	Action
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

56.4.7.6.3 Error Interrupts

- Error interrupts will be least frequent and should be placed last in the interrupt service routine.

Table 56-76. Error Interrupt Events

Interrupt	Action
USB Error Interrupt.	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

56.5 Programmable Registers

This section describes only the registers that are additional to the HS-USB Core register set. For a detailed description of the registers inside the HS-USB controllers, please refer to Register Interface

USBOH1 memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FC_4800	USB Control Register (USBOH1_USB_CTRL)	32	R/W	4040_4002h	56.5.1/ 3271
53FC_4804	UTMI PHY Output Clock Valid Register (USBOH1_UTMI_CLK_VLD)	8	R	00h	56.5.2/ 3273
53FC_4808	OTG UTMI PHY Control 0 Register (USBOH1_OTG_PHY_CTRL_0)	32	R/W	8000_1400h	56.5.3/ 3274
53FC_480C	OTG UTMI PHY Control 1 Register (USBOH1_OTG_PHY_CTRL_1)	32	R/W	0054_1401h	56.5.4/ 3276
53FC_4810	USB Control Register 1 (USBOH1_USB_CTRL_1)	32	R/W	0000_0000h	56.5.5/ 3279

Table continues on the next page...

USBOH1 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53FC_4814	USB Control Register 2 (USBOH1_USB_CTRL_2)	32	R/W	4040_1402h	56.5.6/ 3280
53FC_481C	Host1 UTMI PHY Control 0 Register (USBOH1_UH1_PHY_CTRL_0)	32	R/W	8000_1400h	56.5.7/ 3281
53FC_4820	Host1 UTMI PHY Control 1 Register (USBOH1_UH1_PHY_CTRL_1)	32	R/W	0054_1401h	56.5.8/ 3283
53FC_4824	USB Clock on/off control Register (USBOH1_USB_CLKONOFF_CTRL)	32	R/W	0980_0000h	56.5.9/ 3285

56.5.1 USB Control Register (USBOH1_USB_CTRL)

The USB control register controls the power control and wake-up functionality of the USB module.

Address: USBOH1_USB_CTRL is 53FC_4000h base + 800h offset = 53FC_4800h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OWIR	-			OWIE	-		O_PWR_POL	-							
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H1WIR	-			H1WIE	-		H1_PWR_POL	-							
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0

USBOH1_USB_CTRL field descriptions

Field	Description
31 OWIR	OTG Wake up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt. 1 Wake-up Interrupt Request received 0 No Wake-up detected
30–28 -	Reserved

Table continues on the next page...

USBOH1_USB_CTRL field descriptions (continued)

Field	Description
27 OWIE	OTG Wake-up Interrupt Enable This bit enables or disables the OTG wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
26–25 -	Reserved
24 O_PWR_POL	OTG power Pin polarity The polarity of OTG Power pin(to enable external PMIC to drive VBUS) 1 High active 0 Low active
23–16 -	Reserved
15 H1WIR	Host 1 Wake-up Interrupt Request Indicates a pending Wake-up request on Host port 1. This bit is cleared by disabling the interrupt. The interrupt must be disabled for at least 2 clock cycles of the standby clock. 1 Wake-up interrupt received 0 No Wake-up interrupt received
14–12 -	Reserved
11 H1WIE	Host 1 Wake-up Interrupt Enable This bit enables or disables the Host 1 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
10–9 -	Reserved
8 H1_PWR_POL	Host1 power Pin polarity The polarity of Host1 Power pin(to enable external PMIC to drive VBUS) 1 High active 0 Low active
7–0 -	Reserved

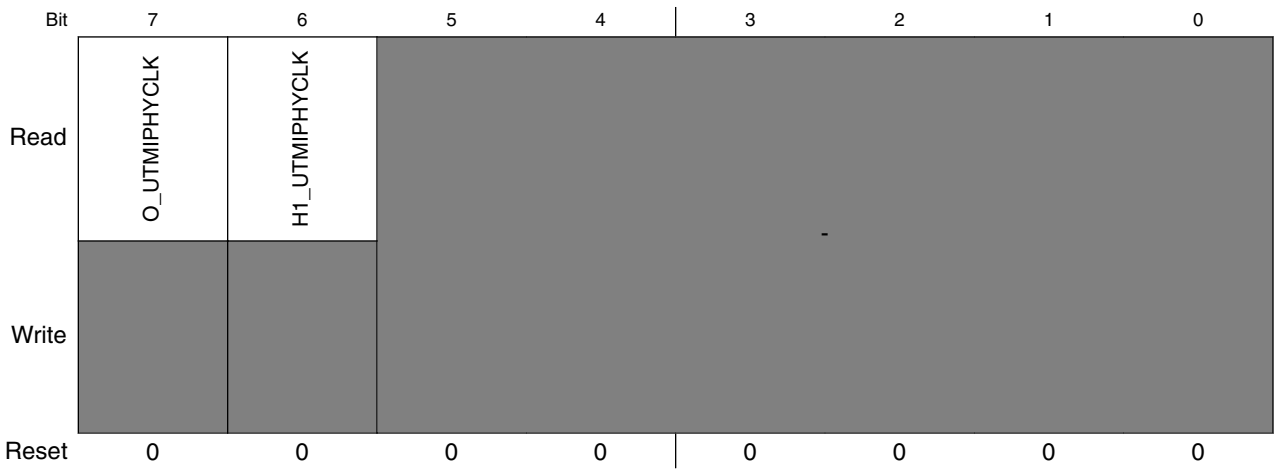
56.5.2 UTMI PHY Output Clock Valid Register (USBOH1_UTMI_CLK_VLD)

The UTMI PHY output clock is not valid after power on reset, it will valid several hundred micro seconds after PHY reference clock is valid. This register provide a flag that SW can detect when the PHY output clock is valid.

NOTE

The clock valid bit only can be cleared by a hardware reset.

Address: USBOH1_UTMI_CLK_VLD is 53FC_4000h base + 804h offset = 53FC_4804h



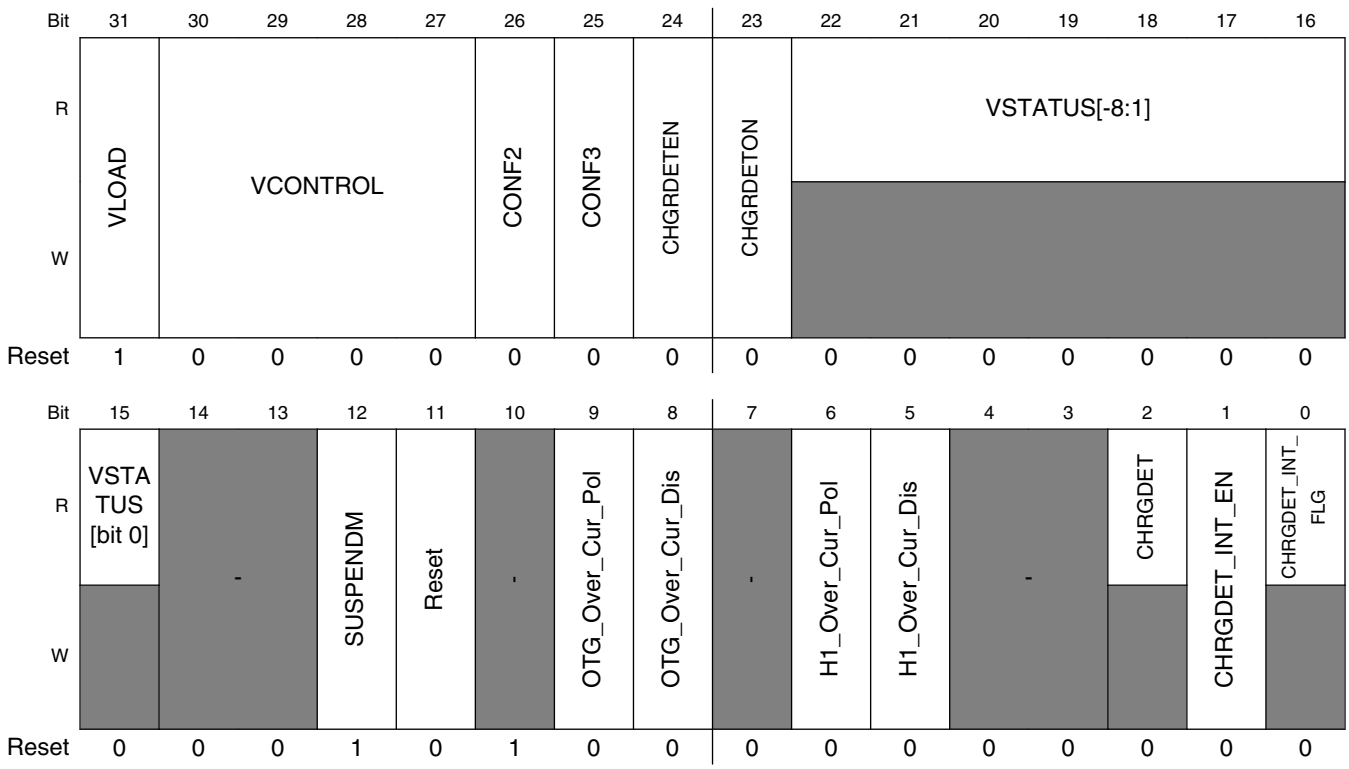
USBOH1_UTMI_CLK_VLD field descriptions

Field	Description
7 O_ UTMIPHYCLK	OTG UTMI PHY Clock on detection This is a readonly status bit, indicating the external OTG UTMI PHY clock is on and sent to USB controller. 1 The OTG utmi input clock is on 0 The OTG utmi input clock is off
6 H1_ UTMIPHYCLK	Host1 UTMI PHY Clock on detection This is a readonly status bit, indicating the external Host1 UTMI PHY clock is on and sent to USB controller. 1 The Host1 utmi input clock is on 0 The Host1 utmi input clock is off
5–0 -	Reserved

56.5.3 OTG UTMI PHY Control 0 Register (USBOH1_OTG_PHY_CTRL_0)

PHY Control Register 0/1 are used to control the internal UTMI PHY.

Address: USBOH1_OTG_PHY_CTRL_0 is 53FC_4000h base + 808h offset = 53FC_4808h



USBOH1_OTG_PHY_CTRL_0 field descriptions

Field	Description
31 VLOAD	UTMI PHY Vload Assertion of this signal loads the Vendor Control register. Active low. 1 Inactive 0 loads the Vendor Control register
30–27 VCONTROL	UTMI PHY Vcontrol
26 CONF2	UTMI CONF2 Active High. When asserted will turn on OTG comparators during suspend(suspend = '0'). 1 will turn on OTG comparators during suspend 0 Inactive
25 CONF3	UTMI PHY CONF3 During non-driving mode (opmode[1:0] = 2'b01) if conf3 = 1 the 15Kohm pull-down resistors will be connected (if dppulldown = dnpulldown = '1'). Should be tied to '0' for device only applications.

Table continues on the next page...

USBOH1_OTG_PHY_CTRL_0 field descriptions (continued)

Field	Description
24 CHGRDETEN	UTMI PHY chgrdeten Active High. Enable Charger Detector. This pin must be asserted 300us (or more) after chgrdeton going to '1'.
23 CHGRDETON	UTMI PHY chgrdeton Active High. Charger Detector Power On Control. This pin controls the internal current mirrors used for charger detection.
22–15 VSTATUS	UTMI PHY Vstatus Vendor Status - vender defined 8-bit parallel output.
14–13 -	Reserved
12 SUSPENDM	UTMI PHY Suspend Active Low. Force PHY into low power suspend mode (only used for test). In normal operation, S/W should set PORTSC.SUSP and PORTSC.PHCD bits to put the PHY into low power suspend mode. 1 Disable 0 Enable
11 Reset	UTMI PHY Reset Active High. Force to Reset the UTMI PHY (only used for test). In normal operation, S/W should set USBCMD.RST bit to reset the UTMI PHY. 1 Reset the PHY 0 Inactive
10 -	Reserved
9 OTG_Over_Cur_Pol	OTG Polarity of Overcurrent The polarity of OTG port overcurrent event. 1 Low active 0 High active
8 OTG_Over_Cur_Dis	OTG Disable Overcurrent Event Disable the OTG overcurrent event. 1 Disable the Overcurrent event 0 Enable the overcurrent event
7 -	Reserved
6 H1_Over_Cur_Pol	Host1 Polarity of Overcurrent The polarity of Host1 port overcurrent event. 1 Low active 0 High active
5 H1_Over_Cur_Dis	Host1 Disable Overcurrent Event Disable the Host1 overcurrent event.

Table continues on the next page...

USBOH1_OTG_PHY_CTRL_0 field descriptions (continued)

Field	Description
	1 Disable the Overcurrent event 0 Enable the overcurrent event
4–3 -	Reserved
2 CHRGDET	UTMI PHY chrgdet Charger detector output. **NOTE:** Maximum response time = 1us 1 When a charger is detected 0 When a Host is detected
1 CHRGDET_INT_EN	Chrgdet detected interrupt enable Charger detected interrupt enable. 1 Enable the charger detected interrupt 0 Disable the charger detected interrupt
0 CHRGDET_INT_FLG	Chrgdet detected interrupt flag Charger detected interrupt flag. This bit will be cleared when CHRGDET_INT_EN is '0' 1 charger detected interrupt occurred 0 no charger detected interrupt

56.5.4 OTG UTMI PHY Control 1 Register (USBOH1_OTG_PHY_CTRL_1)

Address: USBOH1_OTG_PHY_CTRL_1 is 53FC_4000h base + 80Ch offset = 53FC_480Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	hsdrvtimingp	hsdrvtimingn		hsdrvamplitude		hsdrvslope				hsdedvsel		hstedvsel		fstunevsel		
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	icpctrl		fsrftsel		lsrftsel		enpre	preemdepth	calbp	extcal					plldivvalue	
Reset	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1

USBOH1_OTG_PHY_CTRL_1 field descriptions

Field	Description
31 hsdrvtimingp	HS driver timing control for PMOS 1 8x 0 2x
30–29 hsdrvtimingn	HS driver timing control for NMOS 00 2x 01 4x 10 6x 11 8x
28–27 hsdrvamplitude	HS driver amplitude control 00 I (I = 17.78mA) 01 I + 2.5% 10 I + 5% 11 I + 7.5%
26–23 hsdrvslope	HS driver slope control The HS Driver may have its rise/fall times controlled using the hsdrvslope[3:0] control pins. Additional charge is injected, so the HS driver Rise/Fall times change (RC constant changes). Rise/Fall times: Depends on the Package, PCB... Correct value result from Silicon tests
22–21 hsdedvsel	Reference voltage for high speed disconnect envelope detector 00 (46+2)/100*vbg (556.8mV) 01 (46+3)/100*vbg (568.4mV) 10 (46+4)/100*vbg (580mV) 11 (46+5)/100*vbg (591.6mV)
20–19 hstedvsel	Reference voltage for high speed transmission envelope detector 00 vbg*(9)/100 (104.4mV) 01 vbg*(9+1)/100 (116mV) 10 vbg*(9+2)/100 (127.6mV) 11 vbg*(9+3)/100 (139.2mV)
18–16 fstunevsel	Reference voltage control for Calibration circuit 000 (46)/100*vbg (533.6mV) 001 (46+1)/100*vbg (545.2mV) 010 (46+2)/100*vbg (556.8mV) 011 (46+3)/100*vbg (568.4mV) 100 (46+4)/100*vbg (580mV) 101 (46+5)/100*vbg (591.6mV) 110 (46+6)/100*vbg (603.2mV) 111 (46+7)/100*vbg (614.8mV)
15–14 icpctrl	PLL charge pump current control 00 Icp (Icp = 40uA) 01 Icp * 0.5

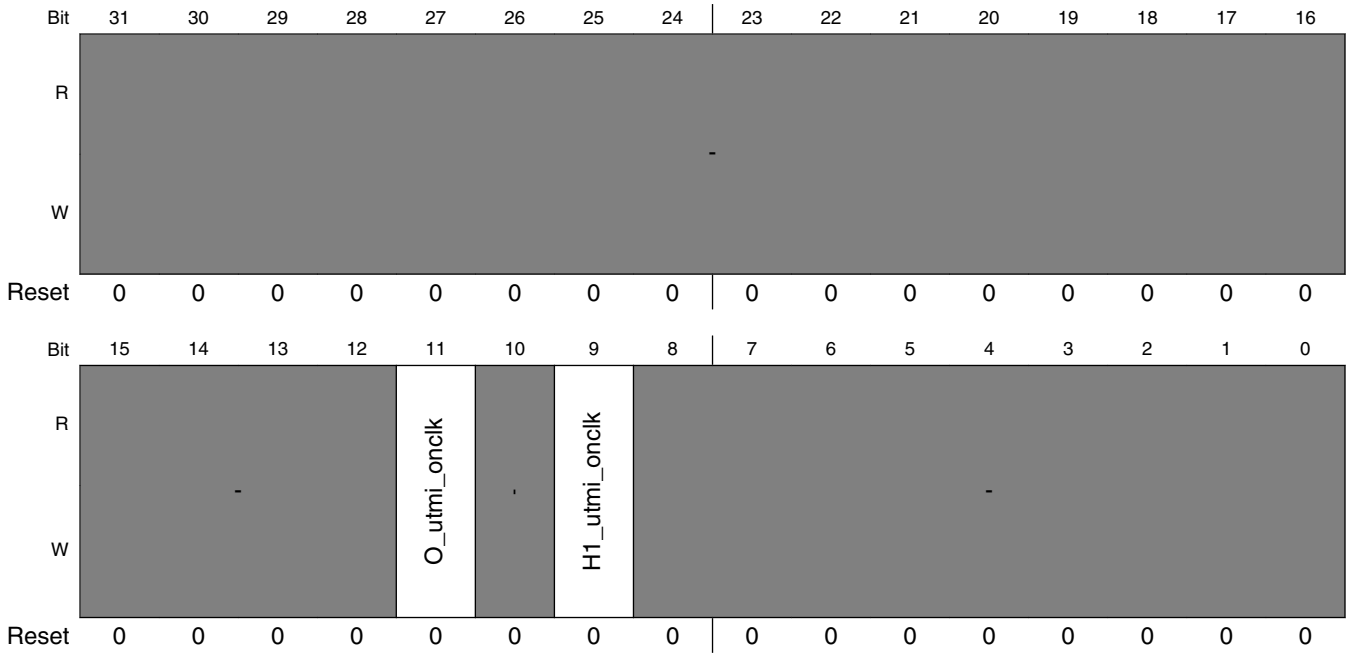
Table continues on the next page...

USBOH1_OTG_PHY_CTRL_1 field descriptions (continued)

Field	Description
	10 Icp * 1.5 11 Icp * 2
13–12 fsrftsel	FS driver rise/fall time control 00 Nominal FS rise time-30% 01 Nominal FS rise time 10 Nominal FS rise time 11 Nominal FS rise time+30%
11–10 lsrftsel	LS driver rise/fall time control 00 Nominal LS rise time-30% 01 Nominal LS rise time 10 Nominal LS rise time 11 Nominal LS rise time+30%
9 enpre	HS driver pre-emphasis enable
8 preemdepth	HS driver pre-emphasis depth enpre, preemdepth 00 I (I = 17.78mA) 01 I + 5% 10 I + 10% 11 I + 20%
7 calbp	Enables calibration bypass for both dp and dn lines
6–2 extcal	Controls calibration value externally. Valid when calbp = '1'
1–0 plldivvalue	Selects between 19.2Mhz, 24Mhz, 26Mhz or 27Mhz reference clock 00 sysclock uses 19.2Mhz 01 sysclock uses 24Mhz 10 sysclock uses 26Mhz 11 sysclock uses 27Mhz

56.5.5 USB Control Register 1 (USBOH1_USB_CTRL_1)

Address: USBOH1_USB_CTRL_1 is 53FC_4000h base + 810h offset = 53FC_4810h



USBOH1_USB_CTRL_1 field descriptions

Field	Description
31–12 -	Reserved
11 O_utmi_onclk	OTG UTMI PHY On clock Allows the system clocks clockout to be available even if suspend is asserted. 1 Enable 0 Disable
10 -	Reserved
9 H1_utmi_onclk	UTMI PHY On clock Allows the system clocks clockout to be available even if suspend is asserted. 1 Enable 0 Disable
8–0 -	Reserved

56.5.6 USB Control Register 2 (USBOH1_USB_CTRL_2)

Address: USBOH1_USB_CTRL_2 is 53FC_4000h base + 814h offset = 53FC_4814h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-															OPMODE_OVERRIDE_EN
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OPMODE_OVERRIDE		-							OVBWK_EN	OIDWK_EN	-				
W																
Reset	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0

USBOH1_USB_CTRL_2 field descriptions

Field	Description
31–17 -	Reserved
16 OPMODE_OVERRIDE_EN	OPMODE Override enable Selects between Controller opmode setting and S/W selected mode 1 Override the OPMODE[1:0] with OPMODE_OVERRIDE bits for the OTG & Host1 on-chip UTMI PHY 0 Disable the OPMODE override function.
15–14 OPMODE_OVERRIDE	OPMODE Override value. OPMODE override value for OTG & Host1 on-chip UTMI PHY. It's only active when OPMODE_OVERRIDE_EN bit is set 01 Non-driving 10 Disable bit stuffing and NRZI encoding 11 Normal operation without automatic generation of SYNC and EOP. 00 Normal operation
13–7 -	Reserved
6 OVBWK_EN	OTG VBUS Wakeup Enable This bit enables or disables the interrupt which is caused OTG VBUS Change. 1 Interrupt Enabled 0 Interrupt Disabled
5 OIDWK_EN	OTG ID Wakeup Enable This bit enables or disables the interrupt which is caused OTG ID Change.

Table continues on the next page...

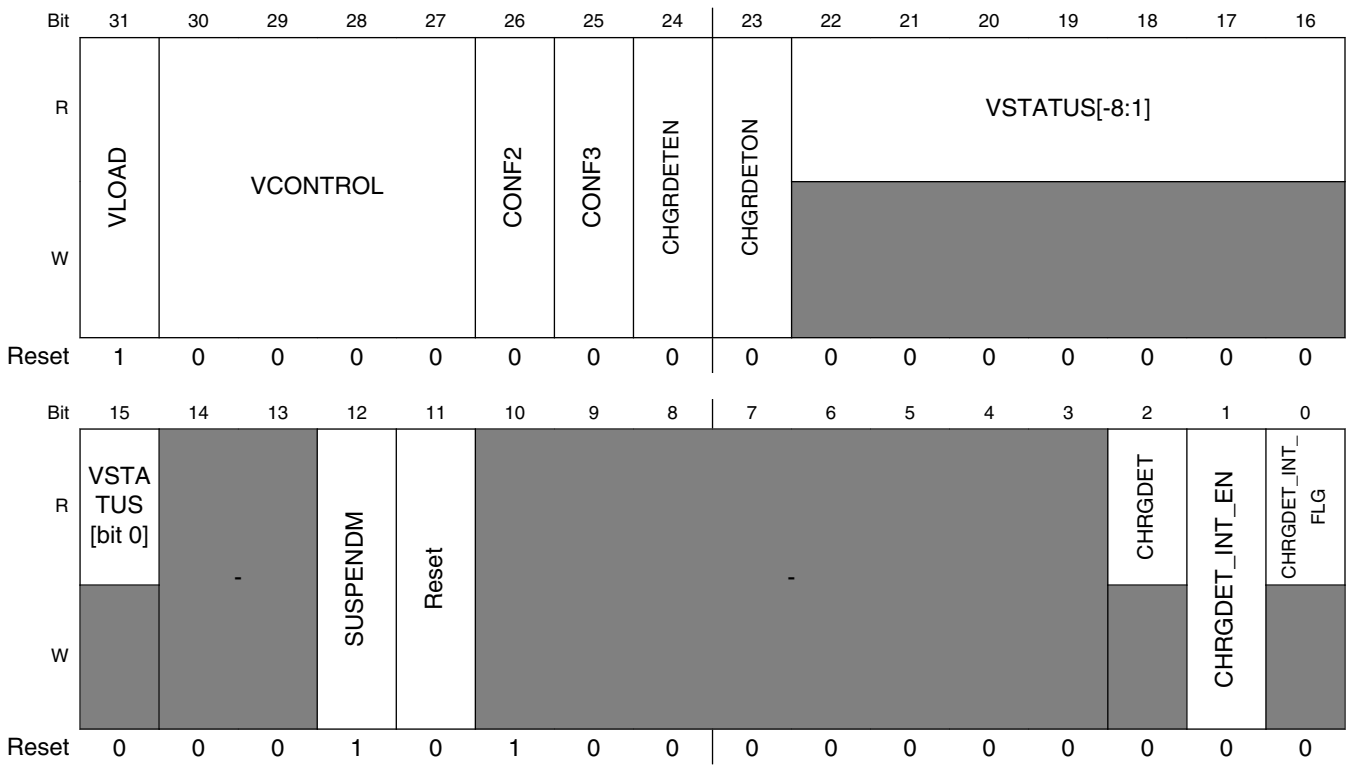
USBOH1_USB_CTRL_2 field descriptions (continued)

Field	Description
	1 Interrupt Enabled 0 Interrupt Disabled
4–0 -	Reserved

56.5.7 Host1 UTMI PHY Control 0 Register (USBOH1_UH1_PHY_CTRL_0)

PHY Control Register 0/1 are used to control the internal UTMI PHY.

