



MAX78000 User Guide

UGXXXX; Rev 0.5; 08/31/2020

Preliminary Draft 08/31/2020

Abstract: This user guide provides application developers information on how to use the memory and peripherals of the MAX78000 microcontroller. Detailed information for all registers and fields in the device are covered. Guidance is given for managing all the peripherals, clocks, power and startup for the device family.

MAX78000 User Guide

Table of Contents

| | |
|--|----|
| MAX78000 User Guide ----- | 2 |
| 1. Overview ----- | 20 |
| 1.1 Block Diagram ----- | 21 |
| 2. Memory, Register Mapping, and Access ----- | 22 |
| 2.1 Memory, Register Mapping, and Access Overview----- | 22 |
| 2.2 Field Access Definitions----- | 27 |
| 2.3 Standard Memory Regions ----- | 27 |
| 2.4 AHB Interfaces----- | 31 |
| 2.4.1.1 I-Code----- | 31 |
| 2.4.1.2 D-Code----- | 31 |
| 2.4.1.3 System----- | 31 |
| 2.4.2.1 Standard DMA----- | 31 |
| 2.4.2.2 CNN and CNN TX FIFO ----- | 31 |
| 2.4.2.3 SPIO----- | 31 |
| 2.5 Peripheral Register Map ----- | 32 |
| 2.6 Error Correction Coding (ECC) Module----- | 33 |
| 3. System, Power, Clocks, Reset ----- | 35 |
| 3.1 Oscillator Sources----- | 35 |
| 3.2 System Oscillator (SYS_OSC) ----- | 36 |
| 3.2.1.1 System Oscillator Selection----- | 36 |
| 3.2.1.2 System Clock (SYS_CLK) ----- | 37 |
| 3.3 Operating Modes ----- | 39 |
| 3.3.2.1 Entering SLEEP----- | 39 |
| 3.3.3.1 Entering LPM----- | 41 |
| 3.3.4.1 Entering UPM----- | 43 |
| 3.3.5.1 Entering STANDBY----- | 45 |
| 3.3.6.1 Entering BACKUP----- | 47 |
| 3.3.7.1 Entering PDM ----- | 49 |
| 3.4 Operating Modes Wakeup Sources ----- | 51 |
| 3.5 Device Resets ----- | 51 |
| 3.6 Unified Internal Cache Controllers----- | 52 |
| 3.7 RAM Memory Management ----- | 55 |
| 3.8 Miscellaneous Control Registers (MCR) ----- | 55 |
| 3.9 Single Inductor Multiple Output (SIMO) Power Supply----- | 59 |
| 3.10 Low-Power General Control Registers (LPGCR)----- | 65 |
| 3.11 Power Sequencer Registers (PWRSEQ) ----- | 67 |
| 3.12 Trim System Initialization Registers (TRIMSIR) ----- | 74 |

Preliminary Draft 08/31/2020

| | | |
|------|---|-------------------------------------|
| 3.13 | <i>Global Control Registers (GCR)</i> | 77 |
| 3.14 | <i>Error Correction Coding (ECC)</i> | 94 |
| 3.15 | <i>System Initialization Registers (SIR)</i> | 95 |
| 3.16 | <i>Function Control Registers (FCR)</i> | 97 |
| 3.17 | <i>General Control Function Registers (GCFR)</i> | 99 |
| 4. | Interrupts and Exceptions | 102 |
| 4.1 | <i>CM4 Interrupt and Exception Features</i> | 102 |
| 4.2 | <i>CM4 Interrupt Vector Table</i> | 102 |
| 4.3 | <i>RV32 Interrupt Vector Table</i> | 104 |
| 5. | General-Purpose I/O and Alternate Function Pins (GPIO) | 106 |
| 5.1 | <i>Instances</i> | 106 |
| 5.2 | <i>Single Inductor Multiple Output Registers (SIMO)</i> | Error! Bookmark not defined. |
| 5.3 | <i>Configuration</i> | 113 |
| 5.4 | <i>Reference Tables</i> | 115 |
| 5.5 | <i>Usage</i> | 116 |
| 5.6 | <i>Configuring GPIO (External) Interrupts</i> | 117 |
| 5.7 | <i>Registers</i> | 119 |
| 6. | Flash Controller (FLC) | 128 |
| 6.1 | <i>Instances</i> | 128 |
| 6.2 | <i>Usage</i> | 128 |
| 6.3 | <i>Registers</i> | 131 |
| 7. | Debug Access Port (DAP) | 137 |
| 7.1 | <i>Instances</i> | 137 |
| 7.2 | <i>Access Control</i> | 137 |
| 7.3 | <i>Pin Configuration</i> | 137 |
| 8. | Semaphores | 138 |
| 8.1 | <i>Instances</i> | 138 |
| 8.2 | <i>Multiprocessor Communications</i> | 138 |
| 8.3 | <i>Registers</i> | 138 |
| 9. | Standard DMA (DMA) | 144 |
| 9.1 | <i>Instances</i> | 144 |
| 9.2 | <i>DMA Channel Operation (DMA_CH)</i> | 144 |
| 9.3 | <i>Usage</i> | 148 |
| 9.4 | <i>Count-To-Zero (CTZ) Condition</i> | 148 |
| 9.5 | <i>Chaining Buffers</i> | 149 |
| 9.6 | <i>DMA Interrupts</i> | 151 |
| 9.7 | <i>Channel Timeout Detect</i> | 151 |

| | | |
|----------|--|-----|
| 9.8 | <i>Memory-to-Memory DMA</i> | 152 |
| 9.9 | <i>DMA Registers</i> | 152 |
| 9.10 | <i>DMA Channel Registers</i> | 153 |
| 10. | Analog to Digital Converter (ADC) and Comparators (COMP) | 159 |
| 10.1 | <i>Features</i> | 159 |
| 10.2 | <i>Instances</i> | 159 |
| 10.3 | <i>Architecture</i> | 160 |
| 10.4 | <i>Clock Configuration</i> | 162 |
| 10.5 | <i>Power-Up Sequence</i> | 163 |
| 10.6 | <i>Conversion</i> | 163 |
| 10.7 | <i>Reference Scaling and Input Scaling</i> | 163 |
| 10.7.5.1 | <i>Power-Down Sequence</i> | 167 |
| 10.8 | <i>Comparator Operation</i> | 167 |
| 10.9 | <i>ADC Registers</i> | 169 |
| 10.10 | <i>Low-Power Comparator Registers</i> | 173 |
| 11. | UART (UART/LPUART) | 175 |
| 11.1 | <i>Instances</i> | 176 |
| 11.2 | <i>DMA</i> | 177 |
| 11.3 | <i>UART Frame</i> | 177 |
| 11.4 | <i>FIFOs</i> | 178 |
| 11.5 | <i>Interrupt Events</i> | 178 |
| 11.6 | <i>Wakeup Events</i> | 181 |
| 11.7 | <i>Resets</i> | 181 |
| 11.8 | <i>RX Sampling</i> | 181 |
| 11.9 | <i>Baud Rate Generation</i> | 182 |
| 11.10 | <i>Baud Rate Calculation</i> | 183 |
| 11.11 | <i>Low-Power 9600 Baud Receiver Operation</i> | 183 |
| 11.12 | <i>Hardware Flow Control</i> | 185 |
| 11.13 | <i>Registers</i> | 186 |
| 12. | Serial Peripheral Interface (SPI) | 194 |
| 12.1 | <i>Instances</i> | 195 |
| 12.2 | <i>SPI Formats</i> | 196 |
| 12.3 | <i>Pin Configuration</i> | 198 |
| 12.4 | <i>SPI Clock Configuration</i> | 199 |
| 12.5 | <i>Registers</i> | 202 |
| 13. | I2C Master/Slave Serial Communications Peripheral (I2C) | 214 |
| 13.1 | <i>I2C Master/Slave Features</i> | 214 |