Address: USBOH1_UH1_PHY_CTRL_0 is 53FC_4000h base + 81Ch offset = 53FC_481Ch



USBOH1_UH1_PHY_CTRL_0 field descriptions

Field	Description
31 VLOAD	UTMI PHY Vload Assertion of this signal loads the Vendor Control register. Active low. 1 Inactive 0 loads the Vendor Control register
30–27 VCONTROL	UTMI PHY Vcontrol

Table continues on the next page...

USBOH1_UH1_PHY_CTRL_0 field descriptions (continued)

Field	Description
26 CONF2	UTMI CONF2 Active High. When asserted will turn on OTG comparators during suspend(suspend = '0'). 1 will turn on OTG comparators during suspend 0 Inactive
25 CONF3	UTMI PHY CONF3 During non-driving mode (opmode[1:0] = 2'b01) if conf3 = 1 the 15Kohm pull-down resistors will be connected (if dppulldown = dnpulldown = '1'). Sshould be tied to '0' for device only applications.
24 CHGRDETEN	UTMI PHY chgrdeten Active High. Enable Charger Detector. This pin must be asserted 300us (or more) after chgrdeton going to '1'.
23 CHGRDETON	UTMI PHY chgrdeton Active High. Charger Detector Power On Control. This pin controls the internal current mirrors used for charger detection.
22–15 VSTATUS	UTMI PHY Vstatus Vendor Status-vender defined 8-bit parallel output.
14–13 -	Reserved
12 SUSPENDM	Host1 UTMI PHY Suspend Active Low. Force Host1 PHY into low power suspend mode (only used for test). In normal operation, S/W should set PORTSC.SUSP and PORTSC.PHCD bits to put the PHY into low power suspend mode. 1 Disable 0 Enable
11 Reset	Host 1 UTMI PHY Reset Active High. Force to Reset the Host1 UTMI PHY (only used for test). In normal operation, S/W should set USBCMD.RST bit to reset the UTMI PHY. 1 Reset the PHY 0 Inactive
10–3 -	Reserved
2 CHRGDET	UTMI PHY chrgdet Charger detector output. **NOTE:** Maximum response time = 1us 1 When a charger is detected 0 When a Host is detected
1 CHRGDET_INT_EN	chrgdet detected interrupt enable Charger detected interrupt enable. 1 Enable the charger detected interrupt 0 Disable the charger detected interrupt

Table continues on the next page...

USBOH1_UH1_PHY_CTRL_0 field descriptions (continued)

Field	Description
0 CHRGDET_INT_FLG	chrgdet detected interrupt flag Charger detected interrupt flag. This bit will be cleared when CHRGDET_INT_EN is '0' 1 charger detected interrupt occurred 0 no charger detected interrupt

56.5.8 Host1 UTMI PHY Control 1 Register (USBOH1_UH1_PHY_CTRL_1)

Address: USBOH1_UH1_PHY_CTRL_1 is 53FC_4000h base + 820h offset = 53FC_4820h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	hsdrvtimingp	hsdrvtimingn		hsdrvamplitude		hsdrvslope				hsdedvsel		hstedvsel		fstunevsel		
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	icpctrl		fsrftsel		lsrftsel		enpre	preemdepth	calbp	extcal					plldivvalue	
Reset	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1

USBOH1_UH1_PHY_CTRL_1 field descriptions

Field	Description
31 hsdrvtimingp	HS driver timing control for PMOS 1 8x 0 2x
30–29 hsdrvtimingn	HS driver timing control for NMOS 00 2x 01 4x 10 6x 11 8x
28–27 hsdrvamplitude	HS driver amplitude control 00 I (I = 17.78 mA) 01 I + 2.5%

Table continues on the next page...

USBOH1_UH1_PHY_CTRL_1 field descriptions (continued)

Field	Description
	10 I + 5% 11 I + 7.5%
26–23 hsdrvslope	HS driver slope control The HS Driver may have its rise/fall times controlled using the hsdrvslope[3:0] control pins. Additional charge is injected, so the HS driver Rise/Fall times change (RC constant changes). Rise/Fall times: Depends on the Package, PCB... Correct value result from Silicon tests
22–21 hsdedvsel	Reference voltage for high speed disconnect envelope detector 00 (46+2)/100*vbg (556.8 mV) 01 (46+3)/100*vbg (568.4 mV) 10 (46+4)/100*vbg (580 mV) 11 (46+5)/100*vbg (591.6 mV)
20–19 hstedvsel	Reference voltage for high speed transmission envelope detector 00 vbg*(9)/100 (104.4 mV) 01 vbg*(9+1)/100 (116 mV) 10 vbg*(9+2)/100 (127.6 mV) 11 vbg*(9+3)/100 (139.2 mV)
18–16 fstunevsel	Reference voltage control for Calibration circuit 000 (46)/100*vbg (533.6 mV) 001 (46+1)/100*vbg (545.2 mV) 010 (46+2)/100*vbg (556.8 mV) 011 (46+3)/100*vbg (568.4 mV) 100 (46+4)/100*vbg (580 mV) 101 (46+5)/100*vbg (591.6 mV) 110 (46+6)/100*vbg (603.2 mV) 111 (46+7)/100*vbg (614.8 mV)
15–14 icpctrl	PLL charge pump current control 00 Icp (Icp = 40 uA) 01 Icp * 0.5 10 Icp * 1.5 11 Icp * 2
13–12 fsrftsel	FS driver rise/fall time control 00 Nominal FS rise time - 30% 01 Nominal FS rise time 10 Nominal FS rise time 11 Nominal FS rise time + 30%
11–10 lsrftsel	LS driver rise/fall time control 00 Nominal LS rise time - 30% 01 Nominal LS rise time 10 Nominal LS rise time 11 Nominal LS rise time + 30%

Table continues on the next page...

USBOH1_UH1_PHY_CTRL_1 field descriptions (continued)

Field	Description
9 enpre	HS driver pre-emphasis enable
8 preemdepth	HS driver pre-emphasis depth enpre, preemdepth 00 I (I = 17.78 mA) 01 I + 5% 10 I + 10% 11 I + 20%
7 calbp	Enables calibration bypass for both dp and dn lines
6–2 extcal	Controls calibration value externally. Valid when calbp = '1'
1–0 plldivvalue	Selects between 19.2 MHz, 24 MHz, 26 MHz or 27 Mhz reference clock 00 sysclock uses 19.2 MHz 01 sysclock uses 24 MHz 10 sysclock uses 26 MHz 11 sysclock uses 27 MHz

56.5.9 USB Clock on/off control Register (USBOH1_USB_CLKONOFF_CTRL)

Address: USBOH1_USB_CLKONOFF_CTRL is 53FC_4000h base + 824h offset = 53FC_4824h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-													H1_AHBCLK_OFF	OTG_AHBCLK_OFF	-
Reset	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBOH1_USB_CLKONOFF_CTRL field descriptions

Field	Description
31–19 -	Reserved

Table continues on the next page...

USBOH1_USB_CLKONOFF_CTRL field descriptions (continued)

Field	Description
18 H1_AHBCLK_OFF	Enable/disable the AHB clock for Host1 This bit is used to enable/disable the AHB clock for Host1 1 turn off the AHB clock for the host1 0 turn on the AHB clock for the host1
17 OTG_AHBCLK_OFF	Enable/disable the AHB clock for OTG This bit is used to enable/disable the AHB clock for OTG 1 turn off the AHB clock for the OTG 0 turn on the AHB clock for the OTG
16–0 -	Reserved

56.6 Core Registers

USB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F8_0000	Identification register (USB_ID)	32	R	E401_FA05h	56.6.1/ 3289
53F8_0004	Hardware General (USB_HWGENERAL)	32	R	See section	56.6.2/ 3289
53F8_0008	Host Hardware Parameters (USB_HWHOST)	32	R	1002_0001h	56.6.3/ 3290
53F8_000C	Device Hardware Parameters (USB_HWDEVICE)	32	R	See section	56.6.4/ 3290
53F8_0010	TX Buffer Hardware Parameters (USB_HWTXBUF)	32	R	See section	56.6.5/ 3291
53F8_0014	RX Buffer Hardware Parameters (USB_HWRXBUF)	32	R	See section	56.6.6/ 3292
53F8_0080	General Purpose Timer #0 Load (USB_GPTIMER0LD - (non-EHCI))	32	R/W	0000_0000h	56.6.7/ 3292
53F8_0084	General Purpose Timer #0 Controller (USB_GPTIMER0CTRL - (non-EHCI))	32	R/W	0000_0000h	56.6.8/ 3293
53F8_0088	General Purpose Timer #1 Load (USB_GPTIMER1LD - (non-EHCI))	32	R/W	0000_0000h	56.6.9/ 3294
53F8_008C	General Purpose Timer #1 Controller (USB_GPTIMER1CTRL - (non-EHCI))	32	R/W	0000_0000h	56.6.10/ 3295

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F8_0090	System Bus Interface Control Register (USB_SBUSCFG)	32	R/W	0000_0002h	56.6.11/ 3296
53F8_0100	Capability Register Length (USB_CAPLENGTH - EHCI Compliant)	8	R	40h	56.6.12/ 3297
53F8_0102	Host Controller Interface Version (USB_HCIVERSION - EHCI Compliant)	16	R	0100h	56.6.13/ 3297
53F8_0104	Host Controller Structural Parameters (USB_HCSPARAMS - EHCI Compliant)	32	R	See section	56.6.14/ 3298
53F8_0108	Host Controller Capability Parameters (USB_HCCPARAMS - EHCI Compliant)	32	R	0000_0006h	56.6.15/ 3300
53F8_0120	Device Controller Interface Version (USB_DCIVERSION - (non-EHCI))	16	R	See section	56.6.16/ 3301
53F8_0124	Device Controller Capability Parameters (USB_DCCPARAMS - (non-EHCI))	32	R	See section	56.6.17/ 3302
53F8_0140	USB Command Register (USB_USBCMD)	32	R/W	See section	56.6.18/ 3303
53F8_0144	USB Status Register (USB_USBSTS)	32	R/W	See section	56.6.19/ 3307
53F8_0148	Interrupt Enable Register (USB_USBINTR)	32	R/W	0000_0000h	56.6.20/ 3310
53F8_014C	USB Frame Index (USB_FRINDEX)	32	R/W	0000_0000h	56.6.21/ 3312
53F8_0154	Frame List Base Address (USB_PERIODICLISTBASE)	32	R/W	0000_0000h	56.6.22/ 3313
53F8_0154	Device Address (USB_DEVICEADDR)	32	R/W	0000_0000h	56.6.23/ 3314
53F8_0158	Next Asynch. Address (USB_ASYNCLISTADDR)	32	R/W	0000_0000h	56.6.24/ 3314
53F8_0158	Endpoint List Address (USB_ENDPTLISTADDR)	32	R/W	0000_0000h	56.6.25/ 3315
53F8_0160	Programmable Burst Size (USB_BURSTSIZE)	32	R/W	0000_0000h	56.6.26/ 3316
53F8_0164	TX FIFO Fill Tuning (USB_TXFILLTUNING)	32	R/W	0002_0000h	56.6.27/ 3316
53F8_016C	IC_USB enable and voltage negotiation (USB_IC_USB)	32	R/W	0000_0000h	56.6.28/ 3318
53F8_0170	ULPI Viewport (USB_ULPIVIEW)	32	R/W	0000_0000h	56.6.29/ 3319
53F8_0184	Port Status and Control (USB_PORTSC)	32	R/W	See section	56.6.30/ 3322

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F8_01A4	On-The-Go Status & control (USB_OTGSC)	32	R/W	0000_0020h	56.6.31/ 3329
53F8_01A8	USB Device Mode (USB_USBMODE)	32	R/W	See section	56.6.32/ 3332
53F8_01AC	Endpoint Setup Status (USB_ENDPTSETUPSTAT)	32	R/W	0000_0000h	56.6.33/ 3333
53F8_01B0	Endpoint Initialization (USB_ENDPTPRIME)	32	R/W	0000_0000h	56.6.34/ 3334
53F8_01B4	Endpoint De-Initialize (USB_ENDPTFLUSH)	32	R/W	0000_0000h	56.6.35/ 3334
53F8_01B8	Endpoint Status (USB_ENDPTSTAT)	32	R	0000_0000h	56.6.36/ 3335
53F8_01BC	Endpoint Complete (USB_ENDPTCOMPLETE)	32	R/W	0000_0000h	56.6.37/ 3336
53F8_01C0	Endpoint Control0 (USB_ENDPTCTRL0)	32	R/W	0080_0080h	56.6.38/ 3337
53F8_01C4	Endpoint Controln (USB_ENDPTCTRL1)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01C8	Endpoint Controln (USB_ENDPTCTRL2)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01CC	Endpoint Controln (USB_ENDPTCTRL3)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01D0	Endpoint Controln (USB_ENDPTCTRL4)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01D4	Endpoint Controln (USB_ENDPTCTRL5)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01D8	Endpoint Controln (USB_ENDPTCTRL6)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01DC	Endpoint Controln (USB_ENDPTCTRL7)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01E0	Endpoint Controln (USB_ENDPTCTRL8)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01E4	Endpoint Controln (USB_ENDPTCTRL9)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01E8	Endpoint Controln (USB_ENDPTCTRL10)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01EC	Endpoint Controln (USB_ENDPTCTRL11)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01F0	Endpoint Controln (USB_ENDPTCTRL12)	32	R/W	0000_0000h	56.6.39/ 3338

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F8_01F4	Endpoint Controln (USB_ENDPTCTRL13)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01F8	Endpoint Controln (USB_ENDPTCTRL14)	32	R/W	0000_0000h	56.6.39/ 3338
53F8_01FC	Endpoint Controln (USB_ENDPTCTRL15)	32	R/W	0000_0000h	56.6.39/ 3338

56.6.1 Identification register (USB_ID)

The ID register identifies the USB 2.0 OTG High-Speed core and its revision.

Address: USB_ID is 53F8_0000h base + 0h offset = 53F8_0000h

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

USB_ID field descriptions

Field	Description
31–0 ID	This field should be E401FA05h

56.6.2 Hardware General (USB_HWGENERAL)

General hardware parameters as defined in System Level Issues and Core Configuration.

Address: USB_HWGENERAL is 53F8_0000h base + 4h offset = 53F8_0004h

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

* Notes:
- Reset value is implementation dependent.

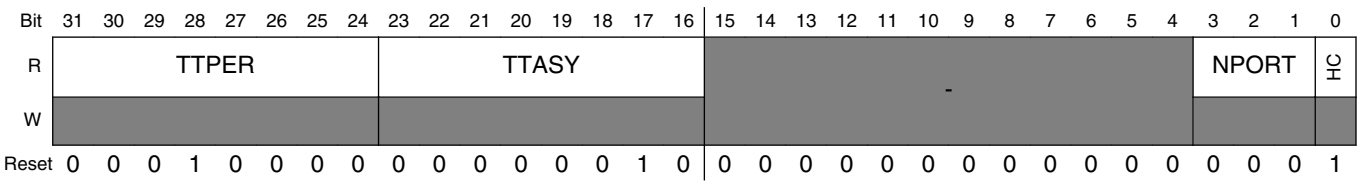
USB_HWGENERAL field descriptions

Field	Description
31–12 -	**Reserved.** These bits are reserved and should be set to zero.
11–0 HWGENERAL	This field should be 00000835h

56.6.3 Host Hardware Parameters (USB_HWHOST)

Host hardware parameters.

Address: USB_HWHOST is 53F8_0000h base + 8h offset = 53F8_0008h



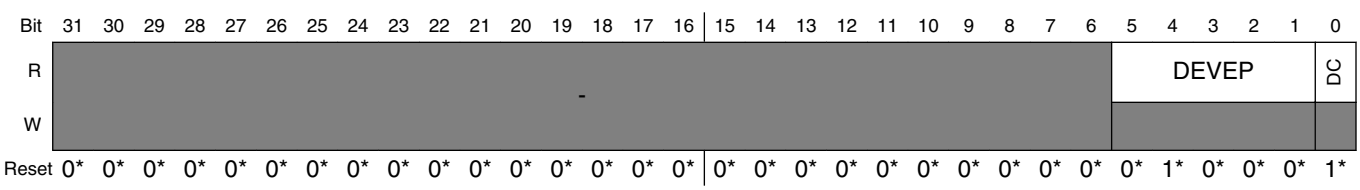
USB_HWHOST field descriptions

Field	Description
31–24 TTPER	VUSB_HS_TT_PERIODIC_CONTEXTS: should be 10h in this design
23–16 TTASY	VUSB_HS_TT_ASYNC_CONTEXTS: should be 02h in this design
15–4 -	**Reserved.** These bits are reserved and should be set to zero.
3–1 NPORT	VUSB_HS_NUM_PORT-1: should be 0 in this design
0 HC	VUSB_HS_HOST: should be 1 in this design

56.6.4 Device Hardware Parameters (USB_HWDEVICE)

Device hardware parameters. This register is only available in OTG core.

Address: USB_HWDEVICE is 53F8_0000h base + Ch offset = 53F8_000Ch



* Notes:
- Reset value is implementation dependent.

USB_HWDEVICE field descriptions

Field	Description
31–6 -	**Reserved.** These bits are reserved and should be set to zero
5–1 DEVEP	VUSB_HS_DEV_EP: Supported device endpoint number, 8 endpoints in this design
0 DC	Device Capable, 1 in this design

56.6.5 TX Buffer Hardware Parameters (USB_HWTXBUF)

TX buffer hardware parameters are as defined in System Level Issues and Core Configuration.

Address: USB_HWTXBUF is 53F8_0000h base + 10h offset = 53F8_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TXCLR	-							TXCHANADD							
W																
Reset	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXADD								TXBURST							
W																
Reset	0*	0*	0*	0*	1*	0*	1*	1*	0*	0*	0*	0*	1*	0*	0*	0*

* Notes:
- Reset value is implementation dependent.

USB_HWTXBUF field descriptions

Field	Description
31 TXCLR	VUSB_HS_TX_LOCAL_CONTEXT_REGISTERS
30–24 -	**Reserved:** These bits are reserved and should be set to zero.
23–16 TXCHANADD	VUSB_HS_TX_CHAN_ADD
15–8 TXADD	VUSB_HS_TX_ADD

Table continues on the next page...

USB_HWTXBUF field descriptions (continued)

Field	Description
7–0 TXBURST	VUSB_HS_TX_BURST

56.6.6 RX Buffer Hardware Parameters (USB_HWRXBUF)

RX buffer hardware parameters.

Address: USB_HWRXBUF is 53F8_0000h base + 14h offset = 53F8_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																RXADD								RXBURST							
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*

* Notes:
- Reset value is implementation dependent.

USB_HWRXBUF field descriptions

Field	Description
31–16 -	Reserved. These bits are reserved and should be set to zero.
15–8 RXADD	VUSB_HS_RX_ADD
7–0 RXBURST	VUSB_HS_RX_BURST

56.6.7 General Purpose Timer #0 Load (USB_GPTIMER0LD - (non-EHCI))

This register controls load value of the count timer in register USB_GPTIMER0CTRL.

Address: USB_GPTIMER0LD - (non-EHCI) is 53F8_0000h base + 80h offset = 53F8_0080h

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

USB_GPTIMER0LD - (non-EHCI) field descriptions

Field	Description
31–24 -	Reserved. These bits are reserved and should be set to zero.
23–0 GPTLD	General Purpose Timer Load Value This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value represents the time in microseconds minus 1 for the timer duration.

56.6.8 General Purpose Timer #0 Controller (USB_GPTIMER0CTRL - (non-EHCI))

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated through the use of the timer interrupts in the USBSTS and USBINTR registers.

Address: USB_GPTIMER0CTRL - (non-EHCI) is 53F8_0000h base + 84h offset = 53F8_0084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPTRUN	GPTRST	-					GPTMODE	GPTCNT[8:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPTCNT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_GPTIMER0CTRL - (non-EHCI) field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run This bit enable the GPT to run. Setting or Clearing this bit will not have an effect on the GPTCNT except. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset This bit will reload the GPTCNT with the value in GPTLD.

Table continues on the next page...

USB_GPTIMER0CTRL - (non-EHCI) field descriptions (continued)

Field	Description
	0 No action 1 Load counter value.
29–25 -	Reserved. These bits are reserved and should be set to zero.
24 GPTMODE	General Purpose Timer Mode In one-shot mode the timer will count to zero, generate an interrupt and stop until the timer is reset by software. In repeat mode the timer will count to zero, generate an interrupt and automatically reload the counter to begin again. 0 One Shot Mode 1 Repeat Mode
23–0 GPTCNT	General Purpose Timer Counter. This field is the count value of the running timer.

56.6.9 General Purpose Timer #1 Load (USB_GPTIMER1LD - (non-EHCI))

This register controls load value of the count timer in register USB_GPTIMER1CTRL.

Address: USB_GPTIMER1LD - (non-EHCI) is 53F8_0000h base + 88h offset = 53F8_0088h

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

USB_GPTIMER1LD - (non-EHCI) field descriptions

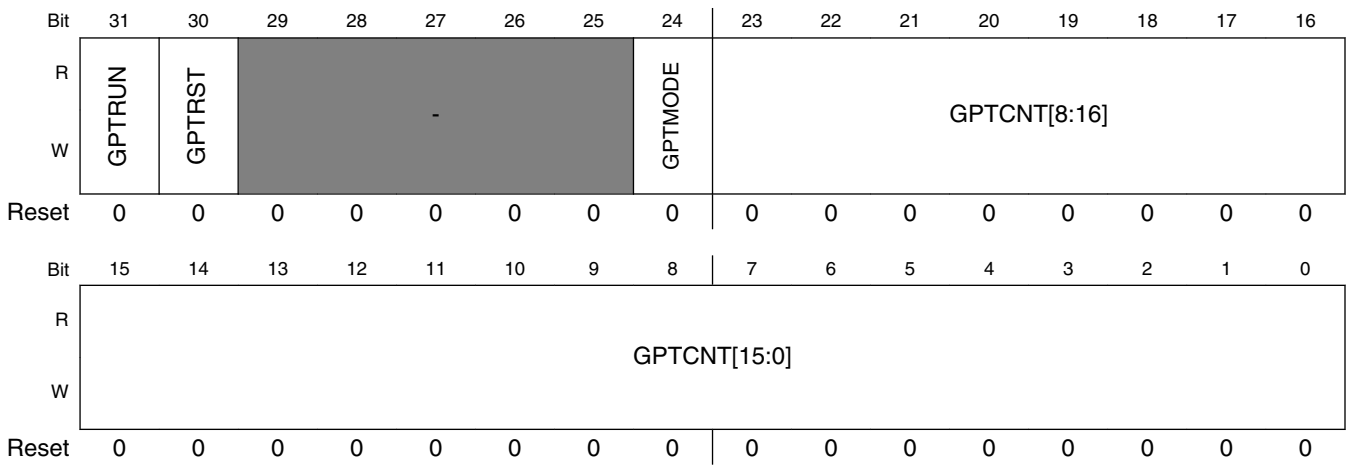
Field	Description
31–24 -	Reserved. These bits are reserved and should be set to zero.
23–0 GPTLD	General Purpose Timer Load Value This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value represents the time in microseconds minus 1 for the timer duration.

56.6.10 General Purpose Timer #1 Controller (USB_GPTIMER1CTRL - (non-EHCI))

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI1 bit in USBSTS register (See USB Status Register (USB_USBSTS)), interrupt enable bit is TIE1 bit in USBINTR register (See Interrupt Enable Register (USB_USBINTR)).

Address: USB_GPTIMER1CTRL - (non-EHCI) is 53F8_0000h base + 8Ch offset = 53F8_008Ch



USB_GPTIMER1CTRL - (non-EHCI) field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run This bit enable the GPT to run. Setting or Clearing this bit will not have an effect on the GPTCNT except. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset This bit will reload the GPTCNT with the value in GPTLD. 0 No action 1 Load counter value.
29–25 -	Reserved. These bits are reserved and should be set to zero.
24 GPTMODE	General Purpose Timer Mode

Table continues on the next page...

USB_GPTIMER1CTRL - (non-EHCI) field descriptions (continued)

Field	Description
	In one-shot mode the timer will count to zero, generate an interrupt and stop until the timer is reset by software. In repeat mode the timer will count to zero, generate an interrupt and automatically reload the counter to begin again. 0 One Shot Mode 1 Repeat Mode
23–0 GPTCNT	General Purpose Timer Counter. This field is the count value of the running timer.

56.6.11 System Bus Interface Control Register (USB_SBUSCFG)

Address: USB_SBUSCFG is 53F8_0000h base + 90h offset = 53F8_0090h

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

USB_SBUSCFG field descriptions

Field	Description
31–3 -	Reserved. These bits are reserved and should be set to zero.
2–0 AHBBRST	This is a R/W field that selects the following options for the m_hburst signal of the AMBA master interface: 000 Incremental burst of unspecified length only. With this setting the AHB master will always issue singles or unspecified length bursts. The singles will be issued when AHB alignment operations are required. With this option the AHB master will issue signal hlock whenever it's requesting the bus. On all other settings the hlock will never be asserted. This is the recommended option to achieve the best performance for the USB core (avoiding overruns and underruns in the latency FIFOs.) 001 Incremental bursts of 4 are forced. If bursts of 4 are not possible (less then 4 words remaining to transfer either reported by traffic controller or by the RX buffer estimator) the transfer will be all singles. 010 Incremental bursts of 8 are forced, unless the transfer is less then 8 words. In the later case if a new burst is due to start but it's less then 8 words, but bigger then 4 words it will do incremental bursts of 4 (INCR4). All transactions less 4 beats will be singles. All other remaining behavioral is similar to the 3'b001. 011 This case is similar to 3'b001 and 3'b010, the difference is that it can do INCR16 bursts. 100 Reserved, don't use 101 Similar to 3'b001, with the difference that it will do unspecified length bursts for accesses smaller than INCR4. The alignments (byte, word accesses and single beats) will be singles. Burst termination due to lost grant or retry/splits will also trigger unspecified length bursts. 110 Similar to 3'b101. If the burst is smaller than 8 words, it will do unspecified length, even if INCR4 is possible. If the burst is equal or greater than 8 words it will issue INCR8. 111 Similar to 3'b110, but it's able to do INCR16. Anything smaller will be done using unspecified length bursts.

56.6.12 Capability Register Length (USB_CAPLENGTH - EHCI Compliant)

This register is used to indicate which offset to add to the register base address at the beginning of the Operational Register.

Address: USB_CAPLENGTH - EHCI Compliant is 53F8_0000h base + 100h offset = 53F8_0100h

Bit	7	6	5	4	3	2	1	0
Read	CAPLENGTH							
Write								
Reset	0	1	0	0	0	0	0	0

USB_CAPLENGTH - EHCI Compliant field descriptions

Field	Description
7–0 CAPLENGTH	These bits are used as an offset to add to register base to find the beginning of the Operational Register. Default value is '40h'.

56.6.13 Host Controller Interface Version (USB_HCIVERSION - EHCI Compliant)

This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.

Address: USB_HCIVERSION - EHCI Compliant is 53F8_0000h base + 102h offset = 53F8_0102h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

USB_HCIVERSION - EHCI Compliant field descriptions

Field	Description
15–0 HCIVERSION	Host Controller Interface Version Number Default value is '10h', which means EHCI rev1.0.

56.6.14 Host Controller Structural Parameters (USB_HCSPARAMS - EHCI Compliant)

Port steering logic capabilities are described in this register.

Address: USB_HCSPARAMS - EHCI Compliant is 53F8_0000h base + 104h offset = 53F8_0104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-				N_TT				N_PTT				-			PI
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	N_CC				N_PCC				-			PPC	N_PORTS			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	1*

* Notes:
- Reset value is implementation dependent.

USB_HCSPARAMS - EHCI Compliant field descriptions

Field	Description
31–28 -	Reserved. These bits are reserved and should be set to zero.
27–24 N_TT	Number of Transaction Translators (N_TT). Default value '0000b' This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. These bits would be set to '0001b' for Multi-Port Host, and '0000b' for all other implementations. This in a non-EHCI field to support embedded TT.
23–20 N_PTT	Number of Ports per Transaction Translator (N_PTT). Default value '0000b' This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. These bits would be set to equal N_PORTS for Multi-Port Host, and '0000b' for all other implementations. This in a non-EHCI field to support embedded TT.
19–17 -	Reserved
16 PI	Port Indicators (P INDICATOR) This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator. This bit will always be 1.
15–12 N_CC	Number of Companion Controllers (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller.

Table continues on the next page...

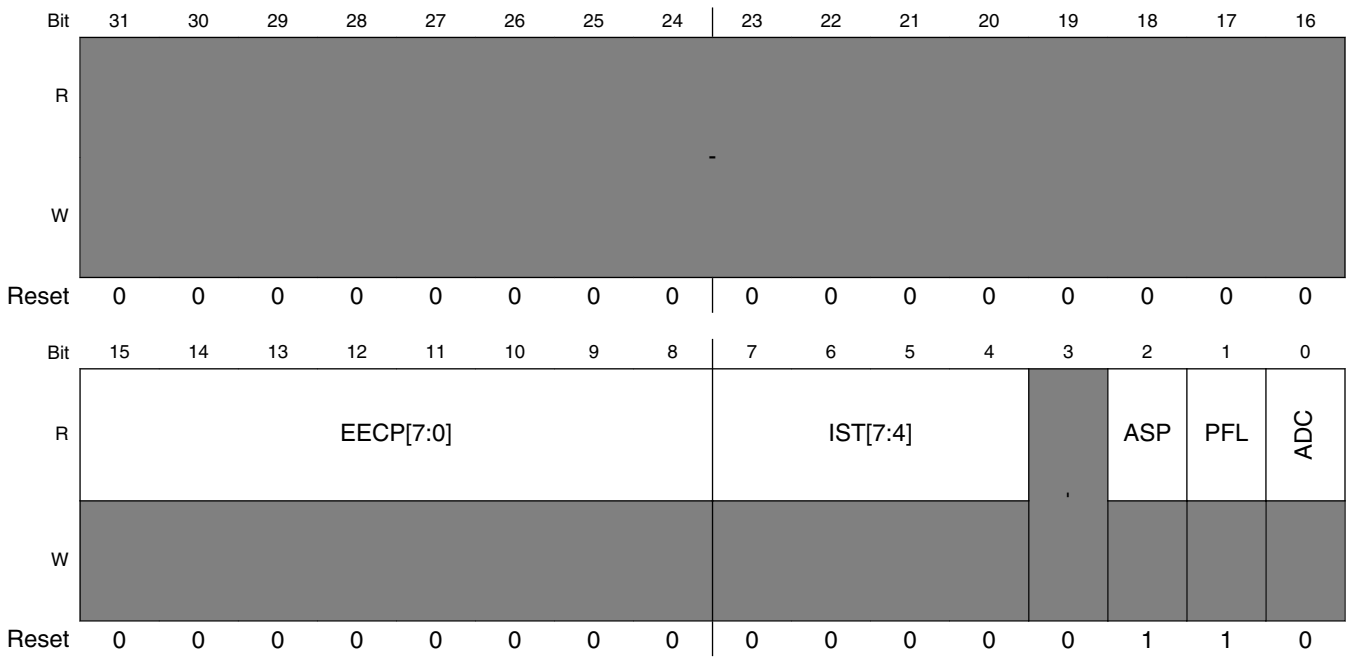
USB_HCSPARAMS - EHCI Compliant field descriptions (continued)

Field	Description
	A zero in this field indicates there are no internal Companion Controllers. Port-ownership hand-off is not supported. A value larger than zero in this field indicates there are companion USB1.1 host controller(s). Port-ownership handoffs are supported. High, Full- and Low-speed devices are supported on the host controller root ports. In this implementation this field will always be "0". 0 There is no internal Companion Controller and port-ownership hand-off is not supported. 1 There are internal companion controller(s) and port-ownership hand-offs is supported.
11–8 N_PCC	Number of Ports per Companion Controllers (N_PCC) This field indicates the number of ports supported per internal transaction translator. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. In this implementation this field will always be '0'
7–5 -	Reserved
4 PPC	Port Power Control This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register. This field will always be "0" for a device only implementation.
3–0 N_PORTS	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 1h to Fh. A zero in this field is undefined. The number of ports for a host implementation is parameterizable from 1 to 8. This field will always be 1 for device only implementation.

56.6.15 Host Controller Capability Parameters (USB_HCCPARAMS - EHCI Compliant)

This register identifies multiple mode control (time-base bit functionality), addressing capability.

Address: USB_HCCPARAMS - EHCI Compliant is 53F8_0000h base + 108h offset = 53F8_0108h



USB_HCCPARAMS - EHCI Compliant field descriptions

Field	Description
31–16 -	Reserved. These bits are reserved and should be set to zero.
15–8 EECP[7:0]	EHCI Extended Capabilities Pointer. Default = 0. This field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device. For this implemetation the field is always '0'.
7–4 IST[7:4]	Isochronous Scheduling Threshold. Default = Implementation Dependent. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be '0'.
3 -	Reserved. These bits are reserved and should be set to zero.

Table continues on the next page...

USB_HCCPARAMS - EHCI Compliant field descriptions (continued)

Field	Description
2 ASP	Asynchronous Schedule Park Capability Default = 1. If this bit is set to a one, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register. This bit is set '1b' in all 4 controller core.
1 PFL	Programmable Frame List Flag If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This bit is set '1b' in all 4 controller core.
0 ADC	64-bit Addressing Capability This bit is set '0b' in all 4 controller core, no 64-bit addressing capability is supported.

56.6.16 Device Controller Interface Version (USB_DCIVERSION - (non-EHCI))

This register indicates the two-byte BCD encoding of the device controller interface version number contained in this register.

Address: USB_DCIVERSION - (non-EHCI) is 53F8_0000h base + 120h offset = 53F8_0120h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DCIVERSION															
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*

* Notes:
- Reset value is implementation dependent.

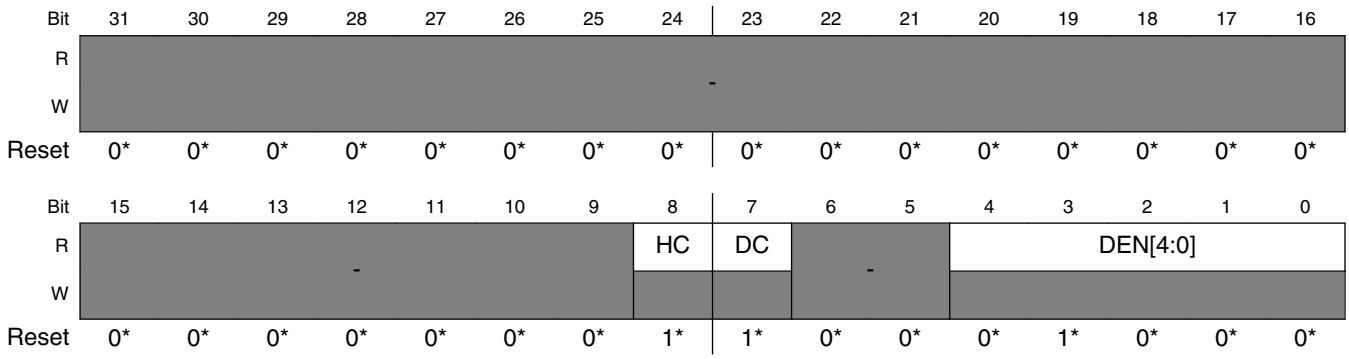
USB_DCIVERSION - (non-EHCI) field descriptions

Field	Description
15–0 DCIVERSION	Device Controller Interface Version Number

56.6.17 Device Controller Capability Parameters (USB_DCCPARAMS - (non-EHCI))

These fields describe the overall device capability of the controller.

Address: USB_DCCPARAMS - (non-EHCI) is 53F8_0000h base + 124h offset = 53F8_0124h



* Notes:
- Reset value is implementation dependent.

USB_DCCPARAMS - (non-EHCI) field descriptions

Field	Description
31–9 -	Reserved. These bits are reserved and should be set to zero.
8 HC	Host Capable When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7 DC	Device Capable When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6–5 -	Reserved. These bits are reserved and should be set to zero.
4–0 DEN[4:0]	Device Endpoint Number This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0 - 15.

56.6.18 USB Command Register (USB_USBCMD)

The serial bus host/device controller executes the command indicated in this register.

Address: USB_USBCMD is 53F8_0000h base + 140h offset = 53F8_0140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-								ITC[7:0]							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FS[2]	-	SUTW	ATDTW	ASPE	-	ASP[1:0]		LR	IAA	ASE	PSE	FS[2:0]		RST	RS
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is implementation dependent.

USB_USBCMD field descriptions

Field	Description
31–24 -	Reserved. These bits are reserved and should be set to zero.
23–16 ITC[7:0]	Interrupt Threshold Control -Read/Write. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 0x00 Immediate (no threshold) 0x01 1 micro-frame 0x02 2 micro-frames 0x04 4 micro-frames 0x08 8 micro-frames 0x10 16 micro-frames 0x20 32 micro-frames 0x40 64 micro-frames

Table continues on the next page...

USB_USBCMD field descriptions (continued)

Field	Description
15 FS[2]	See also bits 3-2 Frame List Size - (Read/Write or Read Only). [host mode only] This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. **NOTE:** This field is made up from USBCMD bits 15, 3 and 2. Value Meaning 000 1024 elements (4096 bytes) Default value 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes)
14 -	Reserved
13 SUTW	Setup TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (SLOM bit in USB core register USBMODE, see USB Device Mode (USB_USBMODE)) then there is a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software. This bit would also be cleared by hardware when a hazard detected. For more information on the use of this bit, see the Device Operational Model section of the USB-HS OTG High-Speed USB On-The-Go DEV reference manual.
12 ATDTW	Add dTD TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit would also be cleared by hardware when state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized. For more information on the use of this bit, see the Device Operational Model section of the USB-HS OTG High-Speed USB On-The-Go DEV reference manual.
11 ASPE	Asynchronous Schedule Park Mode Enable - Read/Write. If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to '1b' in this implementation.
10 -	Reserved
9–8 ASP[1:0]	Asynchronous Schedule Park Mode Count - Read/Write.

Table continues on the next page...

USB_USBCMD field descriptions (continued)

Field	Description
	If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is Read-Only. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when *Park Mode Enable* is a one as this will result in undefined behavior. This field is set to 3h in all 4 controller core.
7 LR	Light Host/Device Controller Reset (OPTIONAL) - Read Only. Not Implemented. This field will always be '0b'.
6 IAA	Interrupt on Async Advance Doorbell - Read/Write. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results.
5 ASE	Asynchronous Schedule Enable - Read/Write. Default 0b. Default = 0b. This bit controls whether the host controller skips processing the Asynchronous Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Asynchronous Schedule. 1 Use the ASYNCLISTADDR register to access the Asynchronous Schedule.
4 PSE	Periodic Schedule Enable- Read/Write. Default 0b. Default = 0b. This bit controls whether the host controller skips processing the Periodic Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Periodic Schedule 1 Use the PERIODICLISTBASE register to access the Periodic Schedule.
3–2 FS[2:0]	Frame List Size - (Read/Write or Read Only). Default 000b. This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. Only the host controller uses this field. Values meaning 000 1024 elements (4096 bytes) Default value 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes)

Table continues on the next page...

USB_USBCMD field descriptions (continued)

Field	Description
1 RST	Controller Reset (RESET) - Read/Write. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host operation mode: When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. Device operation mode: When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, because the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop (RS) - Read/Write. Default 0b. 1=Run. 0=Stop. Host operation mode: When set to '1b', the Controller proceeds with the execution of the schedule. The Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the controller is in the Halted state (that is, HCHalted in the USBSTS register is a one). Device operation mode: Writing a one to this bit will cause the controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the controller has been properly initialized. Writing a 0 to this will cause a detach event.

56.6.19 USB Status Register (USB_USBSTS)

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them.

Address: USB_USBSTS is 53F8_0000h base + 144h offset = 53F8_0144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-						TI1	TI0	-				UPI	UAI	-	NAKI
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AS	PS	RCL	HCH	-			SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is implementation dependent.

USB_USBSTS field descriptions

Field	Description
31–26 -	Reserved. These bits are reserved and should be set to zero.
25 TI1	General Purpose Timer Interrupt 1(GPTINT1)--R/WC. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero, writing a one to this bit will clear it.
24 TI0	General Purpose Timer Interrupt 0(GPTINT0)--R/WC. This bit is set when the counter in the GPTIMER0CTRL register transitions to zero, writing a one to this bit clears it.
23–20 -	Reserved. These bits are reserved and should be set to zero.
19 UPI	USB Host Periodic Interrupt -- RWC. Default = 0b.

Table continues on the next page...

USB_USBSTS field descriptions (continued)

Field	Description
	This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected and the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than expected. This bit is not used by the device controller and will always be zero.
18 UAI	USB Host Periodic Interrupt -- RWC. Default = 0b. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the asynchronous schedule. This bit is also set by the Host Controller when a short packet is detected and the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than expected. This bit is not used by the device controller and will always be zero.
17 -	Reserved. These bits are reserved and should be set to zero.
16 NAKI	NAK Interrupt Bit--RO. Default = 0b. This bit is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when all Enabled TX/RX Endpoint NAK bits are cleared. This bit is not used by the host controller and will always be zero.
15 AS	Asynchronous Schedule Status - Read Only. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
14 PS	Periodic Schedule Status - Read Only. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
13 RCL	Reclamation - Read Only. This is a read-only status bit used to detect an empty asynchronous schedule. Only used in the host operation mode.
12 HCH	HCHalted - Read Only. Default = 1. This bit is a zero whenever the Run/Stop bit is a one. The Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Controller hardware (for example, an internal error). Only used in the host operation mode.

Table continues on the next page...

USB_USBSTS field descriptions (continued)

Field	Description
	Default value is '0b' for OTG core, and '1b' for Host1/Host2/Host3 core. This is because OTG core is not operating as host in default. Please see CM bit in USBMODE register.
11–9 -	Reserved
8 SLI	DCSuspend - R/WC. When a controller enters a suspend state from an active state, this bit will be set to a one. The device controller clears the bit upon exiting from a suspend state. Only used in device operation mode.
7 SRI	SOF Received - R/WC. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Because the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it.
6 URI	USB Reset Received - R/WC. Default = 0. When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used in device operation mode.
5 AAI	Interrupt on Async Advance - R/WC. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the n_USBCMD register. This status bit indicates the assertion of that interrupt source. Only used in host operation mode.
4 SEI	System Error- R/WC. This bit is will be set to '1b' when an Error response is seen to a read on the system interface.
3 FRI	Frame List Rollover - R/WC. The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used in host operation mode.
2 PCI	Port Change Detect - R/WC. The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/ Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively.

Table continues on the next page...

USB_USBSTS field descriptions (continued)

Field	Description
1 UEI	USB Error Interrupt (USBERRINT) - R/WC. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set See Section (Reference Host Operation Model: Transfer/Transaction Based Interrupt - i.e. 4.15.1 in EHCI Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com) for a complete list of host error interrupt conditions. See section Device Error Matrix in the USB-HS OTG High-Speed USB On-The-Go DEV reference manual. The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT) - R/WC. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

56.6.20 Interrupt Enable Register (USB_USBINTR)

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt source is active. The USB Status register (n_USBSTS) still shows interrupt sources even if they are disabled by the n_USBINTR register, allowing polling of interrupt events by the software.

Address: USB_USBINTR is 53F8_0000h base + 148h offset = 53F8_0148h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-						TIE1	TIE0	-				UPIE	UAIE	-	NAKE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-							SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_USBINTR field descriptions

Field	Description
31–26 -	Reserved
25 TIE1	General Purpose Timer #1 Interrupt Enable When this bit is one and the GPTINT1 in USBSTS register is a one the controller will issue an interrupt. The interrupt is acknowledge by software clear the GPTINT1 bit.
24 TIE0	General Purpose Timer #0 Interrupt Enable When this bit is one and the GPTINT0 in USBSTS register is a one the controller will issue an interrupt. The interrupt is acknowledge by software clear the GPTINT0 bit.
23–20 -	Reserved
19 UPIE	USB Host Periodic Interrupt Enable When this bit is a one, and the UPI bit in the EXTSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UPI bit.
18 UAIE	USB Host Asynchronous Interrupt Enable When this bit is a one, and the UAI bit in the EXTSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UAI bit.
17 -	Reserved
16 NAKE	NAK Interrupt Enable When this bit is a one, and the NAKI bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the NAKI bit.
15–9 -	Reserved
8 SLE	Sleep Interrupt Enable When this bit is a one, and the DCSuspend bit in the USBSTS register transitions, the device controller will issue an interrupt. The interrupt is acknowledged by software writing a one to the DCSuspend bit. Only used in device operation mode.
7 SRE	SOF Received Interrupt Enable When this bit is a one, and the SOF Received bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the SOF Received bit.
6 URE	USB Reset Interrupt Enable When this bit is a one, and the USB Reset Received bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used in device operation mode.
5 AAE	Async Advance Interrupt Enable When this bit is a one, and the Interrupt on Async Advance bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used in host operation mode.

Table continues on the next page...

USB_USBINTR field descriptions (continued)

Field	Description
4 SEE	System Error Interrupt Enable When this bit is a one, and the System Error bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the System Error bit. Only used in host operation mode.
3 FRE	Frame List Rollover Interrupt Enable When this bit is a one, and the Frame List Rollover bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used in host operation mode.
2 PCE	Port Change Detect Interrupt Enable When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit.
1 UEE	USB Error Interrupt Enable When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register.
0 UE	USB Interrupt Enable When this bit is a one, and the USBINT bit in the USBSTS register is a one, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit.

56.6.21 USB Frame Index (USB_FRINDEX)

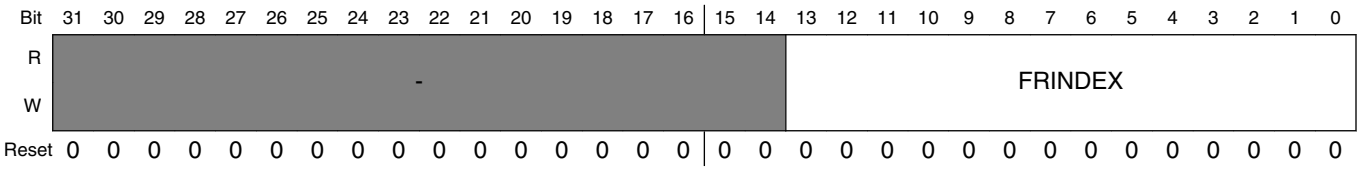
This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the n_USBCMD register.

This register must be written as a DWord. Byte writes produce-undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the

SOF value and FRINDEX [2:0] will be set to zero (that is, SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be increment (that is, SOF for 125 us micro-frame.).