| | | |
|----------|---|-----|
| 13.2 | <i>Instances</i> | 214 |
| 13.3 | <i>I²C Overview</i> | 215 |
| 13.4 | <i>I²C Configuration and Usage</i> | 217 |
| 13.4.4.1 | Hs-Mode Timing | 218 |
| 13.4.4.2 | Hs-Mode Clock Configuration | 218 |
| 13.4.6.1 | I ² C Master Mode Receiver Operation | 221 |
| 13.4.6.2 | I ² C Master Mode Transmitter Operation | 221 |
| 13.4.6.3 | I ² C Multi-Master Operation | 221 |
| 13.4.7.1 | Slave Transmitter | 223 |
| 13.4.7.2 | Slave Receivers | 226 |
| 13.5 | <i>Registers</i> | 231 |
| 14. | Inter-Integrated Sound Interface (I ² S) | 247 |
| 14.1 | <i>Instances</i> | 247 |
| 14.2 | <i>Details</i> | 247 |
| 14.3 | <i>I²S Clocking</i> | 249 |
| 14.4 | <i>Interrupt Events</i> | 250 |
| 14.5 | <i>Wakeup Events</i> | 251 |
| 14.6 | <i>Direct Memory Access</i> | 251 |
| 14.7 | <i>Block Operation</i> | 252 |
| 14.8 | <i>I²S Master/Slave Configuration</i> | 252 |
| 14.9 | <i>Data Formatting</i> | 252 |
| 14.10 | <i>Stereo/Mono Configuration</i> | 255 |
| 14.11 | <i>FIFOs</i> | 255 |
| 14.12 | <i>Registers</i> | 256 |
| 15. | Parallel Camera Interface (PCIF) | 261 |
| 15.1 | <i>Instances</i> | 261 |
| 15.2 | <i>Capture Modes</i> | 262 |
| 15.3 | <i>Timing Modes</i> | 262 |
| 15.4 | <i>Data Width</i> | 262 |
| 15.5 | <i>Data FIFO</i> | 264 |
| 15.6 | <i>Usage</i> | 264 |
| 15.7 | <i>Parallel Camera Registers</i> | 265 |
| 16. | 1-Wire Master (OWM) | 269 |
| 16.1 | <i>1-Wire Master Features</i> | 269 |
| 16.2 | <i>1-Wire Pins and Configuration</i> | 270 |
| 16.3 | <i>1-Wire Protocol</i> | 270 |
| 16.3.1.1 | Physical Layer | 271 |
| 16.3.1.2 | Link Layer | 271 |
| 16.3.1.3 | Network Layer | 274 |
| 16.4 | <i>1-Wire Operation</i> | 277 |

| | | |
|----------|--|-----|
| 16.5 | <i>1-Wire Data Reads</i> | 278 |
| 16.6 | <i>Registers</i> | 279 |
| 17. | Real-Time Clock (RTC) | 284 |
| 17.1 | <i>Overview</i> | 284 |
| 17.2 | <i>Instances</i> | 285 |
| 17.3 | <i>Register Access Control</i> | 285 |
| 17.4 | <i>RTC Alarm Functions</i> | 286 |
| 17.5 | <i>RTC Calibration</i> | 288 |
| 17.6 | <i>Registers</i> | 290 |
| 18. | Timers (TMR/LPTMR) | 295 |
| 18.1 | <i>Instances</i> | 296 |
| 18.2 | <i>Basic Timer Operation</i> | 296 |
| 18.3 | <i>32-Bit Single / 32-Bit Cascade / Dual 16-Bit</i> | 297 |
| 18.4 | <i>Timer Clock Sources</i> | 297 |
| 18.5 | <i>Timer Pin Functionality</i> | 298 |
| 18.6 | <i>Wakeup Events</i> | 300 |
| 18.7 | <i>PWM Operation</i> | 300 |
| 18.8 | <i>Operating Modes</i> | 303 |
| 18.8.5.1 | <i>Capture Event (Timer Input Signal)</i> | 315 |
| 18.8.5.2 | <i>Capture Event (Software Capture)</i> | 315 |
| 18.8.5.3 | <i>Rollover Event</i> | 315 |
| 18.9 | <i>Registers</i> | 323 |
| 19. | Wakeup Timer (WUT) | 332 |
| 19.1 | <i>Basic Operation</i> | 332 |
| 19.2 | <i>Timer Pin Functionality</i> | 333 |
| 19.3 | <i>One-Shot Mode (000b)</i> | 333 |
| 19.4 | <i>Continuous Mode (001b)</i> | 335 |
| 19.5 | <i>Registers</i> | 337 |
| 20. | Watchdog Timer (WDT) | 340 |
| 20.1 | <i>Instances</i> | 341 |
| 20.2 | <i>Usage</i> | 341 |
| 20.3 | <i>WDT Feed Sequence</i> | 342 |
| 20.4 | <i>WDT Events</i> | 342 |
| 20.5 | <i>Initializing the WDT</i> | 346 |
| 20.6 | <i>Resets</i> | 346 |
| 20.7 | <i>Using the WDT as a Long-Interval Timer</i> | 346 |
| 20.8 | <i>Using the WDT as a Long-Interval Wakeup Timer</i> | 347 |
| 21. | Pulse Train Engine (PT) | 352 |

| | | |
|----------|--|-----|
| 21.1 | <i>Instances</i> | 352 |
| 21.2 | <i>Pulse Train Engine Features</i> | 352 |
| 21.3 | <i>Engine</i> | 352 |
| 21.3.1.1 | Pulse Train Mode | 352 |
| 21.3.1.2 | In Pulse Train Mode, Set the Bit Pattern | 353 |
| 21.3.1.3 | Pulse Train Loop Mode | 353 |
| 21.3.1.4 | Pulse Train Loop Delay | 353 |
| 21.3.1.5 | Pulse Train Automatic Restart Mode | 353 |
| 21.4 | <i>Enabling and Disabling a Pulse Train Output</i> | 354 |
| 21.5 | <i>Atomic Pulse Train Output Enable and Disable</i> | 354 |
| 21.6 | <i>Pulse Train Halt and Disable</i> | 354 |
| 21.7 | <i>Pulse Train Interrupts</i> | 355 |
| 21.8 | <i>Registers</i> | 355 |
| 21.8.1.1 | Pulse Train Engine Safe Enable Register | 365 |
| 21.8.1.2 | Pulse Train Engine Safe Enable Register | 366 |
| 22. | CRC | 371 |
| 22.1 | <i>Instances</i> | 371 |
| 22.2 | <i>Usage</i> | 371 |
| 22.3 | <i>Polynomial Generation</i> | 372 |
| 22.4 | <i>Calculations Using Software Writes to FIFO</i> | 372 |
| 22.5 | <i>Calculations Using DMA</i> | 373 |
| 22.6 | <i>Interrupt Events</i> | 374 |
| 22.7 | <i>Wakeup Events</i> | 374 |
| 22.8 | <i>Registers</i> | 374 |
| 23. | AES | 376 |
| 23.1 | <i>Instances</i> | 376 |
| 23.2 | <i>Encryption of 128-Bit Blocks of Data Using FIFO</i> | 376 |
| 23.3 | <i>Encryption of 128-Bit Blocks Using DMA</i> | 376 |
| 23.4 | <i>Encryption of Blocks Less Than 128-Bits</i> | 378 |
| 23.5 | <i>Decryption</i> | 378 |
| 23.6 | <i>Interrupt Events</i> | 378 |
| 23.7 | <i>Wakeup Events</i> | 379 |
| 23.8 | <i>Registers</i> | 379 |
| 24. | TRNG Engine | 382 |
| 24.1 | <i>TRNG Registers</i> | 382 |
| 25. | Bootloader | 384 |
| 25.1 | <i>Instances</i> | 384 |
| 25.2 | <i>Bootloader Operating States</i> | 385 |
| 25.3 | <i>Creating the Motorola SREC File</i> | 386 |

| | | |
|----------|--|-----|
| 25.4 | <i>Bootloader Activation</i> | 387 |
| 25.5 | <i>Bootloader</i> | 387 |
| 25.6 | <i>Secure Bootloader</i> | 388 |
| 25.7 | <i>Command Protocol</i> | 389 |
| 25.8 | <i>General Commands</i> | 390 |
| 25.9 | <i>Secure Commands</i> | 394 |
| 25.10 | <i>Challenge/Response Commands</i> | 397 |
| 26. | Convolutional Neural Network (CNN) | 398 |
| 26.1 | <i>Overview</i> | 398 |
| 26.2 | <i>Instances</i> | 402 |
| 26.3 | <i>Memory Configuration</i> | 403 |
| 26.4 | <i>CNN Global Registers (CNN)</i> | 412 |
| 26.5 | <i>CNNx16 Processor Array (CNNx16_n) Registers</i> | 417 |
| 26.5.2.1 | <i>CNNx16_n Per Layer Registers (CNNx16_n_Ly)</i> | 428 |
| 26.5.2.2 | <i>CNNx16_n Processor Stream Registers</i> | 439 |
| 27. | Revision History | 442 |