Address: USB_FRINDEX is 53F8_0000h base + 14Ch offset = 53F8_014Ch



USB_FRINDEX field descriptions

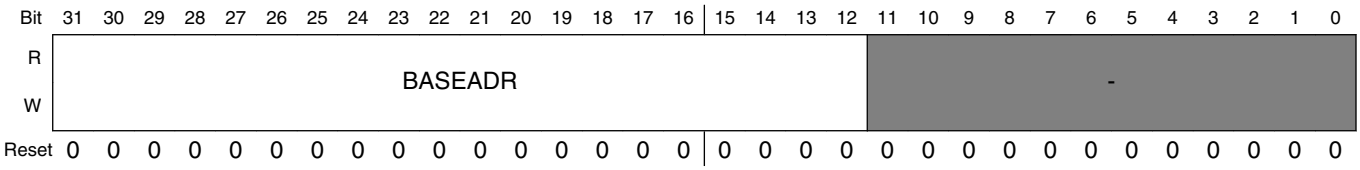
Field	Description
31–14 -	Reserved. These bits are reserved and should be set to zero.
13–0 FRINDEX	Frame Index. The value, in this register, increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current microframe. 000 (1024) 12 001 (512) 11 010 (256) 10 011 (128) 9 100 (64) 8 101 (32) 7 110 (16) 6 111 (8) 5

56.6.22 Frame List Base Address (USB_PERIODICLISTBASE)

Host Controller only

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

Address: USB_PERIODICLISTBASE is 53F8_0000h base + 154h offset = 53F8_0154h



USB_PERIODICLISTBASE field descriptions

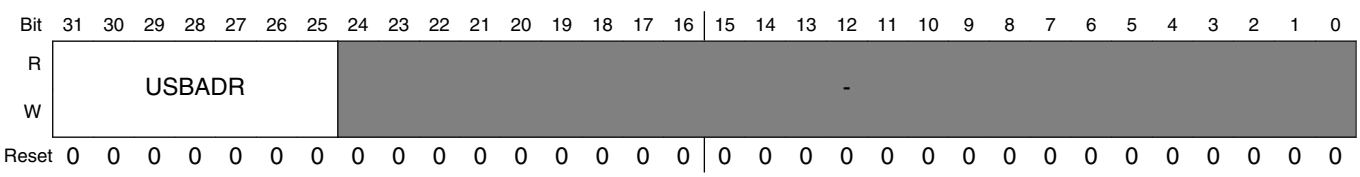
Field	Description
31–12 BASEADR	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
11–0 -	Reserved.Must be written as zeros. During runtime, the values of these bits are undefined.

56.6.23 Device Address (USB_DEVICEADDR)

Device Controller only

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

Address: USB_DEVICEADDR is 53F8_0000h base + 154h offset = 53F8_0154h



USB_DEVICEADDR field descriptions

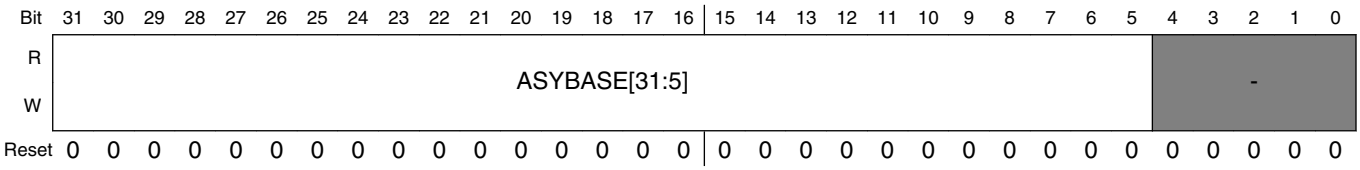
Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address
24–0 -	Reserved.Must be written as zeros. During runtime, the values of these bits are undefined.

56.6.24 Next Asynch. Address (USB_ASYNCLISTADDR)

Host Controller only

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

Address: USB_ASYNCLISTADDR is 53F8_0000h base + 158h offset = 53F8_0158h



USB_ASYNCLISTADDR field descriptions

Field	Description
31–5 ASYBASE[31:5]	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). Only used by the host controller.
4–0 -	Reserved. These bits are reserved and their value has no effect on operation.

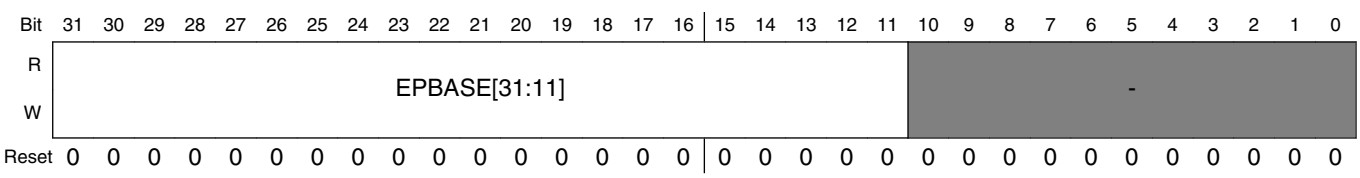
56.6.25 Endpoint List Address (USB_ENDPTLISTADDR)

Device Controller only

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed 64-byte.

Address: USB_ENDPTLISTADDR is 53F8_0000h base + 158h offset = 53F8_0158h



USB_ENDPTLISTADDR field descriptions

Field	Description
31–11 EPBASE[31:11]	Endpoint List Pointer(Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Head (OH) (that is, one queue head per endpoint & direction).
10–0 -	Reserved. These bits are reserved and their value has no effect on operation.

56.6.26 Programmable Burst Size (USB_BURSTSIZE)

This register is used to control dynamically change the burst size used during data movement on the initiator (master) interface.

Address: USB_BURSTSIZE is 53F8_0000h base + 160h offset = 53F8_0160h

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

USB_BURSTSIZE field descriptions

Field	Description
31–17 -	Reserved. These bits are reserved and their value has no effect on operation.
16–8 TXPBURST	Programmable TX Burst Size. (Read/Write) Default is the constant VUSB_HS_TX_BURST. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
7–0 RXPBURST	Programmable RX Burst Size. (Read/Write) Default is the constant VUSB_HS_RX_BURST. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

56.6.27 TX FIFO Fill Tuning (USB_TXFILLTUNING)

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total Packet Flight Time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total Packet Time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a "back-off" event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the n_TSCHHEALTH (T_{ff}) described below.

Address: USB_TXFILLTUNING is 53F8_0000h base + 164h offset = 53F8_0164h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-										TXFIFOTHRES					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TXSCHHEALTH			-					TXSCHOH							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_TXFILLTUNING field descriptions

Field	Description
31–22 -	Reserved. These bits are reserved and their value has no effect on operation.
21–16 TXFIFOTHRES	FIFO Burst Threshold. (Read/Write)[Default = 2] This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
15–13 TXSCHHEALTH	Scheduler Health Counter. (Read/Write To Clear)[Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
12–8 -	Reserved. These bits are reserved and their value has no effect on operation.

Table continues on the next page...

USB_TXFILLTUNING field descriptions (continued)

Field	Description
7–0 TXSCHOH	Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode for OTG and SPH. The time unit represented in this register is always 1.267 the MPH product.

56.6.28 IC_USB enable and voltage negotiation (USB_IC_USB)

This register enable and controls the IC-USB FS/LS transceiver.

Address: USB_IC_USB is 53F8_0000h base + 16Ch offset = 53F8_016Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IC8	IC_VDD8			IC7	IC_VDD7			IC6	IC_VDD6			IC5	IC_VDD5			IC4	IC_VDD4			IC3	IC_VDD3			IC2	IC_VDD2			IC1	IC_VDD1		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_IC_USB field descriptions

Field	Description
31 IC8	Inter-Chip transceiver enable. These bits enables the Inter-Chip transceiver for each port (for MPH case). To enable the interface, the bits PTS must be set to 11 in the portsc. Writing a '1' to each bit selects the IC_USB interface for that port. If the Controller is not MultiPort, IC8 to IC2 will be '0' and Read-Only. If ICx is enabled, IOMUX must set the DP/DM pad into loopback mode.
30–28 IC_VDD8	These bits selects which voltage is being supplied to the peripheral through each port (MPH case) This field is read-only and set to 000 in case of a device controller. IC_VDD8 to IC_VDD2 are read-only and set to 000 in case the controller is not 000 No voltage 001 1.0 V 010 1.2 V 011 1.5 V 100 1.8 V 101 3.0 V 110 Reserved 111 Reserved
27 IC7	See description at bit 31
26–24 IC_VDD7	See description at bits 30-28

Table continues on the next page...

USB_IC_USB field descriptions (continued)

Field	Description
23 IC6	See description at bit 31
22–20 IC_VDD6	See description at bits 30-28
19 IC5	See description at bit 31
18–16 IC_VDD5	See description at bits 30-28
15 IC4	See description at bit 31
14–12 IC_VDD4	See description at bits 30-28
11 IC3	See description at bit 31
10–8 IC_VDD3	See description at bits 30-28
7 IC2	See description at bit 31
6–4 IC_VDD2	See description at bits 30-28
3 IC1	See description at bit 31
2–0 IC_VDD1	See description at bits 30-28

56.6.29 ULPI Viewport (USB_ULPIVIEW)

The register provides indirect access to the ULPI PHY register set. Although the core performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

NOTE

Writes to the ULPI through the ULPI Viewport can substantially harm standard USB operations. Most of the ULPI register settings are under direct control of the USB core and should not be changed by software through viewport writes as this may harm USB operations.

NOTE

Executing read operations though the ULPI Viewport should have no harmful side effects to standard USB operations.

NOTE

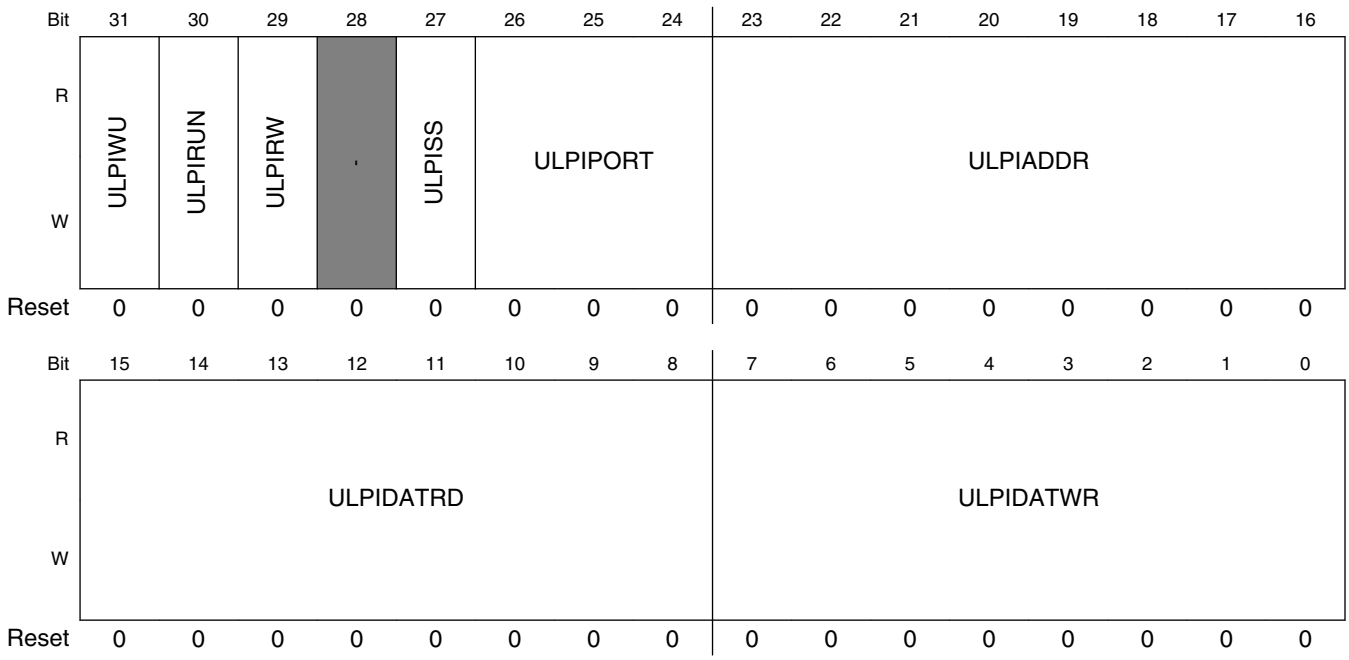
The ULPI Viewport is only synthesized in the design if the ULPI option has been purchased & installed and the constant VUSB_HS_PHY_ULPI is set to 1. If the ULPI interface is not enabled, this register always read zeros.

There are two operations that can be performed with the ULPI Viewport, wake-up and read /write operations. The wake-up operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wake-up operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or carkit mode. The ULPI state can be determined by reading the sync. state bit (ULPISS). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPISS indicates a '0' then read/write operations will not be able execute. Undefined behavior will result if ULPISS = 0 and a read or write operation is performed. To execute a wake-up operation, write all 32-bits of the ULPI Viewport where ULPIPORT is constructed appropriately and the ULPIWU bit is a '1' and ULPIRUN bit is a '0'. Poll the ULPI Viewport until ULPIWU is zero for the operation to complete.

To execute a read or write operation, write all 32-bits of the ULPI Viewport where ULPIDATWR, ULPIADDR, ULPIPORT, ULPIRW are constructed appropriately and the ULPIRUN bit is a '1'. Poll the ULPI Viewport until ULPIRUN is zero for the operation to complete. Once ULPIRUN is zero, the ULPIDATRD will be valid if the operation was a read.

The polling method above could also be replaced and interrupt driven using the ULPI interrupt defined in the USB Status Register (USB_USBSTS) and Interrupt Enable Register (USB_USBINTR) registers. When a wake-up or read/write operation complete, the ULPI interrupt will be set.

Address: USB_ULPIVIEW is 53F8_0000h base + 170h offset = 53F8_0170h



USB_ULPIVIEW field descriptions

Field	Description
31 ULPIWU	ULPI wake-up - Read/Write. Writing the 1 to this bit will begin the wake-up operation. The bit will automatically transition to 0 after the wake-up is complete. Once this bit is set, the driver can not set it back to 0. Note: The driver must never execute a wake-up and a read/write operation at the same time.
30 ULPIRUN	ULPI Read/Write Run - Read/Write. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver can not set it back to 0. **NOTE:** The driver must never execute a wake-up and a read/write operation at the same time.
29 ULPIRW	ULPI Read/Write Control - Read/Write. (0) - Read; (1) - Write. This bit selects between running a read or write operation.
28 -	Reserved. This bit is reserved and its value has no effect on operation.
27 ULPISS	ULPI Sync State - Read Only. (1) - Normal Sync. State. (0) In another state (that is, carkit, serial, low power) This bit represents the state of the ULPI interface. Before reading this bit, the ULPIPORT field should be set accordingly if used with the multi-port host. Otherwise, this field should always remain 0.
26–24 ULPIPORT	ULPI Port Number - Read/Write. For the wake-up or read/write operation to be executed, this value selects the port number the ULPI PHY is attached to in the multi-port host. The range is 0 to 7. This field should always be written as a 0 for the non-multi port products.

Table continues on the next page...

USB_ULPIVIEW field descriptions (continued)

Field	Description
23–16 ULPIADDR	ULPI Data Address - Read/Write. When a read or write operation is commanded, the address of the operation is written to this field.
15–8 ULPIDATRD	ULPI Data Read - Read Only. After a read operation completes, the result is placed in this field.
7–0 ULPIDATWR	ULPI Data Write - Read/Write. When a write operation is commanded, the data to be sent is written to this field.

56.6.30 Port Status and Control (USB_PORTSC)

Host Controller

A host controller could implement one to eight port status and control registers. The number is determined by N_PORTs bits in HWSPARAMs register (please see Host Controller Structural Parameters (USB_HCSPARAMS - EHCI Compliant)). Software could read this parameter register to determine how many ports need service.

All 4 controller cores are Single-Port Host, so there is only one port status and control register for each controller core.

This register is only reset by power on reset or controller core reset. The initial conditions of a port are:

- No device connected
- Port disabled

If the port supports power control, this state remains until port power is supplied (by software).

Device Controller

A device controller has only port register one (portsc) and it does not support power control. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Address: USB_PORTSC is 53F8_0000h base + 184h offset = 53F8_0184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PTS		STS	PTW	PSPD		PFSC	PHCD	WKOC	WKDC	WKCN	PTC[3:0]				PIC[1:0]	PO		PP	LS[1:0]		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS	
Reset	0*	0*	0*	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is implementation dependent.

USB_PORTSC field descriptions

Field	Description
31–30 PTS	Parallel Transceiver Select - Read/Write. This register bit pair is used in conjunction with the configuration constant VUSB_HS_PHY_TYPE to control which parallel transceiver interface is selected. If VUSB_HS_PHY_TYPE is set for 0,1,2, 3, 8 or 10 then this bit is read only. If VUSB_HS_PHY_TYPE is 4,5, 6, 7, 9 or 11then this bit is read/write. **NOTE:** this field is made up from PORTSCx bits 25, 30 and 31. VUSB_HS_PHY_TYPE is 0 in this implementation
29 STS	Serial Transceiver Select - Read/Write. This register bit is used in conjunction with the configuration constant VUSB_HS_PHY_SERIAL to control whether the parallel or serial transceiver interface is selected for FS and LS operation. The Serial Interface Engine can be used in combination with the UTMI+ or ULPI physical interface to provide FS/LS signaling instead of the parallel interface. If VUSB_HS_PHY_SERIAL is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY_SERIAL is 3 or 4 then this bit is read/write. This bit has no effect unless Parallel Transceiver Select is set to UTMI+ or ULPI. The Serial/1.1 physical interface will use the Serial Interface Engine for FS/LS signaling regardless of this bit value. **NOTE:** This bit is reserved for future operation and is a placeholder adding dynamic use of the serial engine in accord with UMTI+ and ULPI characterization logic. This bit is not defined in the EHCI specification
28 PTW	Parallel Transceiver Width - Read/Write. This register bit is used in conjunction with the configuration constant VUSB_HS_PHY8_16 to control whether the data bus width of the UTMI transceiver interface. If VUSB_HS_PHY8_16 is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY8_16 is 2 or 3 then this bit is read/write. This bit is reset to 1 if VUSB_HS_PHY8_16 selects a default UTMI interface width of 16-bits else it is reset to 0. This bit has no effect if serial interface engine is used. These register bits are implementation dependent. This bit is not defined in the EHCI specification. 1 Select the 8-bit UTMI interface [60MHz] 2 Select the 16-bit UTMI interface [30MHz]
27–26 PSPD	Port Speed - Read Only. This register field indicates the speed at which the port is operating. For HS mode operation in the host controller and HS/FS operation in the device controller the port routing steers data to the Protocol engine. For FS and LS mode operation in the host controller, the port routing steers data to the Protocol Engine w/ Embedded Transaction Translator.

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
	00 Full Speed 01 Low Speed 10 High Speed 11 Undefined
24 PFSC	Port Force Full Speed Connect - Read/Write. Default = 0b. When this bit is set to '1b', the port will be forced to only connect at Full Speed, It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is not defined in the EHCI specification and is for debugging purposes. 1 Forced to full speed 0 Normal operation
23 PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD) - Read/Write. Default = 0b. When this bit is set to '1b', the PHY clock is disabled. Reading this bit will indicate the status of the PHY clock. **NOTE:** The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend when the device is not running (USBCMD Run/Stop=0b) or the host has signaled suspend (portsc SUSPEND=1b). PHY Low power suspend will be cleared automatically when the host initials resume. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. This bit is not defined in the EHCI specification. 1 Disable PHY clock 0 Enable PHY clock
22 WKOC	Wake on Over-current Enable (WKOC_E) - Read/Write. Default = 0b. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This bit is output from the controller as signal pwrctl_wake_ovrcurr_en (OTG/host core only) for use by an external power control circuit. This field is zero if *Port Power* is zero.
21 WKDC	Wake on Disconnect Enable (WKDSCNNT_E) - Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit. This field is zero if *Port Power* is zero or in device mode.
20 WKCN	Wake on Connect Enable (WKCNNT_E) - Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit. This field is zero if *Port Power* is zero or in device mode.

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
19–16 PTC[3:0]	Port Test Control - Read/Write. Default = 0000b. Refer to the operational model for using these test modes and the USB Specification Revision 2.0, Chapter 7 for details on each test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. **NOTE:** *Low speed operations are not supported as a peripheral device.* Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000 TEST_MODE_DISABLE 0001 J_STATE 0010 K_STATE 0011 SE0 (host) / NAK (device) 0100 Packet 0101 FORCE_ENABLE_HS 0110 FORCE_ENABLE_FS 0111 FORCE_ENABLE_LS 1000-1111 Reserved
15–14 PIC[1:0]	Port Indicator Control - Read/Write. Default = Ob. Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is zero if *Port Power* is zero. Bit Value Meaning 00 Port indicators are off 01 Amber 10 Green 11 Undefined
13 PO	Port Owner-Read/Write. Default = 0. This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner handoff is not supported in all 4 controller cores, therefore this bit will always be 0.
12 PP	Port Power (PP)-Read/Write or Read Only. The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0 1b *Read Only - Host controller does not have port power control switches. Each port is hard-wired to power.*

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
	1 1b/0b - *Read/Write. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, etc.* When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitional by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in all 4 controller cores (PPC = 1).
11–10 LS[1:0]	Line Status-Read Only. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. In host mode, the use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of linestate by the device controller driver is not necessary. The encoding of the bits are: Bits [11:10] Meaning 00 SE0 10 J-state 01 K-state 11 Undefined
9 HSP	High-Speed Port - Read Only. Default = 0b. When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. **NOTE:** HSP is redundant with PSPD(bit 27, 26) but remained for compatibility.
8 PR	Port Reset - Read/Write or Read Only. Default = 0b. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. Default 0. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. *This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.* In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. This field is zero if *Port Power* is zero.
7 SUSP	Suspend - Read/Write or Read Only. Default = 0b. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
	When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. **NOTE:** The bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the *Force Port Resume* bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (that is, *Port enabled* bit is a zero) the results are undefined. This field is zero if *Port Power* is zero in host mode. In Device Mode: Read Only. In device mode this bit is a read only status bit.
6 FPR	Force Port Resume -Read/Write. 1= Resume detected/driven on port. 0=No resume (K-state) detected/ driven on port. Default = 0. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the *Port Change Detect* bit in the USBSTS register is also set to one. *This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver.* Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no affect because the port controller will time the resume operation clear the bit the port control state switches to HS or FS idle. This field is zero if *Port Power* is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition detected, the *Port Change Detect* bit in the USBSTS register is also set to one.
5 OCC	Over-current Change-R/WC. Default=0. This bit is set '1b' by hardware when there is a change to Over-current Active. Software can clear this bit by writing a one to this bit position. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit shall always be 0.
4 OCA	Over-current Active-Read Only. Default 0. This bit will automatically transition from one to zero when the over current condition is removed. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition.

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
	For device-only implementations this bit shall always be 0. 1 This port currently has an over-current condition 0 This port does not have an over-current condition.
3 PEC	Port Enable/Disable Change-R/WC. 1=Port enabled/disabled status has changed. 0=No change. Default = 0. In Host Mode: For the root hub, this bit is set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if *Port Power* is zero. In Device mode: The device port is always enabled, so this bit is always '0b'.
2 PE	Port Enabled/Disabled-Read/Write. 1=Enable. 0=Disable. Default 0. In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. **NOTE:** the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if *Port Power* is zero in host mode. In Device Mode: The device port is always enabled, so this bit is always '1b'.
1 CSC	Connect Status Change-R/WC. 1 =Change in Current Connect Status. 0=No change. Default 0. In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it. This field is zero if *Port Power* is zero in host mode. In Device Mode: This bit is undefined in device controller mode.
0 CCS	Current Connect Status-Read Only. In Host Mode: 1=Device is present on port. 0=No device is present. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the *Connect Status Change* bit (Bit 1) to be set. This field is zero if *Port Power* is zero in host mode. In Device Mode:

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
	1=Attached. 0=Not Attached. Default=0. A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

56.6.31 On-The-Go Status & control (USB_OTGSC)

This register is availabe only in OTG controller core. It has four sections:

- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)
- OTG Status inputs (Read Only)
- OTG Controls (Read/Write)

The status inputs are debounced using a 1 ms time constant. Values on the status inputs that do not persist for more than 1 ms does not cause an update of the status input register, or cause an OTG interrupt.

See also USB Device Mode (USB_USBMODE) register.