Preliminary Draft 08/31/2020

List of Figures

| | |
|--|-----|
| Figure 1-1: MAX78000 Block Diagram | 21 |
| Figure 2-1: CM4 Code Memory Mapping | 23 |
| Figure 2-2: RISC-V IBUS Code Memory Mapping | 24 |
| Figure 2-3: CM4 Peripheral and Data Memory Mapping..... | 25 |
| Figure 2-4: RV32 Peripheral and Data Memory Mapping..... | 26 |
| Figure 2-5: Unique Serial Number Format..... | 28 |
| Figure 3-1: MAX78000 Clock Block Diagram..... | 38 |
| Figure 3-2: SLEEP Mode Clock Control..... | 40 |
| Figure 3-3: Low Power Mode (LPM) Clock and State Retention Diagram | 42 |
| Figure 3-4: Micro Power Mode (UPM) Clock and State Retention Block Diagram | 44 |
| Figure 3-5: STANDBY Mode Clock and State Retention Block Diagram..... | 46 |
| Figure 3-6: BACKUP Mode Clock and State Retention Block Diagram..... | 48 |
| Figure 3-7: Power Down Mode (PDM) Clock and State Retention Block Diagram | 50 |
| Figure 9-1: DMA Block-Chaining Flowchart | 150 |
| Figure 10-1: Analog to Digital Converter Block Diagram | 161 |
| Figure 10-2: ADC Limit Engine | 166 |
| Figure 11-1: UART Block Diagram | 176 |
| Figure 11-2: UART Frame Structure | 177 |
| Figure 11-3: MAX78000 UART Interrupt and Wakeup Functional Diagram | 179 |
| Figure 11-4: Oversampling Example | 182 |
| Figure 11-5: UART Timing Generation (FDM not supported) | 182 |
| Figure 11-6: LPUART Timing Generation (FDM supported)..... | 183 |
| Figure 11-7. Hardware Flow Control, Physical Connection | 185 |
| Figure 11-8. Hardware Flow Control Signaling, Transmit to External Receiver | 186 |
| Figure 12-1: SPI Block Diagram | 195 |
| Figure 12-2. 4-Wire SPI Connection Diagram | 197 |
| Figure 12-3: Generic 3-Wire SPI Master to Slave Connection | 198 |
| Figure 12-4: Dual Mode SPI Connection Diagram..... | 199 |
| Figure 12-5: SCK Clock Rate Control | 200 |
| Figure 12-6: SPI Clock Polarity | 201 |
| Figure 13-1: I ² C Write Data Transfer..... | 216 |
| Figure 13-2: I ² C SCL Timing for Standard, Fast and Fast-Plus Modes | 217 |
| Figure 14-1: Full Duplex Connection, I ² S Master Mode | 248 |
| Figure 14-2: Full Duplex Connection, I ² S Slave Mode | 249 |
| Figure 14-3: I ² S Interrupt Functional Diagram | 250 |
| Figure 15-1: Horizontal and Vertical Synchronization Timing Mode with 8-Bit Data Width | 263 |
| Figure 15-2: Data Stream Timing Mode with 8-Bit Data Width..... | 263 |
| Figure 15-3: 10 or 12-bit PCIF_VSYNC/PCIF_HSYNC | 264 |
| Figure 16-1: 1-Wire Signal Interface | 271 |
| Figure 16-2: 1-Wire Reset Pulse..... | 272 |
| Figure 16-3: 1-Wire Write Time Slot | 273 |
| Figure 16-4: 1-Wire Read Time Slot | 273 |
| Figure 16-5: 1-Wire ROM ID Fields | 274 |
| Figure 17-1: MAX78000 RTC Block Diagram (12-bit Sub-Second Counter) | 284 |
| Figure 17-2: RTC Interrupt/Wakeup Diagram Wakeup Function..... | 287 |
| Figure 17-3: Internal Implementation of Digital Trim, 4kHz..... | 289 |
| Figure 18-1: MAX78000 TimerA Output Functionality, Modes 0/1/3/5..... | 299 |
| Figure 18-2: MAX78000 TimerA Input Functionality, Modes 2/4/6/7/8/14..... | 300 |

| | |
|---|-----|
| Figure 18-3: PWM Synchronous Mode Circuit Diagram | 301 |
| Figure 18-4: Timer Output Complementary Waveforms..... | 302 |
| Figure 18-5: One-Shot Mode Diagram..... | 308 |
| Figure 18-6: Continuous Mode Diagram..... | 310 |
| Figure 18-7: Counter Mode Diagram | 312 |
| Figure 18-8: Capture Mode Diagram | 316 |
| Figure 18-9: Compare Mode Diagram | 318 |
| Figure 18-10: Gated Mode Diagram | 320 |
| Figure 18-11: Capture/Compare Mode Diagram..... | 322 |
| Figure 19-1: One-Shot Mode Diagram..... | 333 |
| Figure 19-2: Continuous Mode Diagram..... | 335 |
| Figure 20-1: Windowed Watchdog Timer Block Diagram..... | 341 |
| Figure 20-2: WDT Early Interrupt and Reset Event Sequencing Details | 344 |
| Figure 20-3: WDT Late Interrupt and Reset Event Sequencing Details | 345 |
| Figure 25-1: MAX78000 Combined Bootloader Flow | 388 |
| Figure 26-1: CNN Overview..... | 401 |
| Figure 26-2: CNNx16_n Processor Quadrant Block Diagram..... | 402 |
| Figure 26-3: CNN Global and Quad CNNx16n Processor Array APB Memory Map | 403 |