Address: USB_OTGSC is 53F8_0000h base + 1A4h offset = 53F8_01A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	-	DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	-	DPIS	1msS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-	DPS	1msT	BSE	BSV	ASV	AVV	ID	-		IDPU	DP	OT	-	VC	VD
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

USB_OTGSC field descriptions

Field	Description
31 -	Reserved
30 DPIE	Data Pulse Interrupt Enable
29 1msE	1 millisecond timer Interrupt Enable - Read/Write
28 BSEIE	B Session End Interrupt Enable - Read/Write. Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable - Read/Write. Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable - Read/Write. Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable - Read/Write. Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable - Read/Write. Setting this bit enables the USB ID interrupt.
23 -	Reserved
22 DPIS	Data Pulse Interrupt Status - Read/Write to Clear. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC1(0)[PP] = 0. Software must write a one to clear this bit.
21 1msS	1 millisecond timer Interrupt Status - Read/Write to Clear. This bit is set once every millisecond. Software must write a one to clear this bit.
20 BSEIS	B Session End Interrupt Status - Read/Write to Clear. This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit
19 BSVIS	B Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a one to clear this bit.
18 ASVIS	A Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a one to clear this bit.
17 AVVIS	A VBus Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a one to clear this bit.

Table continues on the next page...

USB_OTGSC field descriptions (continued)

Field	Description
16 IDIS	USB ID Interrupt Status - Read/Write. This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.
15 -	Reserved
14 DPS	Data Bus Pulsing Status - Read Only. A '1' indicates data bus pulsing is being detected on the port.
13 1msT	1 millisecond timer toggle - Read Only. This bit toggles once per millisecond.
12 BSE	B Session End - Read Only. Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid - Read Only. Indicates VBus is above the B session valid threshold.
10 ASV	A Session Valid - Read Only. Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid - Read Only. Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID - Read Only. 0 = A device, 1 = B device
7–6 -	Reserved
5 IDPU	ID Pullup - Read/Write This bit provide control over the ID pull-up resister; 0 = off, 1 = on [default]. When this bit is 0, the ID input will not be sampled.
4 DP	Data Pulsing - Read/Write. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination - Read/Write. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2 -	Reserved
1 VC	VBUS Charge - Read/Write. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS_Discharge - Read/Write. Setting this bit causes VBus to discharge through a resistor.

56.6.32 USB Device Mode (USB_USBMODE)

Address: USB_USBMODE is 53F8_0000h base + 1A8h offset = 53F8_01A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																SRT	-										SDIS	SLOM	ES	CM	
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is implementation dependent.

USB_USBMODE field descriptions

Field	Description
31–16 -	Reserved
15 SRT	Shorten Reset Time - RW. Default = 0b. When the Controller is in host mode, this bit enables a bypass of the Chirp J/K reset handshake, saving 6-7ms in simulation time for each reset sequence. This bit should only be used for initial system integration simulations, and should always be set to 0 for normal operation.
14–5 -	Reserved
4 SDIS	Stream Disable Mode. (0 - Inactive [default]; 1 - Active) Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. **NOTE:** In High Speed Mode, all packets received are responded to with a NYET handshake when stream disable is active. Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. **NOTE:** Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TX FIFO Fill Tuning (USB_TXFILLTUNING) and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature. **NOTE:** The use of this feature substantially limits of the overall USB performance that can be achieved.
3 SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model . 0 Setup Lockouts On (default); 1 Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USB Command Register (USB_USBCMD) .
2 ES	Endian Select - Read/Write. This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word.

Table continues on the next page...

USB_USBMODE field descriptions (continued)

Field	Description
	Bit Meaning 0 Little Endian [Default] 1 Big Endian
1–0 CM	Controller Mode - R/WO. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller defaults to an idle state and needs to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the *RESET* bit in the USBCMD register before reprogramming this register. For OTG controller core, reset value is '00b'. For Host1/Host2/Host3 controller core, reset value is '11b'. Bit Meaning 00 Idle [Default for combination host/device] 01 Reserved 10 Device Controller [Default for device only controller] 11 Host Controller [Default for host only controller]

56.6.33 Endpoint Setup Status (USB_ENDPTSETUPSTAT)

Address: USB_ENDPTSETUPSTAT is 53F8_0000h base + 1ACh offset = 53F8_01ACh

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

USB_ENDPTSETUPSTAT field descriptions

Field	Description
31–16 -	Reserved
15–0 ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. See Managing Endpoints in the Device Operational Model. This register is only used in device mode.

56.6.34 Endpoint Initialization (USB_ENDPTPRIME)

This register is only used in device mode.

Address: USB_ENDPTPRIME is 53F8_0000h base + 1B0h offset = 53F8_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PETB[15:0]																PERB[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_ENDPTPRIME field descriptions

Field	Description
31–16 PETB[15:0]	Prime Endpoint Transmit Buffer - R/WS. For each endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[15] - Endpoint #15 PETB[1] - Endpoint #1 PETB[0] - Endpoint #0
15–0 PERB[15:0]	Prime Endpoint Receive Buffer - R/WS. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. Bit 15 - Endpoint #15 Bit 1 - Endpoint #1 Bit 0 - Endpoint #0

56.6.35 Endpoint De-Initialize (USB_ENDPTFLUSH)

This register is only used in device mode.

Address: USB_ENDPTFLUSH is 53F8_0000h base + 1B4h offset = 53F8_01B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FETB[15:0]																FERB[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

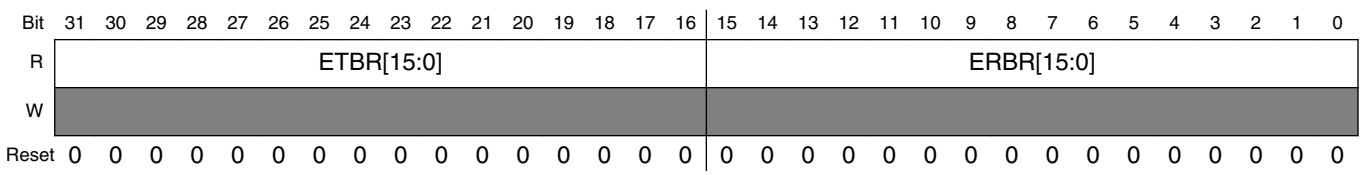
USB_ENDPTFLUSH field descriptions

Field	Description
31–16 FETB[15:0]	Flush Endpoint Transmit Buffer - R/WS. Writing one to a bit(s) in this register causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FETB[15] - Endpoint #15 FETB[1] - Endpoint #1 FETB[0] - Endpoint #0
15–0 FERB[15:0]	Flush Endpoint Receive Buffer - R/WS. Writing one to a bit(s) causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. Bit 15 - Endpoint #15 Bit 1 - Endpoint #1 Bit 0 - Endpoint #0

56.6.36 Endpoint Status (USB_ENDPTSTAT)

This register is only used in device mode.

Address: USB_ENDPTSTAT is 53F8_0000h base + 1B8h offset = 53F8_01B8h



USB_ENDPTSTAT field descriptions

Field	Description
31–16 ETBR[15:0]	Endpoint Transmit Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There is always a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[15] - Endpoint #15 ETBR[1] - Endpoint #1 ETBR[0] - Endpoint #0
15–0 ERBR[15:0]	Endpoint Receive Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPRIME register. There is always a delay between setting a bit in the ENDPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[15] - Endpoint #15 ERBR[1] - Endpoint #1 ERBR[0] - Endpoint #0

56.6.37 Endpoint Complete (USB_ENDPTCOMPLETE)

This register is only used in device mode.

Address: USB_ENDPTCOMPLETE is 53F8_0000h base + 1BCh offset = 53F8_01BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ETCE[15:0]																ERCE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_ENDPTCOMPLETE field descriptions

Field	Description
31–16 ETCE[15:0]	Endpoint Transmit Complete Event - R/WC. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ETCE[15] - Endpoint #15 ETCE[1] - Endpoint #1 ETCE[0] - Endpoint #0
15–0 ERCE[15:0]	Endpoint Receive Complete Event - RW/C. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ERCE[15] - Endpoint #15 ERCE[1] - Endpoint #1 ERCE[0] - Endpoint #0

56.6.38 Endpoint Control0 (USB_ENDPTCTRL0)

Every Device implements Endpoint 0 as a control endpoint.

Address: USB_ENDPTCTRL0 is 53F8_0000h base + 1C0h offset = 53F8_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	-								TXE	-			TXT		-	TXS
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-								RXE	-			RXT		-	RXS
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

USB_ENDPTCTRL0 field descriptions

Field	Description
31–24 -	Reserved
23 TXE	TX Endpoint Enable 1 Enabled Endpoint0 is always enabled.
22–20 -	Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 - Control Endpoint0 is fixed as a Control End Point.
17 -	Reserved
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request.

Table continues on the next page...

USB_ENDPTCTRL0 field descriptions (continued)

Field	Description
15–8 -	Reserved
7 RXE	RX Endpoint Enable 1 Enabled Endpoint0 is always enabled.
6–4 -	Reserved
3–2 RXT	RX Endpoint Type - Read/Write 00 Control Endpoint0 is fixed as a Control End Point.
1 -	Reserved
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request.

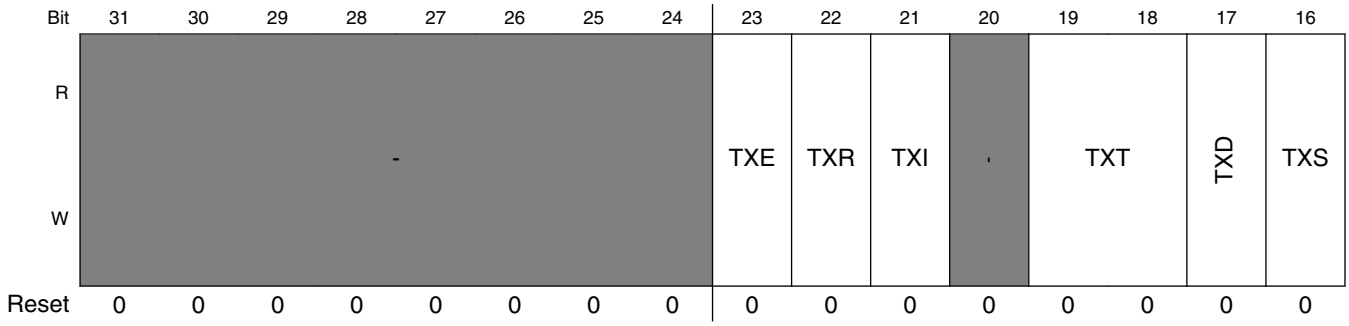
56.6.39 Endpoint ControlIn (USB_ENDPTCTRL*n*)

There is a ENDPTCTRLx register for each endpoint in a device.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Addresses: 53F8_0000h base + 1C4h offset + (4d × *n*), where *n* = 0d to 14d



USB_ENDPTCTRL*n* field descriptions

Field	Description
31–24 -	Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk

Table continues on the next page...

USB_ENDPTCTRL*n* field descriptions (continued)

Field	Description
	11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues to returning STALL until this bit is either cleared by software or automatically cleared as above.
15–8 -	Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Reserved
1 RXD	RX Endpoint Data Sink - Read/Write - TBD 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default]

Table continues on the next page...

USB_ENDPTCTRL*n* field descriptions (continued)

Field	Description
	1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It is cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues to returning STALL until this bit is either cleared by software or automatically cleared as above.

Chapter 57
Watchdog Timer (WDOG-1)

57.1 Introduction

This chapter describes a block integrated into an SoC. The chapter is intended for a block driver software developer. It describes block-level operation and programming. To understand how the block is integrated at the SoC level, a system software developer should see discussions of the block in the appropriate SoC-level chapter(s).

57.2 Overview

This section briefly introduces the block. The full description of the block is in Functional Description.

This section includes a top level diagram that shows the functional organization of the block, including all off-chip signals. The figure below is the block diagram.

The Watchdog Timer (WDOG-1) protects against system failures by providing a method of escaping from unexpected events or programming errors. Once the WDOG-1 is activated, it must be serviced by the software on a periodic basis. If servicing does not take place, the timer times out. Upon a time-out, the WDOG-1 asserts the internal system reset signal, $\overline{\text{wdog\_rst}}$ which will be going to System Reset Controller . There is also a provision for WDOG-1 signal assertion by time-out counter expiration.There is an option of programmable interrupt generation before the counter actually times out. The time at which the interrupt needs to be generated prior to counter time-out is programmable. There is a power down counter which gets enabled out of any reset (POR, Warm / Cold). This counter has a fixed time out period of 16 seconds upon which it will assert the WDOG-1signal. Flow diagrams for the time-out counter, power down counter and interrupt operations are shown in Figure 57-6, Figure 57-7 and Figure 57-8.

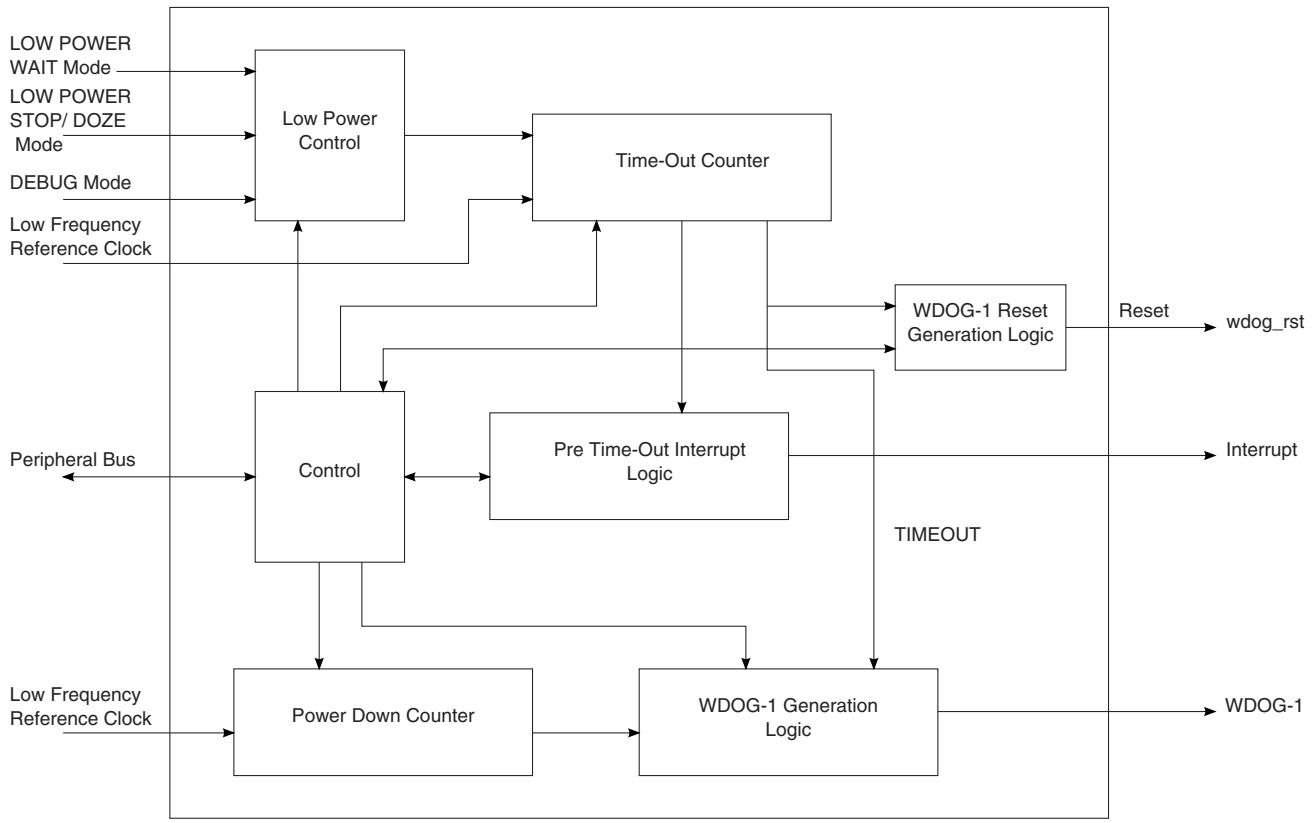


Figure 57-1. WDOG-1 Diagram

57.2.1 Features

The WDOG-1 features are as follows:

- A Configurable time-out counter with Time-out periods from 0.5 seconds up to 128 seconds and after time-out expiration results in assertion of $\overline{\text{wdog\_rst}}$ Reset signal .
- Time resolution of 0.5 seconds.
- Configurable time-out counter that can be programmed to run or stop during low-power modes.
- Configurable time-out counter that can be programmed to run or stop during DEBUG mode.
- Programmable Interrupt generation prior to time-out.
- The time duration between interrupt and time-out events can be programmed from 0 to 127.5 seconds in steps of 0.5 seconds.
- A power down counter with fixed time-out period of 16 seconds which if not disabled after reset will assert $\overline{\text{WDOG-1}}$ signal low.
    - Power down counter will be enabled out of any reset (POR, Warm / Cold reset) by default.

57.2.2 Modes and Operations

The WDOG-1 supports the following modes described in the indicated sections:

- Low Power Modes
- Debug Mode

As described in Operations, the WDOG supports the operations described in the indicated sections:

- Watchdog Reset Generation
- WDOG_B Generation

57.3 External Signals

Table 57-1. Off-Chip Block Signals

Signal	I/O	Description	Reset State[1]	Pull-Up/Down[1]
WDOG-1	O	This signal will powerdown the Chip. Refer to WDOG_B Generation.	1	-
wdog_rst	O	This signal is a reset source for the chip. Refer to Watchdog Reset Generation.	1	-

1. The reset state values and pull-up/down requirements provided in this table are from the block-level perspective. To understand how the block is integrated at the SoC level, the system software developer must see discussions of the block in the appropriate SoC-level chapter(s). For example, a block signal that requires a pull-up could be integrated with a pull-up option built into the SoC. In this case, the system software developer must ensure the proper programming at the SoC level.

57.4 Functional Description

57.4.1 Time-Out Event

The WDOG-1 provides time-out periods from 0.5 seconds up to 128 seconds with a time resolution of 0.5 seconds.

The user can determine the time-out period by writing to the WDOG-1 Time-out field (WT[7:0]) in the Watchdog Control Register (WDOG_WCR). The WDOG-1 has to be enabled by setting the WDE bit of Watchdog Control Register (WDOG_WCR) for the time-out counter to start running. After the WDOG-1 is enabled, the counter is activated, loads the time-out value and begins to count down from this programmed value. The

timer will time-out when the counter reaches zero and the WDOG-1 outputs a system reset signal, wdog_rst and asserts WDOG-1 (WDT bit should be set in Watchdog Control Register (WDOG_WCR)).

However, the Time-out condition can be prevented by reloading the counter with the new time-out value (WT[7:0] of WDOG-1.WCR) if a service routine (see Servicing WDOG-1 To Reload The Counter) is performed before the counter reaches zero. If any system errors occur that prevent the software from servicing the Watchdog Service Register (WDOG_WSR), then the time-out condition occurs. By performing the service routine, the WDOG-1 reloads its counter to the time-out value indicated by bits WT[7:0] of the Watchdog Control Register (WDOG_WCR) and it restarts the countdown.

A system reset will reset the counter and place it in the idle state at any time during the countdown. The counter flow diagram is shown in Figure 57-6.

NOTE

The time-out value is reloaded to the counter at the time when WDOG-1 is enabled or after the service routine has been performed.

57.4.1.1 Servicing WDOG-1 To Reload The Counter

The proper service sequence to reload a time-out value to the counter begins by writing 0x 5555 followed by 0x AAAA to the Watchdog Service Register (WDOG_WSR). Any number of instructions can be executed between the two writes. If the WDOG-1.WSR is not loaded with 0x 5555 prior to writing 0x AAAA to the WDOG-1.WSR, the counter is not reloaded. If any value other than 0x AAAA is written to the WDOG-1.WSR after 0x 5555, the counter is not reloaded. This service sequence will reload the counter with the time-out value WT[7:0] of Watchdog Control Register (WDOG_WCR). The time-out value can be changed at any point of the time and the counter reloads this value whenever the core services the Watchdog.

57.4.2 Interrupt Event

Prior to time-out the WDOG-1 can generate an interrupt which can be considered a warning signal to indicate that the time-out will occur shortly.

The time duration between interrupt event and time-out event can be controlled by writing to the WICT field of Watchdog Interrupt Control Register (WDOG_WICR). It can vary between 0 and 127.5 seconds. If the WDOG-1 is serviced (Servicing WDOG-1

To Reload The Counter) before the interrupt generation then the counter will be reloaded with the time-out value WT[7:0] of Watchdog Control Register (WDOG_WCR) and interrupt will not be triggered.

57.4.3 Power-Down Counter Event

The Power down Counter inside WDOG-1 will be enabled out of reset. This counter has a fixed time-out value of 16 seconds, after which it will drive the $\overline{\text{WDOG-1}}$ signal low.

To prevent this, the software must disable this counter by clearing the PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) within 16 seconds of reset de-assertion. Once disabled, this counter cannot be enabled again until the next system reset occurs. This feature is provided to prevent the hanging up of cores after reset, as WDOG-1 is not enabled out of reset.

57.4.4 Low Power Modes

57.4.4.1 STOP and DOZE Mode

If the WDOG-1 Timer Disable bit for Low power STOP and DOZE mode (WDZST) bit in the Watchdog Control Register (WDOG_WCR), is '0", the WDOG-1 Timer continues to operate using the Low Frequency Reference clock. If Low power Enable (WDZST) bit is set to "1", then the WDOG-1 Timer operation will be suspended in Low power STOP or DOZE mode. Upon exiting Low power STOP or DOZE mode, the WDOG-1 operation returns to what it was prior to entering the STOP or DOZE mode.

57.4.4.2 WAIT Mode

If the WDOG-1 Timer Disable bit for low power WAIT mode (WDW) bit in the Watchdog Control Register (WDOG_WCR), is '0", the WDOG-1 Timer continues to operate using the Low Frequency Reference clock . If low power WAIT Enable (WDW) bit is set to "1", then the WDOG-1 Timer operation will be suspended. Upon exiting low power WAIT mode, the WDOG-1 operation returns to what it was prior to entering the WAIT mode.

NOTE

WDOG-1 Timer will not be able to detect events which happen for periods less than one Low Frequency Reference clock cycle. For example in repeated WAIT mode entry/exit, if the RUN

mode time is less than one Low Frequency Reference clock cycle and if WDW bit is set then WDOG-1 Timer may never time out even though the system is in RUN mode for a finite duration because WDOG-1 may not see any Low Frequency Reference clock edge during its wake time.

57.4.5 Debug Mode

The WDOG-1 Timer can be configured for continual operation, or the operation can be suspended during debug mode. If the WDOG-1 debug enable (WDBG) bit is set to "1" in the Watchdog Control Register (WDOG_WCR), the WDOG-1 Timer operation is suspended in debug mode. In-case of WDBG bit being set to "1" and the Debug mode is entered, WDOG-1 Timer operation is suspended after 2 Low Frequency Reference clocks and similarly WDOG-1 Timer operation is continued after 2 Low frequency reference clocks of Debug mode Exit. Register read and write accesses in Debug mode continue to function normally. Also, while in DEBUG mode, the WDE bit of Watchdog Control Register (WDOG_WCR) can be enabled-disabled directly. If the WDOG-1 debug enable (WDBG) bit is cleared then Watchdog Timer operation is not suspended. Power-down counter is not affected by debug mode entry/exit.

NOTE

If the WDE bit of Watchdog Control Register (WDOG_WCR) is set/cleared while in DEBUG mode, it remains set/cleared even after exiting DEBUG mode.

57.4.6 Operations

This section describes the block's operations.

57.4.6.1 Watchdog Reset Generation

The WDOG-1 generated reset signal $\overline{\text{wdog\_rst}}$ is asserted by the following operations:

- A software write to the Software Reset Signal (SRS) bit of the Watchdog Control Register (WDOG_WCR).
- WDOG-1 time-out. See Time-Out Event.

The $\overline{\text{wdog\_rst}}$ will be asserted for one clock cycle of Low frequency Reference clock for both the time-out Condition and software write occurrence. It remains asserted for 1 clock cycle of Low frequency Reference clock even if a system reset is asserted in between. Figure 57-3 shows the timing diagram of this signal due to time-out condition.