Preliminary Draft 08/31/2020

List of Tables

| | |
|---|----|
| Table 2-1: Field Access Definitions | 27 |
| Table 2-2: System SRAM Configuration..... | 29 |
| Table 2-3: AHB Slave Base Address Map | 32 |
| Table 2-4: APB Peripheral Base Address Map..... | 32 |
| Table 3-1: Available System Oscillators | 36 |
| Table 3-2: Reset Sources and Effect on Oscillator and System Clock | 37 |
| Table 3-3 System RAM Retention in BACKUP Mode..... | 47 |
| Table 3-4: Wakeup Sources for Each Operating Mode in the MAX78000..... | 51 |
| Table 3-5: Operating Mode and Clock Status | 52 |
| Table 3-6: Instruction Cache Controller Register Summary..... | 53 |
| Table 3-7: ICC0 Cache Information Register | 53 |
| Table 3-8: ICC0 Memory Size Register | 54 |
| Table 3-9: ICC0 Cache Control Register | 54 |
| Table 3-10: ICC0 Invalidate Register | 54 |
| Table 3-11: Miscellaneous Control Register Summary | 55 |
| Table 3-12: Error Correction Coding Enable Register | 56 |
| Table 3-13: IPO Manual Register | 56 |
| Table 3-14: Output Enable Register..... | 56 |
| Table 3-15: Comparator 0 Control Register | 57 |
| Table 3-16: Miscellaneous Control Register | 57 |
| Table 3-17: GPIO3 Pin Control Register | 58 |
| Table 3-18: SIMO Power Supply Device Pin Connectivity..... | 59 |
| Table 3-19: SIMO Controller Register Summary | 60 |
| Table 3-20: SIMO Buck Voltage Regulator A Control Register..... | 60 |
| Table 3-21: SIMO Buck Voltage Regulator B Control Register..... | 61 |
| Table 3-22: SIMO Buck Voltage Regulator C Control Register..... | 61 |
| Table 3-23: SIMO High Side FET Peak Current VREGO_A VREGO_B Register..... | 62 |
| Table 3-24: SIMO High Side FET Peak Current VREGO_C Register | 62 |
| Table 3-25: SIMO Maximum High Side FET Time On Register | 63 |
| Table 3-26: SIMO Buck Cycle Count VREGO_A Register | 63 |
| Table 3-27: SIMO Buck Cycle Count VREGO_B Register | 63 |
| Table 3-28: SIMO Buck Cycle Count VREGO_C Register | 63 |
| Table 3-29: SIMO Buck Cycle Count Alert VREGO_A Register | 63 |
| Table 3-30: SIMO Buck Cycle Count Alert VREGO_B Register | 64 |
| Table 3-31: SIMO Buck Cycle Count Alert VREGO_C Register | 64 |
| Table 3-32: SIMO Buck Regulator Output Ready Register | 64 |
| Table 3-33: SIMO Zero Cross Calibration VREGO_A Register | 65 |
| Table 3-34: SIMO Zero Cross Calibration VREGO_B Register | 65 |
| Table 3-35: SIMO Zero Cross Calibration VREGO_C Register | 65 |
| Table 3-36 Low-Power Control Register Summary | 66 |
| Table 3-37: Reset Control Register | 66 |
| Table 3-38: Clock Disable Register | 66 |
| Table 3-39: Power Sequencer Register Summary..... | 67 |
| Table 3-40: Low Power Control Register | 68 |
| Table 3-41: GPIO0 Low Power Wakeup Status Flags | 69 |
| Table 3-42: GPIO0 Low Power Wakeup Enable Registers..... | 69 |
| Table 3-43: GPIO1 Low Power Wakeup Status Flags | 70 |
| Table 3-44: GPIO1 Low Power Wakeup Enable Registers..... | 70 |
| Table 3-45: GPIO2 Low Power Wakeup Status Flags | 70 |
| Table 3-46: GPIO2 Low Power Wakeup Enable Registers..... | 71 |

| | |
|---|-----|
| Table 3-47: GPIO3 Low Power Wakeup Status Flags | 71 |
| Table 3-48: GPIO3 Low Power Wakeup Enable Registers..... | 71 |
| Table 3-49: Low Power Peripheral Wakeup Status Flags..... | 72 |
| Table 3-50: Low Power Peripheral Wakeup Enable Registers | 72 |
| Table 3-51: Low Power General Purpose Register 0..... | 74 |
| Table 3-52: Low Power General Purpose Register 1..... | 74 |
| Table 3-53: Trim System Initialization Register Summary | 75 |
| Table 3-54: RTC Trim System Initialization Register | 75 |
| Table 3-55: SIMO Trim System Initialization Register..... | 75 |
| Table 3-56: IPO Low Trim System Initialization Register | 76 |
| Table 3-57: Control Trim System Initialization Register..... | 76 |
| Table 3-58: INRO Trim System Initialization Register | 77 |
| Table 3-59: Global Control Register Summary..... | 77 |
| Table 3-60: System Control Register..... | 78 |
| Table 3-61: Reset Register 0 | 79 |
| Table 3-62: Clock Control Register..... | 81 |
| Table 3-63: Power Management Register | 82 |
| Table 3-64: Peripheral Clock Divisor Register | 84 |
| Table 3-65: Peripheral Clock Disable Register 0 | 84 |
| Table 3-66: Memory Clock Control Register | 86 |
| Table 3-67: Memory Zeroize Control Register | 87 |
| Table 3-68: System Status Flag Register | 88 |
| Table 3-69: Reset Register 1 | 88 |
| Table 3-70: Peripheral Clock Disable Register 1 | 90 |
| Table 3-71: Event Enable Register | 91 |
| Table 3-72: Revision Register..... | 92 |
| Table 3-73: System Status Interrupt Enable Register | 92 |
| Table 3-74: Error Correction Coding Error Register | 92 |
| Table 3-75: Error Correction Coding Correctable Error Detected Register | 92 |
| Table 3-76: Error Correction Coding Interrupt Enable Register..... | 93 |
| Table 3-77: Error Correction Coding Error Address Register | 93 |
| Table 3-78: General Purpose Register 0 | 94 |
| Table 3-79: Error Correction Coding Enable Register Summary | 94 |
| Table 3-80: Error Correction Coding Enable Register | 94 |
| Table 3-81: System Initialization Register Summary..... | 95 |
| Table 3-82: System Initialization Status Register | 95 |
| Table 3-83: System Initialization Address Error Register | 96 |
| Table 3-84: System Initialization Function Status Register | 96 |
| Table 3-85: System Initialization Security Function Status Register | 96 |
| Table 3-86: Function Control Register Summary | 97 |
| Table 3-87: Function Control 0 Register | 97 |
| Table 3-88: IPO Automatic Calibration 0 Register | 98 |
| Table 3-89: IPO Automatic Calibration 1 Register | 98 |
| Table 3-90: IPO Automatic Calibration 2 Register | 98 |
| Table 3-91: RV32 Boot Address Register | 99 |
| Table 3-92: RV32 Control Register | 99 |
| Table 3-93: General Control Function Register Summary | 100 |
| Table 3-94: General Control Function Register 0..... | 100 |
| Table 3-95: General Control Function Register 1..... | 100 |
| Table 3-96: General Control Function Register 2..... | 101 |
| Table 3-97: General Control Function Register 3..... | 101 |
| Table 4-1: MAX78000 CM4 Interrupt Vector Table | 102 |
| Table 4-2: MAX78000 RV32 Interrupt Vector Table | 104 |

| | |
|---|-----|
| Table 5-1: MAX78000 GPIO Pin Count | 106 |
| Table 5-2: MAX78000 GPIO Pin Function Configuration | 113 |
| Table 5-3: MAX78000 Input Mode Configuration | 113 |
| Table 5-4: MAX78000 Output Mode Configuration | 114 |
| Table 5-5: MAX78000 GPIO Port Interrupt Vector Mapping | 114 |
| Table 5-6: MAX78000 GPIO Alternate Function Configuration Reference | 115 |
| Table 5-7: MAX78000 GPIO Output/Input Configuration Reference | 115 |
| Table 5-8: MAX78000 GPIO Interrupt Configuration Reference | 115 |
| Table 5-9: MAX78000 GPIO Pullup/Pulldown/Drive Strength/Voltage Configuration Reference | 115 |
| Table 5-10: MAX78000 GPIO Wakeup Interrupt Vector | 118 |
| Table 5-11: GPIO Register Summary | 119 |
| Table 5-12: GPIO Port n Configuration Enable Bit 0 Register | 120 |
| Table 5-13: GPIO Port n Configuration Enable Atomic Set Bit 0 Register | 120 |
| Table 5-14: GPIO Port n Configuration Enable Atomic Clear Bit 0 Register | 120 |
| Table 5-15: GPIO Port n Output Enable Register | 120 |
| Table 5-16: GPIO Port n Output Enable Atomic Set Register | 121 |
| Table 5-17: GPIO Port n Output Enable Atomic Clear Register | 121 |
| Table 5-18: GPIO Port n Output Register | 121 |
| Table 5-19: GPIO Port n Output Atomic Set Register | 121 |
| Table 5-20: GPIO Port n Output Atomic Clear Register | 121 |
| Table 5-21: GPIO Port n Input Register | 122 |
| Table 5-22: GPIO Port n Interrupt Mode Register | 122 |
| Table 5-23: GPIO Port n Interrupt Polarity Register | 122 |
| Table 5-24: GPIO Port n Input Enable Register | 122 |
| Table 5-25: GPIO Port n Interrupt Enable Register | 123 |
| Table 5-26: GPIO Port n Interrupt Enable Atomic Set Register | 123 |
| Table 5-27: GPIO Port n Interrupt Enable Atomic Clear Register | 123 |
| Table 5-28: GPIO Port n Interrupt Status Register | 123 |
| Table 5-29: GPIO Port n Interrupt Clear Register | 123 |
| Table 5-30: GPIO Port n Wakeup Enable Register | 124 |
| Table 5-31: GPIO Port n Wakeup Enable Atomic Set Register | 124 |
| Table 5-32: GPIO Port n Wakeup Enable Atomic Clear Register | 124 |
| Table 5-33: GPIO Port n Interrupt Dual Edge Mode Register | 124 |
| Table 5-34: GPIO Port n Pad Configuration 1 Register | 124 |
| Table 5-35: GPIO Port n Pad Configuration 2 Register | 125 |
| Table 5-36: GPIO Port n Configuration Enable Bit 1 Register | 125 |
| Table 5-37: GPIO Port n Configuration Enable Atomic Set Bit 1 Register | 125 |
| Table 5-38: GPIO Port n Configuration Enable Atomic Clear Bit 1 Register | 125 |
| Table 5-39: GPIO Port n Configuration Enable Bit 2 Register | 126 |
| Table 5-40: GPIO Port n Configuration Enable Atomic Set Bit 2 Register | 126 |
| Table 5-41: GPIO Port n Configuration Enable Atomic Clear Bit 2 Register | 126 |
| Table 5-42: GPIO Port n Output Drive Strength Bit 0 Register | 126 |
| Table 5-43: GPIO Port n Output Drive Strength Bit 1 Register | 126 |
| Table 5-44: GPIO Port n Pulldown/Pullup Strength Select Register | 127 |
| Table 5-45: GPIO Port n Voltage Select Register | 127 |
| Table 6-1. MAX78000 Internal Flash Memory Organization | 128 |
| Table 6-2: Valid Addresses Flash Writes | 129 |
| Table 6-3: Flash Controller Register Summary | 131 |
| Table 6-4: Flash Controller Address Pointer Register | 132 |
| Table 6-5: Flash Controller Clock Divisor Register | 132 |
| Table 6-6: Flash Controller Control Register | 132 |
| Table 6-7: Flash Controller Interrupt Register | 133 |
| Table 6-8: Flash Controller Data 0 Register | 134 |

| | |
|--|-----|
| Table 6-9: Flash Controller Data Register 1 | 134 |
| Table 6-10: Flash Controller Data Register 2 | 134 |
| Table 6-11: Flash Controller Data Register 3 | 134 |
| Table 6-12: Flash Controller Access Control Register | 135 |
| Table 6-13: Flash Write/Lock 0 Register | 135 |
| Table 6-14: Flash Write/Lock 1 Register | 135 |
| Table 6-15: Flash Read Lock 0 Register | 135 |
| Table 6-16: Flash Read Lock 1 Register | 136 |
| Table 7-1: MAX78000 DAP Instances..... | 137 |
| Table 8-1: MAX78000 Semaphore Instances..... | 138 |
| Table 8-2: Semaphore Register Summary | 139 |
| Table 8-3: Semaphore 0 Register..... | 139 |
| Table 8-4: Semaphore 1 Register..... | 139 |
| Table 8-5: Semaphore 2 Register..... | 140 |
| Table 8-6: Semaphore 3 Register..... | 140 |
| Table 8-7: Semaphore 4 Register..... | 140 |
| Table 8-8: Semaphore 5 Register..... | 140 |
| Table 8-9: Semaphore 6 Register..... | 141 |
| Table 8-10: Semaphore 7 Register | 141 |
| Table 8-11: Semaphore IRQ 0 Register | 141 |
| Table 8-12: Semaphore Mailbox 0 Register | 142 |
| Table 8-13: Semaphore IRQ 1 Register | 142 |
| Table 8-14: Semaphore Mailbox 1 Register | 142 |
| Table 8-15: Semaphore Status Register..... | 143 |
| Table 9-1: MAX78000 DMA and Channel Instances | 144 |
| Table 9-2: DMA Source and Destination by Peripheral | 145 |
| Table 9-3: Data Movement from Source to DMA FIFO..... | 146 |
| Table 9-4: Data Movement from the DMA FIFO to Destination | 147 |
| Table 9-5: DMA Channel Timeout Configuration..... | 151 |
| Table 9-6: DMA Register Summary | 152 |
| Table 9-7: DMAm Interrupt Flag Register | 152 |
| Table 9-8: DMAm Interrupt Enable Register | 152 |
| Table 9-9: Standard DMA Channel 0 to Channel 3 Register Summary | 153 |
| Table 9-10: DMA Channel Registers Summary | 153 |
| Table 9-11: DMA_CH n Control Register..... | 153 |
| Table 9-12: DMA Status Register | 156 |
| Table 9-13: DMA Channel n Source Register | 157 |
| Table 9-14: DMA Channel n Destination Register | 157 |
| Table 9-15: DMA Channel n Count Register | 157 |
| Table 9-16: DMA Channel n Source Reload Register | 157 |
| Table 9-17: DMA Channel n Destination Reload Register | 158 |
| Table 9-18: DMA Channel n Count Reload Register | 158 |
| Table 10-1: MAX78000 ADC Peripheral Pins..... | 159 |
| Table 10-2: ADC Clock Frequency and ADC Conversion Time ($f_{SYSCLK} = 100MHz$, $f_{PCLK} = 50MHz$) | 162 |
| Table 10-3: Input and Reference Scale Support by ADC Input Channel | 164 |
| Table 10-4: ADC Data Register Alignment Options..... | 164 |
| Table 10-5. ADC Registers Summary..... | 169 |
| Table 10-6: ADC Control Register | 169 |
| Table 10-7: ADC Status Register | 171 |
| Table 10-8: ADC Data Register | 171 |
| Table 10-9: ADC Interrupt Control Register | 171 |
| Table 10-10: ADC Limit 0 to 3 Registers..... | 172 |
| Table 10-11. Low-Power Comparator Registers Summary | 173 |

| | |
|--|-----|
| Table 10-12: Low-Power Comparator n Registers | 173 |
| Table 11-1: MAX78000 UART/LPUART Instances | 177 |
| Table 11-2: MAX78000 Interrupt Events | 179 |
| Table 11-3: Frame Error Detection (FDM not present, or FDM = 0 and DPFE = 0) | 179 |
| Table 11-4: Frame Error Detection (FDM = 1 and DPFE = 1) | 180 |
| Table 11-5: MAX78000 Wakeup Events..... | 181 |
| Table 11-6: Slow Baud Rate Generation Example (FDM=1) | 184 |
| Table 11-7: UART/LPUART Register Summary..... | 186 |
| Table 11-8: UART Control Register | 187 |
| Table 11-9: UART Status Register | 189 |
| Table 11-10: UART Interrupt Enable Register..... | 190 |
| Table 11-11: UART Interrupt Flag Register | 190 |
| Table 11-12: UART Clock Divisor Register..... | 191 |
| Table 11-13: UART Oversampling Control Register | 191 |
| Table 11-14: UART Transmit FIFO Register | 191 |
| Table 11-15: UART Pin Control Register | 192 |
| Table 11-16: UART Data Register..... | 192 |
| Table 11-17: UART DMA Register | 192 |
| Table 11-18: UART Wakeup Enable | 193 |
| Table 11-19. UART Wakeup Flag Register..... | 193 |
| Table 12-1: MAX78000 SPI Instances..... | 195 |
| Table 12-2: MAX78000 SPI Peripheral Pins..... | 196 |
| Table 12-3: Four-Wire Format Signals | 196 |
| Table 12-4: Three-Wire Format Signals | 197 |
| Table 12-5: SPI Modes Clock Phase and Polarity Operation..... | 201 |
| Table 12-6: SPIn Register Summary | 202 |
| Table 12-7: SPIn FIFO32 Register | 203 |
| Table 12-8: SPIn 16-bit FIFO Register..... | 203 |
| Table 12-9: SPIn 8-bit FIFO Register..... | 203 |
| Table 12-10: SPIn Control 0 Register | 203 |
| Table 12-11: SPIn Control 1 Register | 205 |
| Table 12-12: SPIn Control 2 Register | 205 |
| Table 12-13: SPIn Slave Select Timing Register..... | 207 |
| Table 12-14: SPIn Master Clock Configuration Registers..... | 207 |
| Table 12-15: SPIn DMA Control Registers..... | 208 |
| Table 12-16: SPIn Interrupt Status Flags Registers | 209 |
| Table 12-17: SPIn Interrupt Enable Registers | 210 |
| Table 12-18: SPIn Wakeup Status Flags Registers | 212 |
| Table 12-19: SPIn Wakeup Enable Registers..... | 212 |
| Table 12-20: SPIn Slave Select Timing Registers | 212 |
| Table 13-1: MAX78000 I ² C Peripheral Pins | 214 |
| Table 13-2: I ² C Bus Terminology | 215 |
| Table 13-3: Calculated I ² C Bus Clock Frequencies | 219 |
| Table 13-4: I ² C Slave Address Format | 219 |
| Table 13-5: I ² C Register Summary | 232 |
| Table 13-6: I ² C Control Register | 232 |
| Table 13-7: I ² C Status Register..... | 234 |
| Table 13-8: I ² C Interrupt Flag 0 Register..... | 235 |
| Table 13-9: I ² C Interrupt Enable 0 Register | 237 |
| Table 13-10: I ² C Interrupt Flag 1 Register | 238 |
| Table 13-11: I ² C Interrupt Enable 1 Register | 239 |
| Table 13-12: I ² C FIFO Length Register..... | 239 |
| Table 13-13: I ² C Receive Control 0 Register..... | 240 |