57.4.6.2 $\overline{\text{WDOG\_B}}$ Generation

The WDOG-1 asserts $\overline{\text{WDOG}}$ under the following scenarios:

- Software write to WDA bit of Watchdog Control Register (WDOG_WCR). $\overline{\text{WDOG}}$ Signal remains asserted as long as the WDA bit is "0".
- WDOG-1 time-out condition, WDT bit of Watchdog Control Register (WDOG_WCR) must be set for this scenario. Description of the time-out condition can be found in the Time-Out Event.$\overline{\text{WDOG-1}}$ Signal remains asserted until a Power-on Reset (POR) occurs. It gets cleared after the POR occurs and not due to any other system reset. Figure 57-4 shows the timing diagram of $\overline{\text{WDOG-1}}$ due to time-out condition.
- WDOG-1 Power down Counter time-out, PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) should not be cleared for this scenario. Description of this counter can be found in the Power-Down Counter Event.$\overline{\text{WDOG-1}}$ Signal remains asserted for one clock cycle of Low frequency Reference clock.

Figure 57-2 shows the scenarios under which $\overline{\text{WDOG-1}}$ gets asserted.

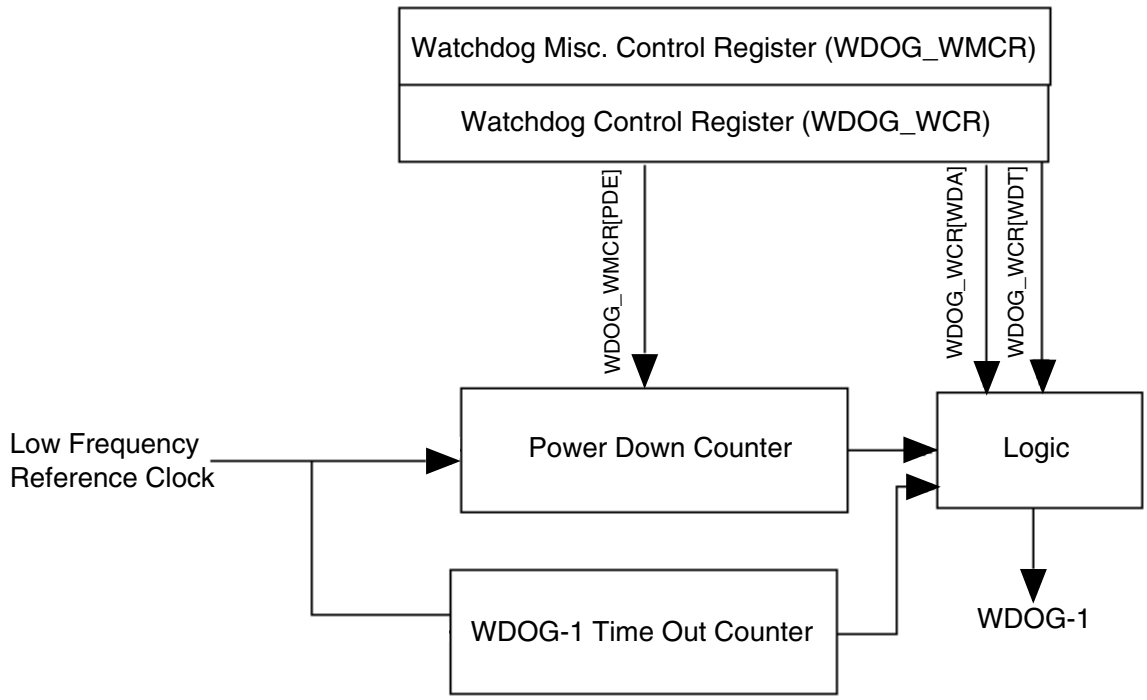


Figure 57-2. $\overline{\text{WDOG-1}}$ Generation

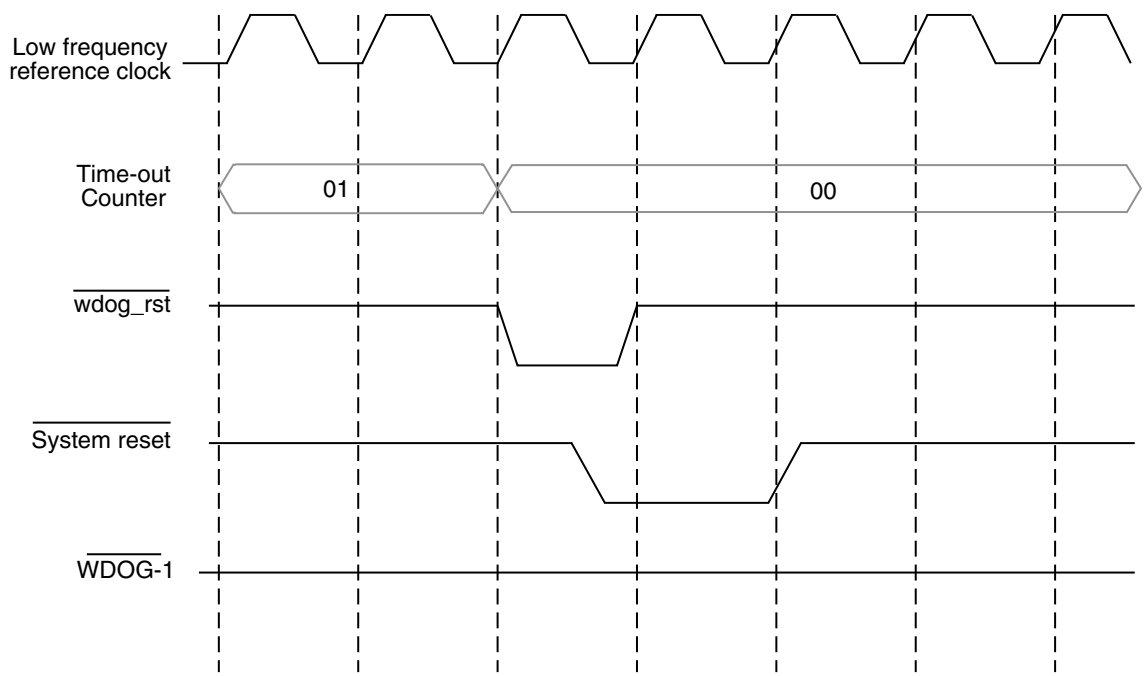


Figure 57-3. WDOG-1 Time-Out Condition/WDT Bit Is Not Set

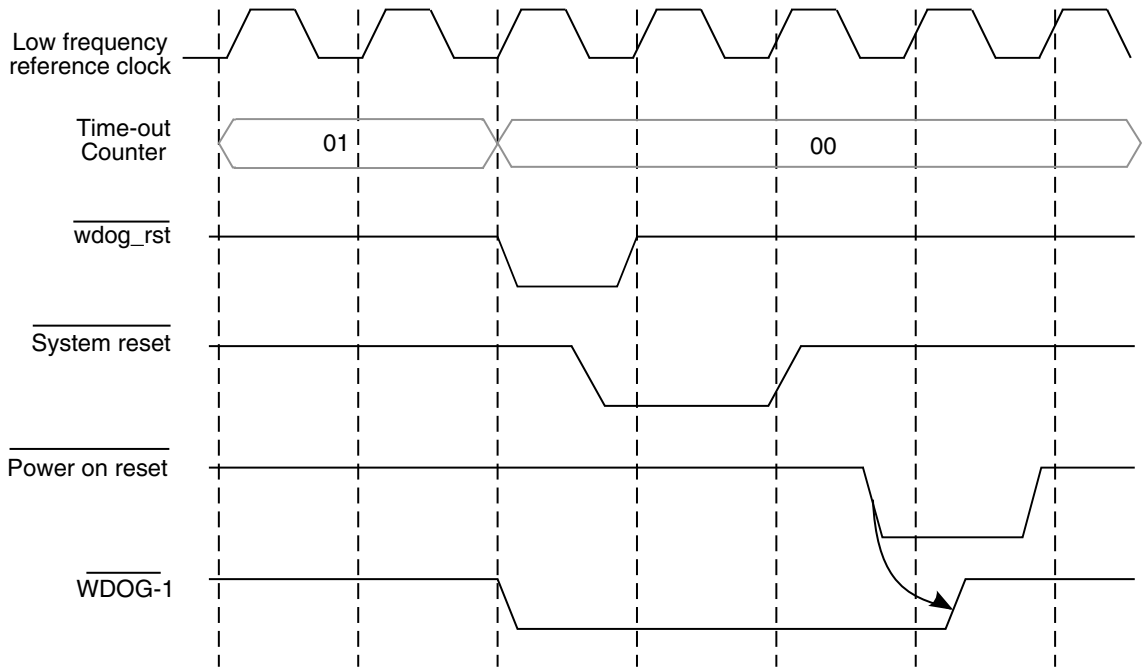


Figure 57-4. WDOG-1 Time-Out Condition/WDT Bit Is Set

57.4.7 Clocks

This section describes clocks and special clocking requirements of the block.

The WDOG-1 uses the Low frequency Reference clock for its counter and control operations. Peripheral Bus clock is used for register Read/Write operations.

Low frequency Reference clock is a free running clock and cannot be gated. Peripheral Bus clock cannot be gated and is selectively switched ON whenever Read/Write operations takes place.

57.4.8 Reset

This section describes how to reset the block and explains special requirements related to reset.

The block is reset by a system reset and will have the following consequences: WDOG-1 counter is disabled.

Power-Down counter is enabled and starts counting.

57.4.9 Interrupt

The WDOG-1 has the feature of Interrupt generation before time-out.

The interrupt will be generated only if the WIE bit in Watchdog Interrupt Control Register (WDOG_WICR) is set. The exact time at which the interrupt should occur prior to time-out depends on the value of WICT field of Watchdog Interrupt Control Register (WDOG_WICR). As an example if the WICT field has a value 0x04, then the interrupt will be generated 2 seconds prior to time-out. Once the interrupt is triggered the WTIS bit in Watchdog Interrupt Control Register (WDOG_WICR) will be set. The software need to clear this bit to de-assert the Interrupt. If the WDOG-1 is serviced before the interrupt generation then the counter will be reloaded with the time-out value WT[7:0] of Watchdog Control Register (WDOG_WCR) and interrupt would not be triggered.

57.4.10 Flow Diagrams

A flow diagram of watchdog operation is shown in Figure 57-6, Figure 57-7 and Figure 57-8.

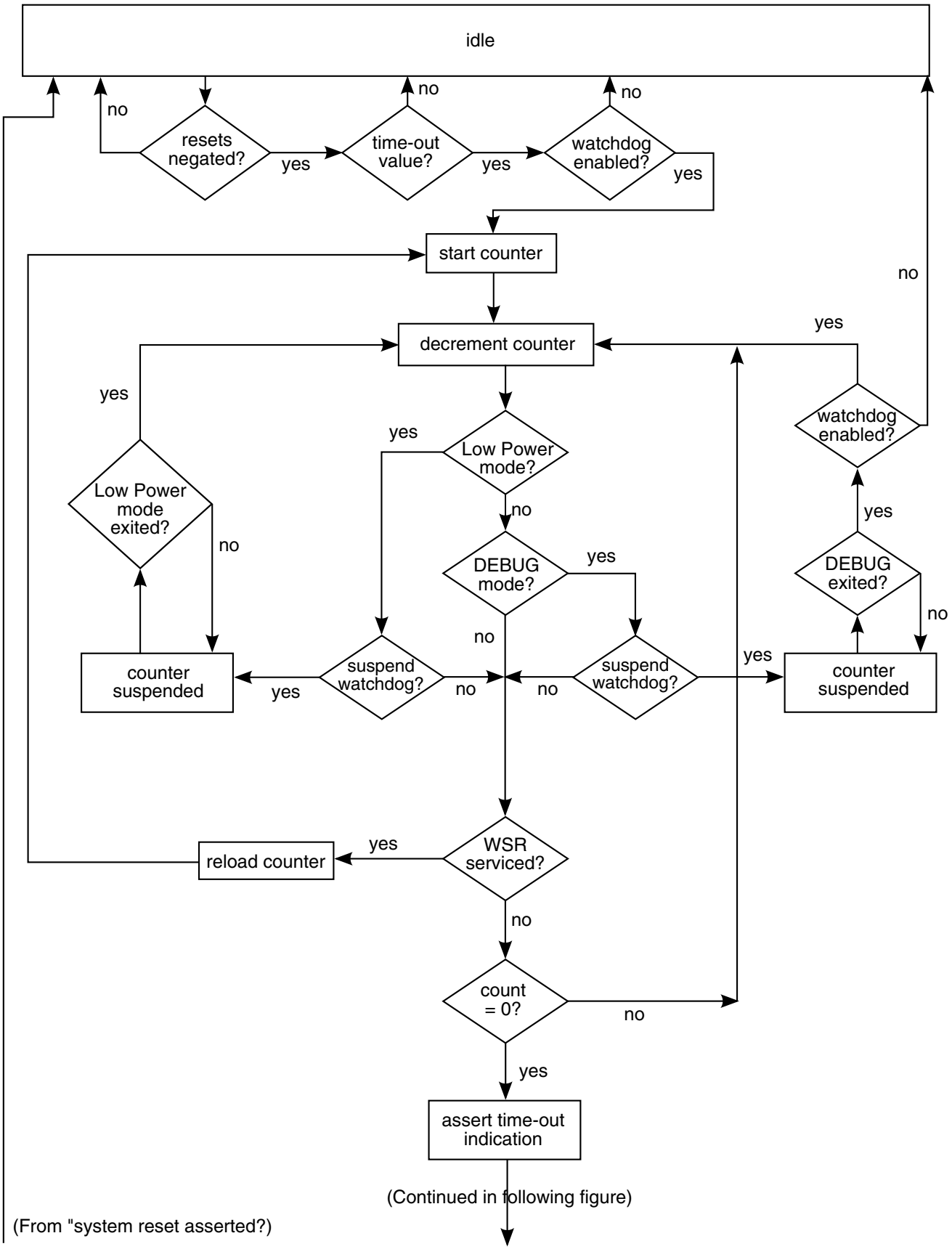
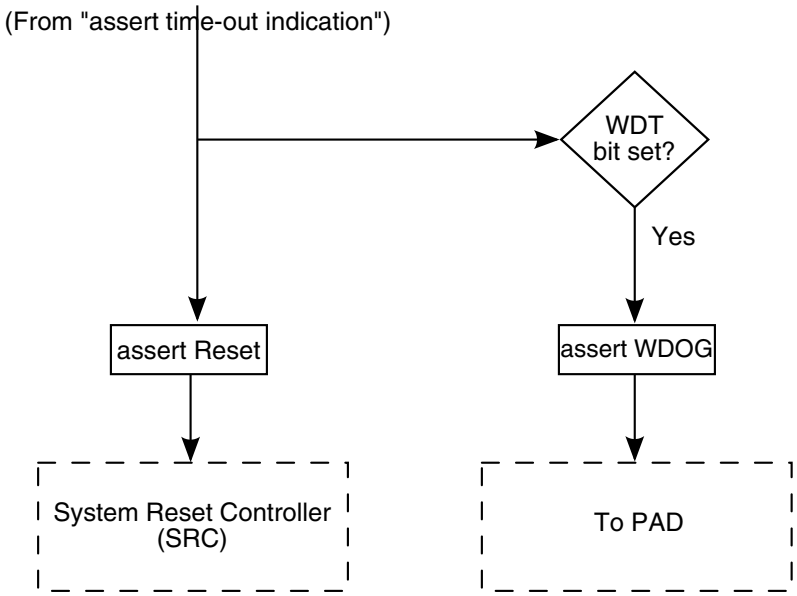


Figure 57-5. Time-Out Counter Flow Diagram



Note: A system reset will force the state machine to "idle" at any time during countdown.

Figure 57-6. Time-Out Counter Flow Diagram

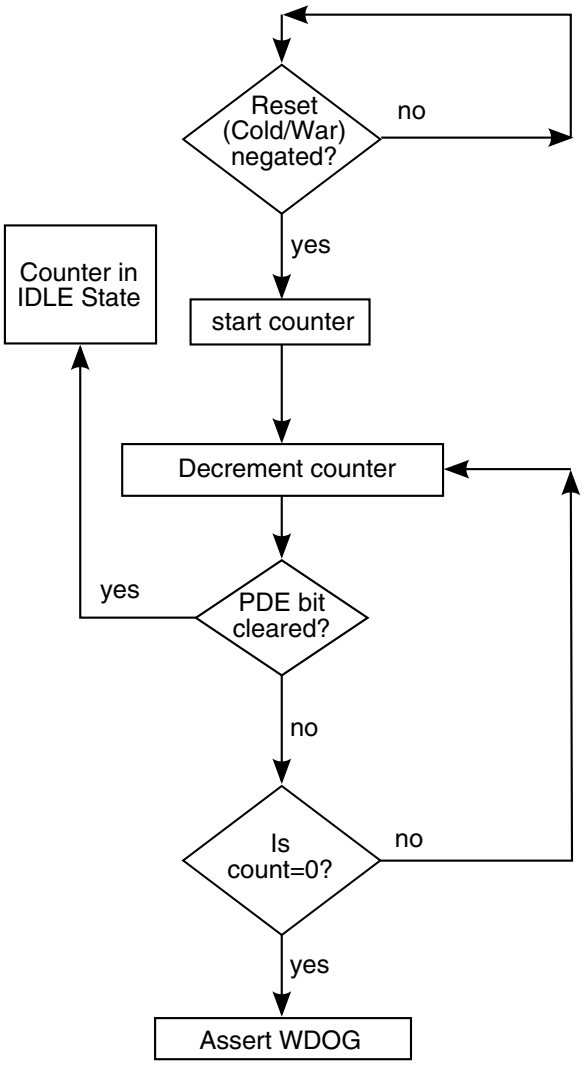


Figure 57-7. Power-Down Counter Flow Diagram

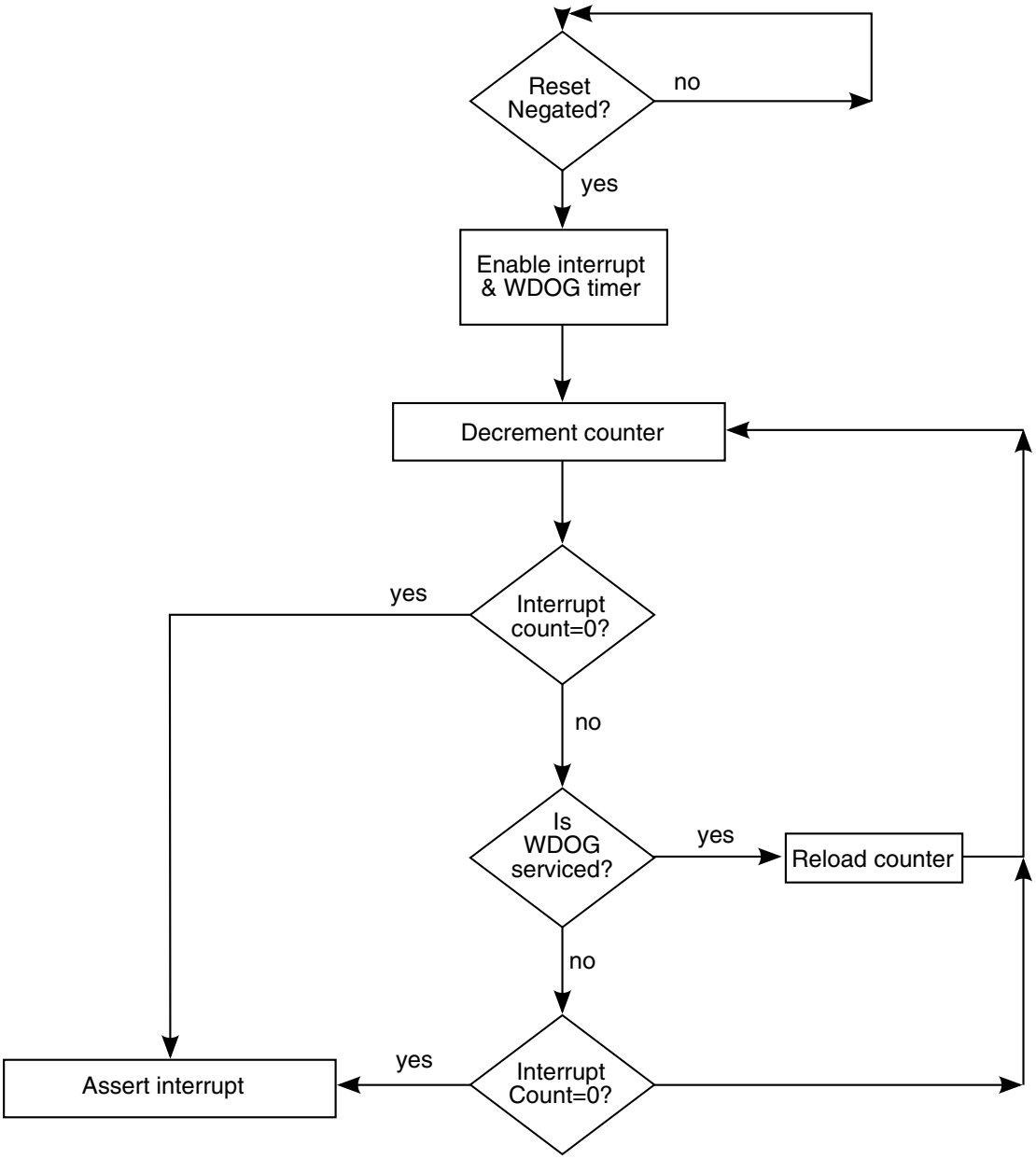


Figure 57-8. Interrupt Generation Flow Diagram

57.5 Initialization

The following sequence of programming should be performed for WDOG-1 initialization:

- PDE bit of Watchdog Miscellaneous Control Register (WDOG_WMCR) should be cleared for disabling the power down counter.

- WDT field of Watchdog Control Register (WDOG_WCR) should be programmed for sufficient time-out value.
- WDOG-1 should be enabled by setting WDE bit of Watchdog Control Register (WDOG_WCR) so that the Time-out counter loads the WDT field value of Watchdog Control Register (WDOG_WCR) and starts counting.

57.6 Programmable Registers

The WDOG-1 has user-accessible, 16-bit registers used to configure, operate, and monitor the state of the Watchdog Timer. Byte operations can be performed on these registers. If a 32-bit access is performed on these registers then the WDOG-1 will not generate any Peripheral Bus error and will behave normally, like a 16-Bit access, making Read/Write possible. A 32-Bit access should be avoided, as the system might go to an unknown state.

WDOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
53F9_8000	Watchdog Control Register (WDOG_WCR)	16	R/W	0030h	57.6.1/ 3357
53F9_8002	Watchdog Service Register (WDOG_WSR)	16	R/W	0000h	57.6.2/ 3359
53F9_8004	Watchdog Reset Status Register (WDOG_WRSR)	16	R	0000h	57.6.3/ 3360
53F9_8006	Watchdog Interrupt Control Register (WDOG_WICR)	16	R/W	0004h	57.6.4/ 3361
53F9_8008	Watchdog Miscellaneous Control Register (WDOG_WMCR)	16	R/W	0001h	57.6.5/ 3362

57.6.1 Watchdog Control Register (WDOG_WCR)

The Watchdog Control Register (WDOG_WCR) controls the WDOG-1 operation.

- WDZST, WDBG and WDW bits are write-once only bits. Once the software does a write access to these bits, all these bits will get locked and cannot be reprogrammed till the next system reset assertion.

- WDE is a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared till the next system reset.
- WDT is also a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared till the next POR (Power-on Reset). This bit does not gets reset/cleared due to any system reset.

Address: WDOG_WCR is 53F9_8000h base + 0h offset = 53F9_8000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WT								WDW	-	WDA	SRS	WDT	WDE	WDBG	WDZST
Write																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

WDOG_WCR field descriptions

Field	Description
15–8 WT	Watchdog Time-out Field. This 8-bit field contains the time-out value that is loaded into the Watchdog counter after the service routine has been performed or after the Watchdog is enabled. After reset, WT[7:0] must have a value written to it before enabling the Watchdog otherwise count value of zero which is 0.5 seconds is loaded into the counter. **NOTE:** The time-out value can be written at any point of time but it is loaded to the counter at the time when WDOG-1 is enabled or after the service routine has been performed. For more information see Time-Out Event . 0x00 - 0.5 Seconds (Default). 0x01 - 1.0 Seconds. 0x02 - 1.5 Seconds. 0x03 - 2.0 Seconds. 0xff - 128 Seconds.
7 WDW	Watchdog Disable for Wait. This bit determines the operation of WDOG-1 during Low Power WAIT mode. This is a write once only bit. 0 Continue WDOG-1 timer operation (Default). 1 Suspend WDOG-1 timer operation.
6 -	Reserved adopt Reserved
5 WDA	$\overline{\text{WDOG}}$ assertion. Controls the software assertion of the $\overline{\text{WDOG}}$ signal. 0 Assert $\overline{\text{WDOG}}$ output. 1 No effect on system (Default).
4 SRS	Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal $\overline{\text{wdog\_rst}}$. This bit automatically resets to "1" after it has been asserted to "0". **NOTE:** This bit does not generate the software reset to the block.

Table continues on the next page...

WDOG_WCR field descriptions (continued)

Field	Description
	0 Assert system reset signal. 1 No effect on the system (Default).
3 WDT	$\overline{\text{WDOG}}$ Time-out assertion. Determines if the $\overline{\text{WDOG}}$ gets asserted upon a Watchdog Time-out Event. This is a write-one once only bit. **NOTE:** There is no effect on $\overline{\text{wdog\_rst}}$ (WDOG Reset) upon writing on this bit. $\overline{\text{WDOG}}$ gets asserted along with $\overline{\text{wdog\_rst}}$ if this bit is set. 0 No effect on $\overline{\text{WDOG}}$ (Default). 1 Assert $\overline{\text{WDOG}}$ upon a Watchdog Time-out event.
2 WDE	Watchdog Enable. Enables or disables the WDOG-1 block. This is a write one once only bit. It is not possible to clear this bit by a software write, once the bit is set. **NOTE:** This bit can be set/reset in debug mode (exception). 0 Disable the Watchdog (Default). 1 Enable the Watchdog.
1 WDBG	Watchdog DEBUG Enable. Determines the operation of the WDOG-1 during DEBUG mode. This bit is write once-only. 0 Continue WDOG-1 timer operation (Default). 1 Suspend the watchdog timer.
0 WDZST	Watchdog Low Power. Determines the operation of the WDOG-1 during low-power modes. This bit is write once-only. **NOTE:** The WDOG-1 can continue/suspend the timer operation in the low-power modes (STOP and DOZE mode). 0 Continue timer operation (Default). 1 Suspend the watchdog timer.

57.6.2 Watchdog Service Register (WDOG_WSR)

When enabled, the WDOG-1 requires that a service sequence be written to the Watchdog Service Register (WSR) to prevent the time-out condition.

NOTE

Executing the service sequence will reload the WDOG-1 time out counter.

Address: WDOG_WSR is 53F9_8000h base + 2h offset = 53F9_8002h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	WSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WDOG_WSR field descriptions

Field	Description
15–0 WSR	Watchdog Service Register. This 16-bit field contains the Watchdog service sequence. Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes. The service sequence must be performed as follows: 0x5555 Write to the Watchdog Service Register (WDOG_WSR). 0xAAAA Write to the Watchdog Service Register (WDOG_WSR).

57.6.3 Watchdog Reset Status Register (WDOG_WRSR)

The WRSR is a read-only register that records the source of the output reset assertion. It is not cleared by a hard reset. Therefore, only one bit in the WRSR will always be asserted high. The register will always indicate the source of the last reset generated due to WDOG-1. Read access to this register is with one wait state. Any write performed on this register will generate a Peripheral Bus Error .

A reset can be generated by the following sources, as listed in priority from highest to lowest:

- Power On Reset
- Watchdog Time-out
- Software Reset

Address: WDOG_WRSR is 53F9_8000h base + 4h offset = 53F9_8004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0											POR	0		TOUT	SFTW
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WDOG_WRSR field descriptions

Field	Description
15–5 Reserved	This read-only field is reserved and always has the value zero. Reserved.
4 POR	Power On Reset. Indicates whether the reset is the result of a power on reset. 0 Reset is not the result of a power on reset. 1 Reset is the result of a power on reset.
3–2 Reserved	This read-only field is reserved and always has the value zero. Reserved.
1 TOUT	Time-out. Indicates whether the reset is the result of a WDOG-1 time-out.

Table continues on the next page...

WDOG_WRSR field descriptions (continued)

Field	Description
	0 Reset is not the result of a WDOG-1 time-out. 1 Reset is the result of a WDOG-1 time-out.
0 SFTW	Software Reset. Indicates whether the reset is the result of a WDOG-1 software reset by asserting SRS bit 0 Reset is not the result of a software reset. 1 Reset is the result of a software reset.

57.6.4 Watchdog Interrupt Control Register (WDOG_WICR)

The WDOG_WICR controls the WDOG-1 interrupt generation.

Address: WDOG_WICR is 53F9_8000h base + 6h offset = 53F9_8006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	WIE	WTIS	0						WICT							
Write		w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

WDOG_WICR field descriptions

Field	Description
15 WIE	Watchdog Timer Interrupt enable bit. Reset value is 0. **NOTE:** This bit is a write once only bit. Once the software does a write access to this bit, it will get locked and cannot be reprogrammed till the next system reset assertion 0 Disable Interrupt (Default). 1 Enable Interrupt.
14 WTIS	Watchdog TImer Interrupt Status bit will reflect the timer interrupt status, whether interrupt has occurred or not.Once the interrupt has been triggered software must clear this bit by writing 1 to it. 0 No interrupt has occurred (Default). 1 Interrupt has occurred
13–8 Reserved	This read-only field is reserved and always has the value zero. Reserved.
7–0 WICT	Watchdog Interrupt Count Time-out (WICT) field determines, how long before the counter time-out must the interrupt occur. The reset value is 0x04 implies interrupt will occur 2 seconds before time-out. The maximum value that can be programmed to WICT field is 127.5 seconds with a resolution of 0.5 seconds. **NOTE:** This field is write once only. Once the software does a write access to this field, it will get locked and cannot be reprogrammed till the next system reset assertion. 0x00 WICT[7:0] = Time duration between interrupt and time-out is 0 seconds. 0x01 WICT[7:0] = Time duration between interrupt and time-out is 0.5 seconds. 0x04 WICT[7:0] = Time duration between interrupt and time-out is 2 seconds (Default). 0xff WICT[7:0] = Time duration between interrupt and time-out is 127.5 seconds.

57.6.5 Watchdog Miscellaneous Control Register (WDOG_WMCR)

WDOG_WMCR Controls the Power Down counter operation.

Address: WDOG_WMCR is 53F9_8000h base + 8h offset = 53F9_8008h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0															PDE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

WDOG_WMCR field descriptions

Field	Description
15–1 Reserved	This read-only field is reserved and always has the value zero. Reserved.
0 PDE	Power Down Enable bit. Reset value of this bit is 1, which means the power down counter inside the WDOG-1 is enabled after reset. The software must write 0 to this bit to disable the counter within 16 seconds of reset de-assertion. Once disabled this counter cannot be enabled again. See Power-Down Counter Event for operation of this counter. **NOTE:** This bit is write-one once only bit. Once software sets this bit it cannot be reset till the next system reset. 0 Power Down Counter of WDOG-1 is disabled. 1 Power Down Counter of WDOG-1 is enabled (Default).

Chapter 58
Crystal Oscillator (XTALOSC)

58.1 Introduction

This is a Pierce oscillator with Automatic Level Controller (ALC), that detects the amplitude of the oscillation on “extal” and adjust the amount of current sourced to the crystal accordingly. As the oscillator is initially turned on and there’s no oscillation yet, the oscillator will source its maximum current of a few milliamps to kick start the crystal. Once the oscillation starts and the amplitude of oscillation grows and is detected and monitored by the ALC circuitry, the oscillator slowly cuts back its current to a stable value of a few hundreds microamps.

58.1.1 Interface Specification

Most signals in the xtalosc interface are either analog in nature or clock signals with no specific timing. They are represented in Figure 58-1 and Table 58-1.

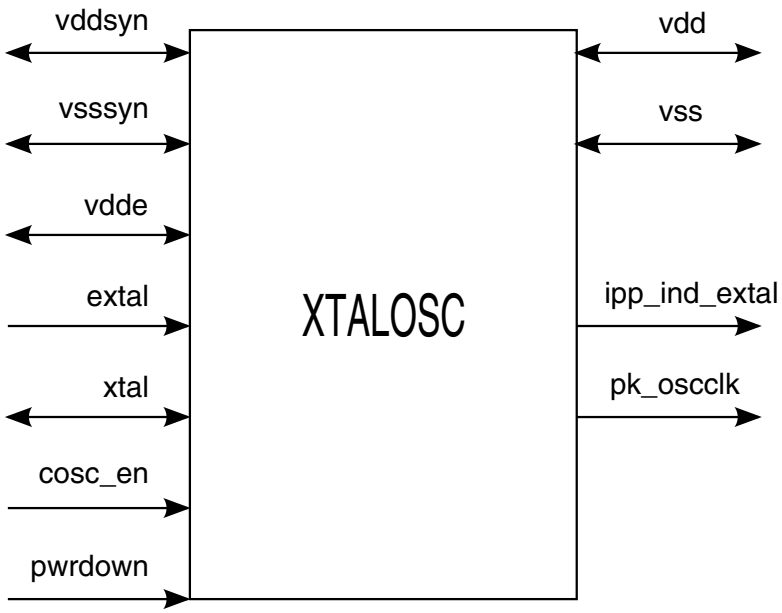


Figure 58-1. XTALOSC Interface

Table 58-1. XTALOSC Interface Signals

Port Name	Direction	Voltage Level	Purpose
cosc_en	I	either 2.5V or core logic voltage	enable crystal osc mode, enable bypass mode when deasserted
extal	I	same as clock supply voltage in bypass mode	clock input in crystal or bypass mode
pwrdown	I	either 2.5V or core logic voltage	enable low power mode
xtal	I/O	same as clock supply voltage in bypass mode	clock input in crystal mode
vddsyn	I/O	2.5V	osc power supply
vsssyn	I/O	analog ground	osc ground
vdd	I/O	1.2V	core vdd
vss	I/O	digital ground	digital ground
vdde	I/O	clock supply voltage	clock power supply
pk_oscclk	O	2.5V	2.5V osc clock for 2.5V PLL reference clock
ipp_ind_extal	O	core logic voltage	1.2V osc clock for 1.2V pll reference clock or 1.2V bypass clock

58.1.2 Crystal Operating Frequency

This oscillator uses crystals with fundamental frequency 24 MHz

58.1.3 Power Supply

This oscillator is powered by a 2.5 V power supply ±10% tolerance. The core supply is 1.2 V ±10% tolerance.

58.1.4 Input Clock in Bypass Mode

Input clock voltage range in bypass mode is from 1.8 V to 2.5 V (50% duty cycle). It is recommended that the input clock voltage is same as Vdde for proper output clock duty cycle.

58.1.5 Input Control Signal

To prevent voltage mismatch problem, input control voltage range can be high (2.5 V) or low voltage (1.2 V for cmos065lp)

58.1.6 Output Clock

The output clock that are the same as the 2.5 V power supply are pk_oscclk,. The output clock that are the same as the core logic voltage are ipp_ind_extal. All output clocks, either 2.5 V or 1.2 V, are required to have duty cycle in the range of 45% to 55% for 50% clock inputs

58.2 Operation Modes

58.2.1 Crystal Osc Mode

In this mode, both extal and xtal are connected to a crystal and “cosc_en” is asserted with “pwrdown” deasserted. Two output clocks are available: “pk_oscclk” at 2.5 V for 2.5 V PLL and ipp_ind_extal at 1.2V for 1.2V PLL.

58.2.2 Bypass Mode

In this mode, the crystal oscillator amplifier is off with “cosc_en” deasserted. Only “extal” can be driven by an external clock source. Any clock at voltage level of 1.2 V to 2.5 V is level-shifted to 2.5 V for 2.5 V PLL or 1.2 V for 1.2 V logic. The level shifters are actually comparators.

58.2.3 Low Power Mode

The “pwrdown” signal turns most of the circuitry in the osc, including the bias circuit.

58.2.4 Oscillator Safety Margin Requirements

Oscillator safety margin requirements are normally 5 or 10 times the maximum crystal ESR, depending who the crystal vendor is. NDK requires 5x and KDS requires 10x. Oscillator safety margin depends heavily on the capacitive loads on “extal” and “xtal” pins so care must be taken when doing simulation to include parasitic loads on chip as well as parasitic caps on boards.

Appendix A
SDMA Scripts

A.1 Introduction

This appendix provides descriptions of scripts that may be used to perform data transfers using the Smart DMA (SDMA) block of the SoC. The SDMA block supports data transfer from core memory space to core memory, from core memory space to peripherals, and vice versa.

A.2 SDMA Scripts Overview

Table A-1 gives an overview of the SDMA scripts.

Table A-1. SDMA Scripts Overview

Data Transfer	Script to Use	Use Case	Parameters Through Context	Parameters Through Buffer Descriptor
Memory	ap_2_ap	Memory Copy	None	Word length (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter Fist address is memory source address, second address is memory destination address
Shared UART	uartsh_2_mcu	Uart Rx	UART Rx FIFO address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01), Counter Mem destination address, Second address not used
UART	uart_2_mcu	Uart Rx	Uart Rx FIFO address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01), Counter Memory source address, Second address not used
SSI	mcu_2_app, mcu_2_shp	Audio Playback	SSI Tx FIFO address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter Memory source address Second address not used

Table continues on the next page...

Table A-1. SDMA Scripts Overview (continued)

Data Transfer	Script to Use	Use Case	Parameters Through Context	Parameters Through Buffer Descriptor
	app_2_mcu, shp_2_mcu	Audio Record	SSI Rx FIFO address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory source address Second address not used
	mcu_2_ssish	Audio Playback	SSI Tx FIFO 0 address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter Memory source address, Second address not used
	ssish_2_mcu	Audio Record	SSI Rx FIFO 0 address (r6) Event_mask (r1) Event2_mask (r0) Watermark level (r7)	Periph size (2'b01 or 2'b10 or 2'b11 or 2'b00), Counter, Memory source address Second address not used

In Table A-1, the data transfer column lists the possible types of DMA channels that involve the SDMA and a specific script is attached to each channel. It must be noted that some scripts cover several DMA channels. The scripts deal with the channels that have generic peripherals where only the watermark level is needed (no aging timer, no end_of_packet feature, and so on). The location refers to whether a script resides in ROM or in RAM and the size of the script is given in the instructions. The parameters columns are explained in the next section.

The following terms are used in Table A-1 which lists all the supported data transfers.

EMI or External Memory:

The external memories are connected to the External Memory Interface. Thus, a data transfer to EMI means a data transfer to any of the external memories (NAND Flash, NOR Flash, PSRAM, SDRAM, and so on).

Host mem or ARM internal memory:

The memory space accessible through the MAX of the ARM platform. It does not correspond to the peripherals, although they are accessible through the MAX as well. A data transfer from Host mem means data is read from the internal memory of the ARM or from the external memory. It is possible to point to an external memory through the MAX because it is also connected to EMI module. The next diagram replaces the SDMA and its DMA port in the hardware architecture. The Host mem is accessible through the Per DMA port of the SDMA, the EMI is accessible through the Burst DMA and the DSP mem through the DSP port.

Shared Peripheral:

A same peripheral can be connected to the shared peripheral bus (output of SPBA module) and to the ARM platform. This is the case for some UART, CSPI, and SSI. Therefore, "shared UART" indicates that the UART is connected to the shared peripheral, whereas "UART" means the UART is connected to the ARM platform.

Figure A-1 gives an overview of the SDMA in its hardware environment.

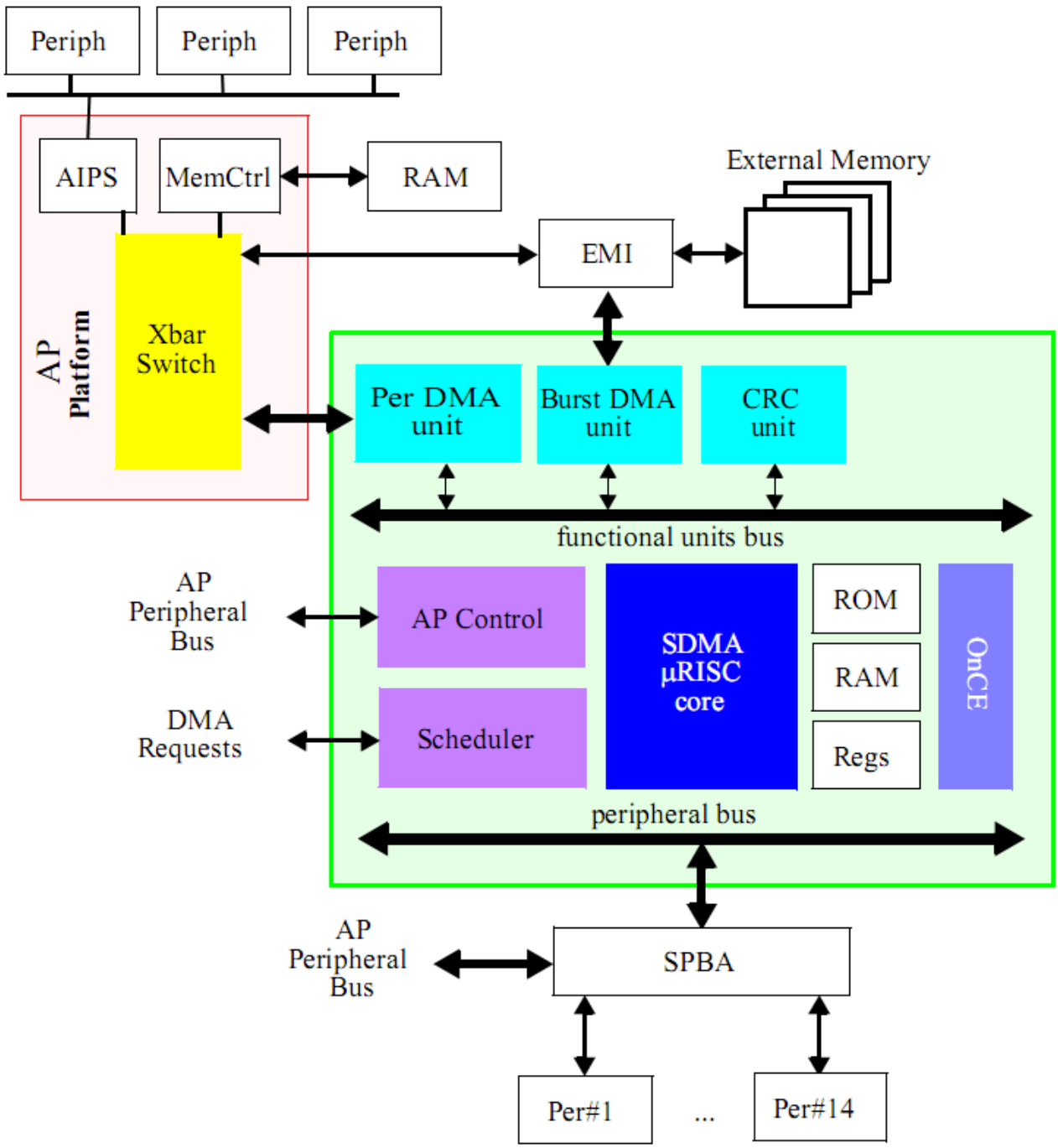


Figure A-1. SDMA Connections

A.3 Scripts Parameters: Definition

A.3.1 Parameters Definition

A.3.1.1 Parameters Required by Data Transfer Script Involving a Peripheral

The scripts that have a peripheral as a player in the transfer of data need to have the following input parameters:

A.3.1.1.1 Watermark Level (WML)

It is the upper or the lower threshold of the receive or transmit FIFO of the peripheral that triggers a DMA request to the SDMA. The WML determines the data transfer loop size, meaning the number of bytes that will be read/write from/to the receive/transmit FIFO. This parameter must be a multiple of the peripheral FIFO data size. As an example, if UART1 is the data transfer destination, data must be written to the Tx FIFO of the peripheral. For instance, the watermark level can be set to 8 bytes (the UART is a 1 byte FIFO data size), meaning the UART_TX DMA request will be sent to the SDMA each time there are more than 8 free locations in the FIFO; therefore SDMA loop size will be set to 8 and 8 bytes will be written to the UART_TX FIFO. In fact, the loop size is the minimum between the WML and the count field of the buffer descriptor attached to the channel. This concept is explained later on in the chapter. Similarly, if the UART is the source of a data transfer and if the WML is set to 16, each time the DMA request is received by the SDMA, which triggers the associated channel, the data transfer loop size is set to 16. This means 16 bytes will be read from the UART_RX FIFO. The watermark level is a "long" (32-bit data) programmed by the ARM and is not supposed to evaluate a lot during application. The WML is always given in bytes.

NOTE

For the transmit FIFO, the WML = FIFO_SIZE - FIFOTX_THRESHOLD For the receive FIFO, the WML = FIFORX_THRESHOLD The FIFOTX_THRESHOLD/ FIFORX_THRESHOLD are the values of the peripheral transmit and receive FIFO level at which a DMA request is triggered. For instance, if TX FIFO is 32 bytes depth and if the DMA request is sent when the number of bytes present in the FIFO is below a threshold of 10, the WML will be 22. It means that when the SDMA will receive the DMA request from the peripheral, it will be possible to store 22 bytes in the TX FIFO. For the receive FIFO, the WML equals the threshold.

A.3.1.1.2 Event_mask and Event2_mask

As previously said, when the WML threshold is over or under passed, a DMA request is sent to the scheduler part of the SDMA. The SDMA scheduler has 48 input pins, called "events", which are connected to the different DMA request. The mapping between the DMA request name and the event number is SoC dependent and defined in the "Interrupts and SDMA Events" chapter. For instance, in SoC A, the UART_RX DMA request is connected to events pin 5; whereas for SoC B, it may be connected to events pin 15. The script reads the EVENTS(32 events requests) and EVENTS2(remaining 16 events requests) register to check if the received DMA request is compliant with the expected one and it guaranteed that a second DMA request sent during the data transfer is not lost.

For instance, while SDMA is reading the UART_RX FIFO (again, the number of data to be read is given by WML value), the peripheral may receive new data from its input ports. Therefore after the first data transfer, the number of data available in the UART received FIFO can be still higher than the WML threshold, so a second DMA request may be sent.

The event_mask value is generally a 1-bit high value, which means that if the script attached to the channel must be triggered by DMA request number I, event_mask[I] must be set to 1, if the script attached to the channel must be triggered by DMA request number 32+I, event2_mask[I] must be set to 1. Some scripts (like ata_2_mcu) may be triggered by several DMA requests; in that case several bits of the event_mask must be asserted. The event_mask and event2_mask are long (32-bit) data.

A.3.1.1.3 Request sources

The SDMA has 48 sources of hardware requests coming from various peripherals either integrated inside the SoC, or connected outside the SoC (external DMA requests). Each request, or event, is able to trigger a single or several channels. A channel transfer can be also simply triggered in software if the script does not use the even_mas/event_mask2 variables or if the user wants to force a transfer.

Typically, the scripts (like uart_2_mcu) working with peripherals FIFO use the event_mask/event_mask2 variables to perform a data burst upon FIFO requests until count is reached. Whereas other scripts (like ap_2_ap) are just triggered once through an event (hardware or software request), and do not stop until count is reached.

A.3.1.1.4 Peripheral Address

The scripts of the SDMA scripts library are SoC independent but as their goal is to address peripheral, they required the base address or the FIFO address of the peripheral. Passing FIFO or peripheral base address depends on the scripts.

A.3.1.1.5 Data Length

The scripts whose name contains the key word "app" or "shp" are generic: they are used to transfer data from/to a 8, 16, 24 or 32-bit peripheral data size. Some jumps to some routines of these scripts depend on this peripheral size. This parameter is required as input of these scripts. This parameter is passed through the command field of the buffer descriptor and is coded on bits 25 and 24 of the mode:

00: 32-bit data transfer.

01: 8-bit data transfer.

10: 16-bit datatransfer.

11: 24-bit data transfer.

A.4 SDMA Scripts on i.MX50

The following scripts are all based on buffer descriptor mechanism.