| | |
|---|-----|
| Table 13-14: I ² C Receive Control 1 Register..... | 240 |
| Table 13-15: I ² C Transmit Control 0 Register..... | 241 |
| Table 13-16: I ² C Transmit Control 1 Register..... | 242 |
| Table 13-17: I ² C Data Register | 243 |
| Table 13-18: I ² C Master Control Register | 243 |
| Table 13-19: I ² C SCL Low Control Register..... | 244 |
| Table 13-20: I ² C SCL High Control Register | 244 |
| Table 13-21: I ² C Hs-Mode Clock Control Register..... | 245 |
| Table 13-22: I ² C Timeout Register | 245 |
| Table 13-23: I ² C DMA Register..... | 245 |
| Table 13-24: I ² C Slave Address Register..... | 246 |
| Table 14-1: MAX78000 I ² S Instances | 247 |
| Table 14-2: MAX78000 I ² S Pin Mapping | 247 |
| Table 14-3: I ² S Interrupt Events (Channel 0 Example) | 251 |
| Table 14-4: I ² S Channel mode Configuration (Channel 0 Example) | 252 |
| Table 14-5: Byte Ordering when I2Sn_CTRL0CH0.wsize = 0 (Byte..... | 253 |
| Table 14-6: Half-Word Ordering when I2Sn_CTRL0CH0.wsize = 1 | 254 |
| Table 14-7: Word Ordering when I2Sn_CTRL0CH0.wsize = 2 or 3 | 254 |
| Table 14-8: Number of BCLK per Half Frame and Sample Number | 254 |
| Table 14-9: I ² S Register Summary | 256 |
| Table 14-10: Global Mode Control Channel 0 | 256 |
| Table 14-11: Local Setup Channel 0 Register..... | 257 |
| Table 14-12: I ² S DMA Channel 0 Register | 258 |
| Table 14-13: I ² S FIFO Channel 0 Register | 259 |
| Table 14-14: I ² S Interrupt Flag Register | 259 |
| Table 14-15: I ² S Interrupt Enable Register | 259 |
| Table 14-16: I ² S External Setup Register..... | 260 |
| Table 15-1: MAX78000 PCIF Instances | 261 |
| Table 15-2: MAX78000 PCIF Signals..... | 261 |
| Table 15-3: Parallel Camera Interface Register Summary | 265 |
| Table 15-4: PCIF Configuration Register | 265 |
| Table 15-5: PCIF Interrupt Enable Register | 266 |
| Table 15-6: PCIF Status Flags Register | 267 |
| Table 15-7: PCIF Timing Codes Register | 268 |
| Table 15-8: PCIF FIFO Data Register | 268 |
| Table 16-1: MAX78000 1-Wire Master Peripheral Pins | 270 |
| Table 16-2: 1-Wire ROM Commands | 274 |
| Table 16-3: 1-Wire Slave Device ROM ID Field | 275 |
| Table 16-4: OWM Register Summary | 279 |
| Table 16-5: OWM Configuration Register..... | 279 |
| Table 16-6: OWM Clock Divisor Register | 280 |
| Table 16-7: OWM Control Status Register | 281 |
| Table 16-8: OWM Data Buffer Register | 281 |
| Table 16-9: OWM Interrupt Flag Register..... | 282 |
| Table 16-10: OWM Interrupt Enable Register | 282 |
| Table 17-1: MAX78000 RTC Counter and Alarm Registers | 285 |
| Table 17-2: RTC Register Access | 285 |
| Table 17-3: MAX78000 RTC Square Wave Output Configuration..... | 288 |
| Table 17-4: RTC Register Summary | 290 |
| Table 17-5: RTC Seconds Counter Register | 290 |
| Table 17-6: RTC Sub-Second Counter Register (12-bit) | 291 |
| Table 17-7: RTC Time-of-Day Alarm Register..... | 291 |
| Table 17-8: RTC Sub-Second Alarm Register..... | 291 |

| | |
|---|-----|
| Table 17-9: RTC Control Register | 291 |
| Table 17-10: RTC 32KHz Oscillator Digital Trim Register | 293 |
| Table 17-11: RTC 32KHz Oscillator Control Register | 294 |
| Table 18-1: MAX78000 TMR/LPTMR Instances | 296 |
| Table 18-2: MAX78000 I/O Signal Configuration | 296 |
| Table 18-3: TimerA/TimerB 32-Bit Field Allocations | 297 |
| Table 18-4: MAX78000 Wakeup Events | 300 |
| Table 18-5: MAX78000 Operating Mode Signals, TMR0/TMR1 | 303 |
| Table 18-6: MAX78000 Operating Mode Signals, TMR2/TMR3 | 305 |
| Table 18-7: MAX78000 Operating Mode Signals, LPTMR0/LPTMR1 | 307 |
| Table 18-8: Timer Register Summary | 323 |
| Table 18-9: Timer Count Register | 323 |
| Table 18-10: Timer Compare Register | 323 |
| Table 18-11: Timer PWM Register | 324 |
| Table 18-12: Timer Interrupt Register | 324 |
| Table 18-13: Timer Control 0 Register | 325 |
| Table 18-14: Timer Non-Overlapping Compare Register | 328 |
| Table 18-15: Timer Control 1 Register | 328 |
| Table 18-16: Timer Wakeup Status Register | 330 |
| Table 19-1: MAX78000 WUT Clock Period | 332 |
| Table 19-2: Wakeup Timer Register Summary | 337 |
| Table 19-3: Wakeup Timer Count Register | 337 |
| Table 19-4: Wakeup Timer Compare Register | 337 |
| Table 19-5: Wakeup Timer Interrupt Register | 337 |
| Table 19-6: Wakeup Timer Control Register | 338 |
| Table 20-1: MAX78000 WDT Instances Summary | 341 |
| Table 20-2: MAX78000 WDT Event Summary | 343 |
| Table 20-3: WDT Register Summary | 347 |
| Table 20-4: WDT Control Register | 347 |
| Table 20-5: WDT Reset Register | 350 |
| Table 20-6: WDT Clock Source Select Register | 350 |
| Table 20-7: WDT Count Register | 351 |
| Table 21-1: Pulse Train Engine Register Summary | 355 |
| Table 21-2: Pulse Train Engine Global Enable/Disable Register | 357 |
| Table 21-3: Pulse Train Engine Resync Register | 358 |
| Table 21-4: Pulse Train Engine Stopped Interrupt Flag Register | 361 |
| Table 21-5: Pulse Train Engine Interrupt Enable Register | 363 |
| Table 21-6: Pulse Train Engine Safe Enable Register | 365 |
| Table 21-7: Pulse Train Engine Safe Disable Register | 367 |
| Table 21-8: Pulse Train Engine Configuration Register | 368 |
| Table 21-9: Pulse Train Mode Bit Pattern Register | 369 |
| Table 21-10: Pulse Train n Loop Configuration Register | 369 |
| Table 21-11: Pulse Train n Automatic Restart Configuration Register | 369 |
| Table 22-1: MAX78000 CRC Instances | 371 |
| Table 22-2: Organization of Calculated Result in CRC_VAL.value | 372 |
| Table 22-3: Common CRC Polynomials | 372 |
| Table 22-4: CRC Register Summary | 374 |
| Table 22-5: CRC Control Register | 374 |
| Table 22-6: CRC Data Input Register | 375 |
| Table 22-7: CRC Polynomial Register | 375 |
| Table 22-8: CRC Value Register | 375 |
| Table 23-1: MAX78000 AES Instances | 376 |
| Table 23-2: MAX78000 Interrupt Events | 378 |