A.4.1 SDMA ROM Scripts

The ROM holds the generic memory to memory scripts (ap_2_ap, ap_2_bp, bp_2_ap, bp_2_bp) and the peripheral most useful scripts (app_2_mcu, mcu_2_app, uart_2_mcu, uartsh_2_mcu, mcu_2_shp, shp_2_mcu).

In these memory to memory scripts, the number of bytes to transfer from the source memory to the destination is divided by 4 to get the number of 32-bit words that can be transferred. Most of the time, the transfer will be a transfer of words. There will be a byte or half data transfer at the end of the transfer if the total number of bytes to transfer is not an multiple of 4.

A.4.1.1 ap_2_ap

This script is used to transfer data inside the application processor memory using BurstDMA.

Using this script there is no restriction on the number of bytes to transmit, the source address and destination address alignment, or on the source memory and destination memory type. But word alignment and source/destination memory type will impact performance. In case that the source and destination are in the same memory type, being word aligned allows the SDMA to use the copy capability of the DMA units. When it is not the case the SRAM of the SDMA is used as temporary buffer.

A.4.1.1.1 Parameters Transmitted Through the Context ap_2_ap

None.

A.4.1.1.2 Parameters Transmitted Through the Buffer Descriptor ap_2_ap

The number of bytes to transmit is stored in the first Buffer descriptor word

The source address is stored in the second Buffer descriptor word (address field).

The destination address is stored in the Extended Buffer address

A.4.1.1.3 Use of the General Register During the Execution ap_2_ap

- r0: nb of bytes to transfer, counter during loop
- r2: channel cb address, scratch
- r3: current buffer descriptor pointer, AP destination address, scratch
- r4: mode
- r5: source address loaded by getbd,
- r6: destination address loaded by getcb, scratch
- r7:

A.4.1.1.4 Overview of Script Functionality ap_2_ap

The parameters of the transfer (number of bytes to transfer, source address and destination address) are retrieved from the buffer descriptor.

Depending on the address alignment and the DMA units involved, if possible, the transfer is performed using the copy capability of the DMA unit, or using the SDMA SRAM as a temporary buffer.

When the transfer is finished, this algorithm restarts (a new buffer descriptor is read).

The count in the BD mode word after the script is over is meaningful only if error was reported, else it has the same value as was fed as input to the script.

A.4.1.2 uartsh_2_mcu

This script is used to transfer data from the shared UART to the External memory. The UART is connected to the ARM platform. The UART is an 8-bit peripheral, but when reading the FIFO, 16 bits are retrieved. In fact, the 8 MSB are the status bytes for the current data. At each access the value of this byte is checked to verify that a valid data was retrieved. If an error is detected the transfer is stopped and the information is sent back to the Host through the buffer descriptor with the byte count field updated.

A.4.1.2.1 Parameters Transmitted Through the Context uartsh_2_mcu

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Rx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.4.1.2.2 Parameters Transmitted Through the Buffer Descriptor uartsh_2_mcu

The number of bytes to transmit is stored in the first Buffer descriptor word

The destination address is stored in the second Buffer descriptor word (address field).

The Extended Buffer address is not used

A.4.1.2.3 Use of the General Register During the Execution uartsh_2_mcu

r0 = mask to check events2

r1 = mask to check events

r2 = AP M3 destination address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.4.1.2.4 Overview of Script Functionality uartsh_2_mcu

When the UART sends a DMA request, it can be for three different reasons:

When the RX threshold is over passed. It deals with the most frequent event.

When the AGEING timer has reached its final value, meaning there is less data than the RX threshold in the RX FIFO, and they have been present for too long. It is called the aging DMA request.

When IDLE timer reached its final value; meaning the RX FIFO has been empty for too long. It is called the iddle dma request. The IDLE timer is not used, so only normal and AGEING DMA requests are supported by the script.

The first time the script is executed, a buffer descriptor is opened, the buffer destination address and its size are retrieved from the BD. Then the script checks if the RRDY bit of USR1 is set. If yes, it means that the Watermark level has been reached, there are "WML" bytes in RX FIFO. If the count field of the opened BD is higher than WML-1, the SDMA script will perform WML-1 read accesses to the RX FIFO and WML write access to the destination buffer (a read access then a write access is call a data transfer loop). If the count field is lesser than the WML-1 value, the count field will be the data transfer loop size. So the data transfer loop size equals min(WML-1,BD count)

So there will be always one data remaining in the UART RX FIFO after every data transfer loop and the ageing timer will always trigger a DMA request. It means that the channel on which the script is run will be always closed on an AGEING DMA request.

Now assume the script is triggered by a DMA request, but RRDY is not set. It means that it deals with an AGEING DMA request; there is at least one data in the RX FIFO. The data is read and written into the destination buffer, and then the RDR is checked again. If it is set and if the destination data buffer is not full (BD count is not 0), a new data is read and so on until RDR is 0. When RDR is 0, the ageing timer interrupt flag is disabled by setting the AGTIM bit, the count field of BD is updated and the current BD is closed. Then the script tries to open the next BD if the Continuous bit of the current BD was set, if there is no more BD to open, the channel is stopped.

When reading a data from the UART RX FIFO, the MSB are check to verify that a valid data has been retrieved. If an error is detected the transfer is stopped, the error bit is set in the BD, the count is updated, the BD is closed and an interrupt is sent to the ARM.

When the buffer is full, this algorithm restarts (a new buffer descriptor is read, if it exists).

The next diagram illustrates the script algorithm, it is quite complex but things become more obvious when we notice that there is a loop for the normal DMA request (in sky-blue) and on when the AGEING DMA request was detected (in yellow).

A.4.1.3 shp_2_mcu

This generic script is used to transfer data from a 8, 16, 24 or 32-bit peripheral connected to the shared peripheral bus accessed through the SPBA to memories accessed by the BurstDMA (External memories). It can be used for SSI (8, 16, 24 or 32-bit data size), for CSPI (32-bit data size) or for SDHC(MMC)/SIM (16-bit data size), these peripherals being connected to the shared peripheral bus.

A.4.1.3.1 Parameters Transmitted Through the Context shp_2_mcu

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Rx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.4.1.3.2 Parameters Transmitted Through the Buffer Descriptor shp_2_mcu

The peripheral size/data length is set in the command field of the first Buffer descriptor word, specially bits 24 and 25.

The number of bytes to transmit is stored in the first Buffer descriptor word (count field)

The destination address in the external memory is stored in the second Buffer descriptor word (address field).

The Extended Buffer address is not used.

A.4.1.3.3 Use of the General Register During the Execution shp_2_mcu

r0 = mask to check events2

r1 = mask to check events

r2 = BD destination address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.4.1.3.4 Overview of Script Functionality shp_2_mcu

The parameters of the transfer (peripheral size, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the peripheral sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" data in the FIFO.

During this data transfer loop, the number of bytes that are transferred from the RX FIFO of the peripheral to the EMI is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory write access is fixed by the parameter peripheral size.

The necessary number of transfer loops are executed in order to fill the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is filled, this algorithm restarts (a new buffer descriptor is read).

A.4.1.4 mcu_2_shp

This generic script is used to transfer data from memories accessed by the BurstDMA (External memories) to a 8, 16, 24 or 32-bit peripheral connected to the shared peripheral bus accessed through the SPBA. It can be used for SSI (8, 16, 24 or 32-bit data size), for CSPI (32-bit data size), for SDHC(MMC)/SIM (16-bit data size) or for UART3,4 (8-bit data size), these peripherals being connected to the shared peripheral bus.

A.4.1.4.1 Parameters Transmitted Through the Context mcu_2_shp

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check events - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Tx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be retrieved from the peripheral each time the channel is started.

A.4.1.4.2 Parameters Transmitted Through the Buffer Descriptor mcu_2_shp

The peripheral size/data length is set in the command field of the first Buffer descriptor word, specially bits 24 and 25.

The number of bytes to transmit is stored in the first Buffer descriptor word (count field)

The source address in the external memory is stored in the second Buffer descriptor word (address field).

The Extended Buffer address is not used.

A.4.1.4.3 Use of the General Register During the Execution mcu_2_shp

r0 = mask to check events2

r1 = mask to check events

r2 = EVENTS / EVENTS2

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.4.1.4.4 Overview of Script Functionality mcu_2_shp

The parameters of the transfer (peripheral size, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the peripheral sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" empty rooms in the FIFO.

During this data transfer loop, the number of bytes that are transferred from the EMI to the TX FIFO of the peripheral is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory read access is fixed by the parameter peripheral size.

The necessary number of transfer loops are executed in order to empty the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is empty, this algorithm restarts (a new buffer descriptor is read).

A.4.1.5 uart_2_mcu

This script is similar to uartsh_2_mcu script. The only difference between these scripts is the way SDMA access UART RX FIFO. In case of uartsh_2_mcu, the UART RX FIFO can directly be accessed by SDMA as it is on Shared Peripheral Bus. In case of uart_2_mcu, SDMA script uses one of its DMA (Peripheral DMA) to access UART RX FIFO.

A.4.1.6 app_2_mcu

This generic script is used to transfer data from a 8, 16, 24 or 32-bit peripheral connected to the AIPS accessed through the Periphera DMA of SDMA, to memories accessed by the BurstDMA (External memories). It can be used for SSI (8, 16, 24 or 32-bit data size), for CSPI (32-bit data size) or for SDHC(MMC)/SIM (16-bit data size).

A.4.1.6.1 Parameters Transmitted Through the Context app_2_mcu

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Rx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.4.1.6.2 Parameters Transmitted Through the Buffer Descriptor app_2_mcu

The Data Length peripheral size/data length is set in the command field of the first buffer descriptor word, specially bits 24 and 25.

The number of bytes to transmit is stored in the first buffer descriptor word (count field)

The destination address in the external memory is stored in the second buffer descriptor word (address field).

The Extended Buffer address is not used.

A.4.1.6.3 Use of the General Register During the Execution app_2_mcu

r0 = mask to check events2

r1 = mask to check events

r2 = AP M3 destination address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.4.1.6.4 Overview of Script Functionality app_2_mcu

The parameters of the transfer (peripheral size, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the peripheral sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" data in the FIFO.

During this data transfer loop, the number of bytes that are transferred from the RX FIFO of the peripheral to the EMI is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory write access is fixed by the parameter peripheral size.

The necessary number of transfer loops are executed in order to fill the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is filled, this algorithm restarts (a new buffer descriptor is read).

A.4.1.7 mcu_2_app

This generic script is used to transfer data from memories accessed by the BurstDMA (External memories), to a 8, 16, 24 or 32-bit peripheral connected to the AIPS accessed through the Periphera DMA of SDMA. It can be used for SSI (8, 16, 24 or 32-bit data size), for CSPI (32-bit data size) or for SDHC(MMC)/SIM (16-bit data size).

A.4.1.7.1 Parameters Transmitted Through the Context mcu_2_app

r0: mask to check events2 - If script is triggered by event 32+I, r0[I] must be set to 1. (Project dependent)

r1: mask to check event - If script is triggered by event I, r1[I] must be set to 1. (Project dependent)

r6: address of the peripheral Tx FIFO (project dependent)

r7: Watermark level - Used to determine the maximum of data that can be sent to the peripheral each time the channel is started.

A.4.1.7.2 Parameters Transmitted Through the Buffer Descriptor mcu_2_app

The peripheral size/data length is set in the command field of the first buffer descriptor word, specially bits 24 and 25.

The number of bytes to transmit is stored in the first buffer descriptor word (count field).

The source address in the external memory is stored in the second buffer descriptor word (address field).

The Extended Buffer address is not used.

A.4.1.7.3 Use of the General Register During the Execution mcu_2_app

r0 = mask to check events2

r1 = mask to check events

r2 = TX FIFO address

r3 = Ram channel context address

r4 = Bd mode

r5 = Bd mode Count

r6 = SDMA memory mapped regs base address

r7 = Watermark

A.4.1.7.4 Overview of Script Functionality mcu_2_app

The parameters of the transfer (peripheral size, number of bytes to transfer and source address) are retrieved from the buffer descriptor.

When the peripheral sends an event, a data transfer loop is executed. This event is set when there is at least "watermark level" empty room in the FIFO.

During this data transfer loop, the number of bytes that are transferred from the EMI to the TX FIFO of the peripheral is equal to the minimum value between the "watermark level" and the number of data that have not been transferred yet. For each data transfer (one read, one write), the size of the memory read access is fixed by the parameter peripheral size.

The necessary number of transfer loops are executed in order to fill the buffer, but they are executed only if the event is set. While the event is not set, the channel is not executable.

When the buffer is filled, this algorithm restarts (a new buffer descriptor is read).

A.4.2 SDMA RAM scripts

A.4.2.1 mcu_2_ssish

This script is similar to mcu_2_ssiapp. The only difference is the way SDMA core accesses the SSI TX FIFO. For script of mcu_2_ssiapp, SDMA core accesses the TX FIFO through Functional Unit Bus and regards it as the peripheral DMA, whereas for the mcu_2_ssish, the access is through the Shared Peripheral Bus and SSI can be directly accessed by SDMA.

A.4.2.2 ssish_2_mcu

This script is similar to ssiapp_2_mcu. The only difference is the way SDMA core accesses the SSI RX FIFO. For script of ssiapp_2_mcu, SDMA core accesses the RX FIFO through Functional Unit Bus and regards it as the peripheral DMA, whereas for the ssish_2_mcu, the access is through the Shared Peripheral Bus and SSI can be directly accessed by SDMA.

A.4.2.3 p_2_p

This script performs a data move of 32 bits from the peripheral's FIFO connected either on SPBA or AIPS bus to another. If destination FIFO belongs to ASRC then pad adding is performed after every N samples from the source device. If the source FIFO belongs to ASRC then after transmitting N samples a pad is swallowed. In case there is a transaction between ASRC(source) and SPDIF (destination) no pad adding or swallowing is required because SPDIF expects even number of samples.

This script presently does not deal with the transactions involving two devices connected to AIPS bus.

A.4.2.3.1 Parameters Transmitted Through the Context p_2_p

r0: LWML event mask

r1: HWML event mask

r2: source address

r6: destination address

r7: INFO. See details in Table A-1 :

p_2_p (r7 INFO)

Bits	Name	Description
0-7	Lower WML	Watermark Level
8	PS	1 : Pad Swallowing 0 : No Pad swallowing
9	PA	1 : Pad Adding 0 : No Pad swallowing
10	SPDIF	If this bit is set both source and destination are on SPBA
11	Source Bit(SP)	1 : Source on SPBA 0 : Source on AIPS

Table continues on the next page...

Bits	Name	Description
12	Destination Bit (DP)	1 : Destination on SPBA 0 : Destination on AIPS
13-15	----------	MUST BE 0
16-23	Higher WML	HWML
24-27	N	Total number of samples after which Pad adding/Swallowing must be done. It must be odd.
28	Lower WML Event (LWE)	SDMA events reg to check for LWML event mask 0 : LWE in EVENTS register 1 : LWE in EVENTS2 register
29	Higher WML Event(HWE)	SDMA events reg to check for HWML event mask 0 : HWE in EVENTS register 1 : HWE in EVENTS2 register
30	----------	MUST BE 0
31	CONT	1 : Amount of samples to be transferred is unknown and script will keep on transferring samples as long as both events are detected and script must be manually stopped by the application 0 : The amount of samples to be is equal to the count field of mode word

A.4.2.3.2 Parameters Transmitted Through the Buffer Descriptor p_2_p

In the first buffer descriptor word, the mode and number of types to transfer is included.

The Buffer address and Extended Buffer address are not used.

A.4.2.3.3 Use of the General Register During the Execution p_2_p

The register usage for each case(spba_2_spba, asrc_2_spba) are different, so don't present here.

A.4.2.3.4 Overview of Script Functionality p_2_p

The script get the number of data to transfer per event according to the bit 31(CONT) of INFO. If CONT is 0, each transfer loop will transmit the number of data as in the mode count; Or else, it means the number of samples to be transferred is not known and the transfer must take place as long as both events are detected and the script must be manually stopped by the application in case the transfer needs to be stopped.

When PS or PA is set, then after each N words transfer, a pad swallowing or pad adding will be done. This is to handle the ASRC involved transfer.

The transfer is executed only when the event is set. After each transfer for the event, the script will try to obtain a new BD.

Appendix B
i.MX50 Revision History

B.1 i.MX50 Reference Manual Revision History

The following table contains global changes made to this reference manual.

Cross-Reference	Customer-facing Description
-	Programmable register sections have been moved to the end of each chapter.
-	This chip does not support TrustZone. All references to "trust" and "TrustZone" have been removed throughout the reference manual.
SDMA Scripts	Added Appendix A, "SDMA Scripts".

B.2 Memory Map Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
ARM Platform Memory Map	Updated Block Mnemonic from DPLLIP to DPLLC
ARM Platform Memory Map	Added TEMPSENSOR address space

B.3 Interrupts and SDMA Events Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
AP Interrupts	Removed references to "TrustZone" and "secure interrupts." This product does not support Trust.

B.4 External Signals and Pin Multiplexing Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
External Signals	Modes related to ANATOP Instances were removed from I2C2_SCL, I2C2_SDA and PWM2 Pads in External Signals (Pin Assignments Report) Table.
External Signals	SRTC Instances were updated with PMIC_ON_REQ port / pad and removed from ANATOP Instances in the External Signals (Muxing Options Report) Table.

B.5 CCM Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
LP-APM Mode Restrictions	New topic added to describe LP-APM entry and exit flows valid for DDR types which support 24MHz.
Introduction	High-level clock tree diagram updated, ddr_clk source added.
CCM_CLK_SWITCHER	Switcher clock gen figure updated, index added.
CCM_CLK_ROOT_GEN	Root clock gen figures were updated and legacy clock tree branches were removed.
CCM Analog Memory Map/Register Definition	Correction to ANATOP_MISCn[PLL_HOLD_RING_OFF] (CCM_ANALOG_MISCn[PLL_HOLD_RING_OFF]) access from RO to RW.
CCM Memory Map/Register Definition	Correction to ANATOP_PLLCTRLn[FORCE_LOCK] (CCM_ANALOG_PLLCTRLn[FORCE_LOCK]) access from Reserved to RW.
CCM Analog Memory Map/Register Definition	Changed "ANATOP" to "CCM Analog"
CCM External Signals Description	Removed internal signals not referenced in clock figures or register summary.

B.6 System Boot Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
Boot OCOTP bit Descriptions	Changed BT_MMU_ENABLE bit name to BT_MMU_DISABLE and updated description.
Enabling MMU/Caches	Added description of effect of BT_MMU_DISABLE fuse bit on MMU/caches.
Error Logging	Removed ROM address and associated ROM symbol of error status log.
External Entry	New description of access to ROM USB downloader from program image code.
Memory Map	Updated OCROM and OCRAM Memory Map to make address range 0xF800_4000 to 0xF800_5FFF Reserved.
Boot OCOTP bit Descriptions	Removed section ROM updates to tapeout 1.1 and absorbed those changes to TO 1.1 in Boot OCOTP Bit descriptions.

B.7 AIPSTZ Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
Introduction	Added NOTE in Introduction to clarify that AIPSTZ in this chip does not support TrustZone. Removed references to "trust" and "TrustZone" throughout the chapter.

B.8 DPLLC Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
DPLLC Memory Map/Register Definition	In Section "DPLLC Memory Map/Register Definition," corrected the positions of bits.
DPLLC Memory Map/Register Definition	In Section "DPLLC Memory Map/Register Definition," corrected restes of UPEN and ref_clk_sel[1:0]. Also, replaced "10 clock2 is selected" with "10 clock2 is selected (24MHz oscillator clock)" in the field description of ref_clk_sel[1:0].

B.9 EIM Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
Multiplexed Address/Data Mode	Updated second sentcnce of Section "Multiplexed Address/ Data Mode"
Multiplexed Address/Data Mode	Updated Table 23-1 as per the IMX53 Data sheet.
EIM Memory Map/Register Definition	In Section "EIM Memory Map/Register Definition" updated offset/base addresses for "Programmable Registers."
EIM Memory Map/Register Definition	In Table "EIM_CSnWCR1 field descriptions," updated WWSC row.
Boot Mode	In Section "Boot Mode," added a cross reference to Chapter "System Boot."
Boot Mode	In Table "Table 23-2," removed first 7 rows.
Other Important Block I/O Signals Internal to the SoC	Updated EIM_BOOT row of the table in Section "Other Important Block I/O Signals Internal to the SoC."
EIM Memory Map/Register Definition	Updated NOTEs in AUS, CSREC, DSZ in EIM_CSnGCR1 field descriptions in Table "EIM_CSnGCR1 field descriptions."
EIM Memory Map/Register Definition	Removed the NOTE in ERRST row of Table "EIM_WIAR field descriptions."
Booting from EIM	Updated second paragraph of Section "Booting from EIM."
Access to AMD Flash	Removed Section "AMD Flash Boot."
Access to Intel Sibley Flash	Removed Section "Intel Sibley Flash Boot."
MDOC Device Boot	Updated Section "MDOC Device Boot."
Samsung OneNAND Boot	Updated Section "Samsung OneNAND Boot."
Access to Spansion Flash	Removed Section "Spansion Flash Boot."
Overview	Replaced "EIM_BOOT[12:0]" with "EIM_BOOT[2:0]" in Figure "EIM Diagram."
EIM Memory Map/Register Definition	In Section "Chip Select n Read Configuration Register 1," added footnotes for RWSC and OEA bit fields.
EIM Memory Map/Register Definition	In Section "Chip Select n Write Configuration Register 1," added a footnote for WWSC bit field.
EIM Memory Map/Register Definition	In Section "Chip Select n General Configuration Register 1," added a footnotes for CSREC, DSZ, and MUM bit fields.
EIM Memory Map/Register Definition	In Section "Chip Select n General Configuration Register 1," updated the NOTE in in the field description of CSREC row.

B.10 GPIO Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
-	Removed 30.2 External Signals Description and 30.2.1 External Signals Overview topics, as they contained only internal signals.

B.11 I2C Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
I2C Memory Map/Register Definition	Added description of the I2C SION bit, which needs to be configured by software.

B.12 IOMUXC Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
IOMUXC	Removed ANATOP select from SMUXC_PI2C2_SCL[MUX_MODE], SMUXC_PI2C2_SDA[MUX_MODE] and SMUXC_PPWM2[MUX_MODE] bitfields.

B.13 ROMCP Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
ROMC Architecture Diagram	Updated ROMC Internal Architecture Figure

B.14 SDMA Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
ARM Platform Memory Map and Control Register Definitions	In Section "ARM platform Channel 0 Pointer (SDMAARM_MC0PTR)," replaced the text "Refer to the API document i.MX53 SDMA Scripts User Manual for the use of this register" with "Appendix A fully describes the SDMA Application Programming Interface (API) in SDMAARM_MC0PTR field descriptions."
Standard Boot Sequence	In Section "Standard Boot Sequence," replaced the text "4. Perform any necessary setup as required by the standard boot script in ROM (this is described in i.MX53 SDMA Scripts User Manual)" with "4. Perform any necessary setup as required by the standard boot script in ROM (this is described in Appendix A)."
SDMA Initialization	In Section "SDMA Initialization," replaced the text "The API document i.MX53 SDMA Scripts User Manual describes the setup of the SDMA" with "The Appendix A describes the setup of the SDMA."
References	In Section "References," updated the text to "See Appendix A for the SDMA Application Programming Interface (API)."
SDMA Internal (Core) Memory Map and Internal Register Definitions	Changed reset value of SDMACORE_ENDIANNESS[APEND] to 1 and added that bit is tied to logic '1'.

B.15 SSI Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
Features	Added min/max audio sampling rates for I2S. Added min/max frame rates for AC97.

B.16 TZIC Revision History

The following table contains changes made to this block guide.

Topic Cross-Reference	Customer-facing Description
Overview	Added NOTE in Overview to clarify that TZIC does not support TrustZone. Removed references to "trust" and "TrustZone" throughout the chapter.
Overview	In NOTE, removed product-specific reference and updated reference from "System Interrupts chapter" to "Interrupts and SDMA Events chapter."

Table continues on the next page...

Topic Cross-Reference	Customer-facing Description
Interrupt Priority	Removed redundant table - content is described in the register description.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: IMX50RM
Rev. 1, 10/2011

freescale™
semiconductor