| | |
|--|-----|
| Table 23-3: AES Register Summary | 379 |
| Table 23-4: AES Control Register | 379 |
| Table 23-5: AES Status Register | 380 |
| Table 23-6: AES Interrupt Flag Register | 381 |
| Table 23-7: AES Interrupt Enable Register | 381 |
| Table 23-8: AES FIFO Register | 381 |
| Table 24-1: TRNG Register Summary | 382 |
| Table 24-2: TRNG Control Register | 382 |
| Table 24-3: TRNG Status Register | 382 |
| Table 24-4: TRNG Data Register | 383 |
| Table 25-1: MAX78000 Bootloader Instances | 384 |
| Table 25-2: Bootloader Operating States and Prompts | 385 |
| Table 25-3: PERMLOCK Command Summary | 385 |
| Table 25-4: CHALLENGE Command Summary | 389 |
| Table 25-5: MAX78000 General Command Summary | 390 |
| Table 25-6: L – Load | 390 |
| Table 25-7: P – Page Erase | 390 |
| Table 25-8: V – Verify | 391 |
| Table 25-9: LOCK – Lock Device | 391 |
| Table 25-10: PLOCK – Permanent Lock | 391 |
| Table 25-11: UNLOCK – Unlock Device | 392 |
| Table 25-12: H – Check Device | 392 |
| Table 25-13: I – Get ID | 392 |
| Table 25-14: S – Status | 393 |
| Table 25-15: Q – Quit | 393 |
| Table 25-16: MAX78000 Secure Command Summary | 394 |
| Table 25-17: LK – Load Application Key | 394 |
| Table 25-18: LK – Load Challenge Key | 394 |
| Table 25-19: VK – Verify Application Key | 394 |
| Table 25-20: VC – Verify Challenge Key | 395 |
| Table 25-21: AK – Activate Application Key | 395 |
| Table 25-22: AC – Activate Challenge Key | 395 |
| Table 25-23: WL – Write Code Length | 396 |
| Table 25-24: MAX78000 Challenge/Response Command Summary | 397 |
| Table 25-25: GC – Get Challenge | 397 |
| Table 25-26: SR – Send Response | 397 |
| Table 26-1: CNN Memory Configuration (APB Accessible) for the Quad CNNx16n | 404 |
| Table 26-2: CNNx16 Processor Array 0 TRAM Mapping Details (APB Accessible) | 405 |
| Table 26-3: CNNx16 Processor Array 1 TRAM Mapping Details (APB Accessible) | 405 |
| Table 26-4: CNNx16 Processor Array 2 TRAM Mapping Details (APB Accessible) | 407 |
| Table 26-5: CNNx16 Processor Array 3 TRAM Mapping Details (APB Accessible) | 407 |
| Table 26-6: CNNx16 Processor Array 0 MRAM Mapping Details (APB Accessible) | 410 |
| Table 26-7: CNNx16 Processor Array 1 Mask RAM Mapping Details (APB Accessible) | 411 |
| Table 26-8: CNNx16 Processor Array 2 Mask RAM Mapping Details (APB Accessible) | 411 |
| Table 26-9: CNNx16 Processor Array3 Mask RAM Mapping Details (APB Accessible) | 412 |
| Table 26-10: Global CNN Register Summary | 412 |
| Table 26-11: CNN FIFO Control Register | 413 |
| Table 26-12: CNN FIFO Status Register | 414 |
| Table 26-13: CNN FIFO 0 Write Register | 415 |
| Table 26-14: CNN FIFO 1 Write Register | 415 |
| Table 26-15: CNN FIFO 2 Write Register | 415 |
| Table 26-16: CNN FIFO 3 Write Register | 415 |
| Table 26-17: CNN Always On Domain Control Register | 415 |

| | |
|---|-----|
| Table 26-18: CNNx16_n Instances and Base Offset Address | 417 |
| Table 26-19: CNNx16_n Processor Array Registers | 417 |
| Table 26-20: CNNx16_n Control Register | 418 |
| Table 26-21: CNNx16_n SRAM Control Register | 421 |
| Table 26-22: CNNx16_n Layer Count Maximum Register | 423 |
| Table 26-23: CNNx16_n SRAM Test Register | 424 |
| Table 26-24: CNNx16_n_Ly Row Count Register | 428 |
| Table 26-25: CNNx16_n_Ly Column Count Register | 429 |
| Table 26-26: CNNx16_n_Ly One Dimensional Control Register | 429 |
| Table 26-27: CNNx16_n_Ly Pool Row Count Register | 430 |
| Table 26-28: CNNx16_n_Ly Pool Column Count Register | 431 |
| Table 26-29: CNNx16_n_Ly Stride Count Register | 431 |
| Table 26-30: CNNx16_n_Ly Write Pointer Base Address Register | 431 |
| Table 26-31: CNNx16_n_Ly Write Pointer Timeslot Offset Register | 431 |
| Table 26-32: CNNx16_n_Ly Write Pointer Mask Offset Register | 432 |
| Table 26-33: CNNx16_n_Ly Write Pointer Multi-Pass Channel Offset Register | 432 |
| Table 26-34: CNNx16_n_Ly Read Pointer Base Address Register | 433 |
| Table 26-35: CNNx16_n_Ly Layer Control 0 Register | 433 |
| Table 26-36: CNNx16_n_Ly Layer Mask Count Register | 435 |
| Table 26-37: CNNx16_n_Ly TRAM Pointer Register | 435 |
| Table 26-38: CNNx16_n_Ly Enable Register | 436 |
| Table 26-39: CNNx16_n_Ly Post Processing Register | 436 |
| Table 26-40: CNNx16_n_Ly Layer Control 1 Register | 437 |
| Table 26-41: CNNx16_n_Muchselerator Data Register | 438 |
| Table 26-42: CNNx16_n_Sz Stream Control 0 Register | 439 |
| Table 26-43: CNNx16_n_Sz Stream Control 1 Register | 439 |
| Table 26-44: CNNx16_n_Sz Stream Frame Buffer Size Register | 440 |
| Table 26-45: CNNx16_n Input FIFO Frame Size Register | 440 |
| Table 27-1: Revision History | 442 |

1. Overview

The MAX78000 is a new breed of low-power microcontrollers built to thrive in the rapidly evolving AI at the edge market. These products include Maxim's proven ultra-low power MCU IP along with deep neural network AI acceleration.

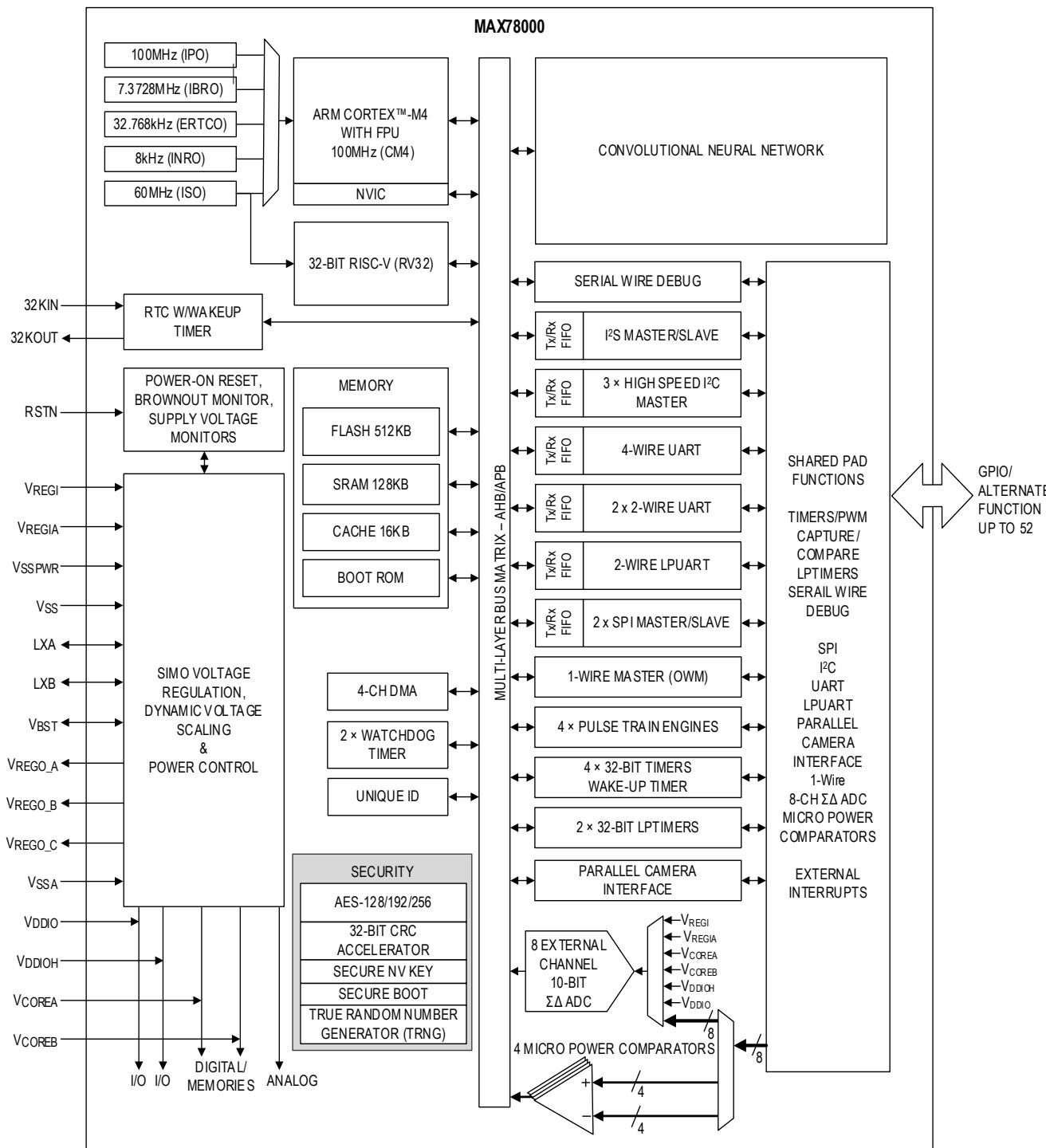
The MAX78000 is an advanced system-on-chip featuring an Arm® Cortex®-M4 with FPU CPU for efficient computation of complex functions and algorithms with integrated power management. It also includes 442KB weight CNN accelerator. The devices offer large on-chip memory with 512KB flash and up to 128KB SRAM. Multiple high-speed and low-power communications interfaces are supported including high-speed SPI, I2C serial interface, and LPUART. Additional low-power peripherals include flexible LPTMR and analog comparators.

The device is available in 81-CTBGA, 8mm × 8mm, 0.8mm pitch.

The high-level block diagram for the MAX78000 is shown in [Figure 1-1](#).

1.1 Block Diagram

Figure 1-1: MAX78000 Block Diagram



2. Memory, Register Mapping, and Access

2.1 Memory, Register Mapping, and Access Overview

The Arm Cortex-M4 architecture defines a standard memory space for unified code and data access. This memory space is addressed in units of single bytes but is most typically accessed in 32-bit (4 byte) units. It may also be accessed, depending on the implementation, in 8-bit (1 byte) or 16-bit (2 byte) widths. The total range of the memory space is 32 bits wide (4GB addressable total), from addresses 0x0000 0000 to 0xFFFF FFFF.

It is important to note, however, that the architectural definition does not require the entire 4GB memory range to be populated with addressable memory instances.

Figure 2-1: CM4 Code Memory Mapping

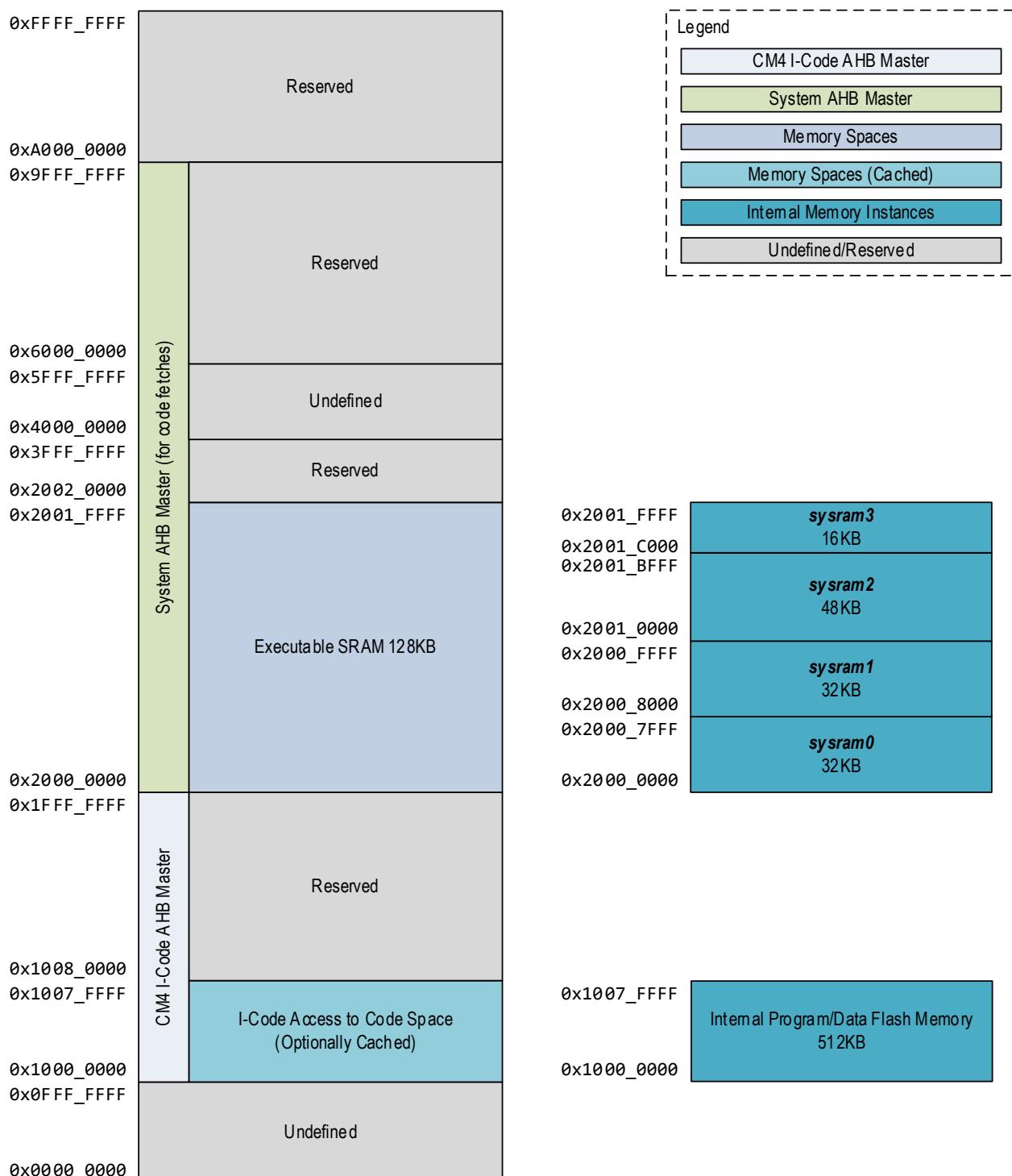


Figure 2-2: RISC-V IBUS Code Memory Mapping

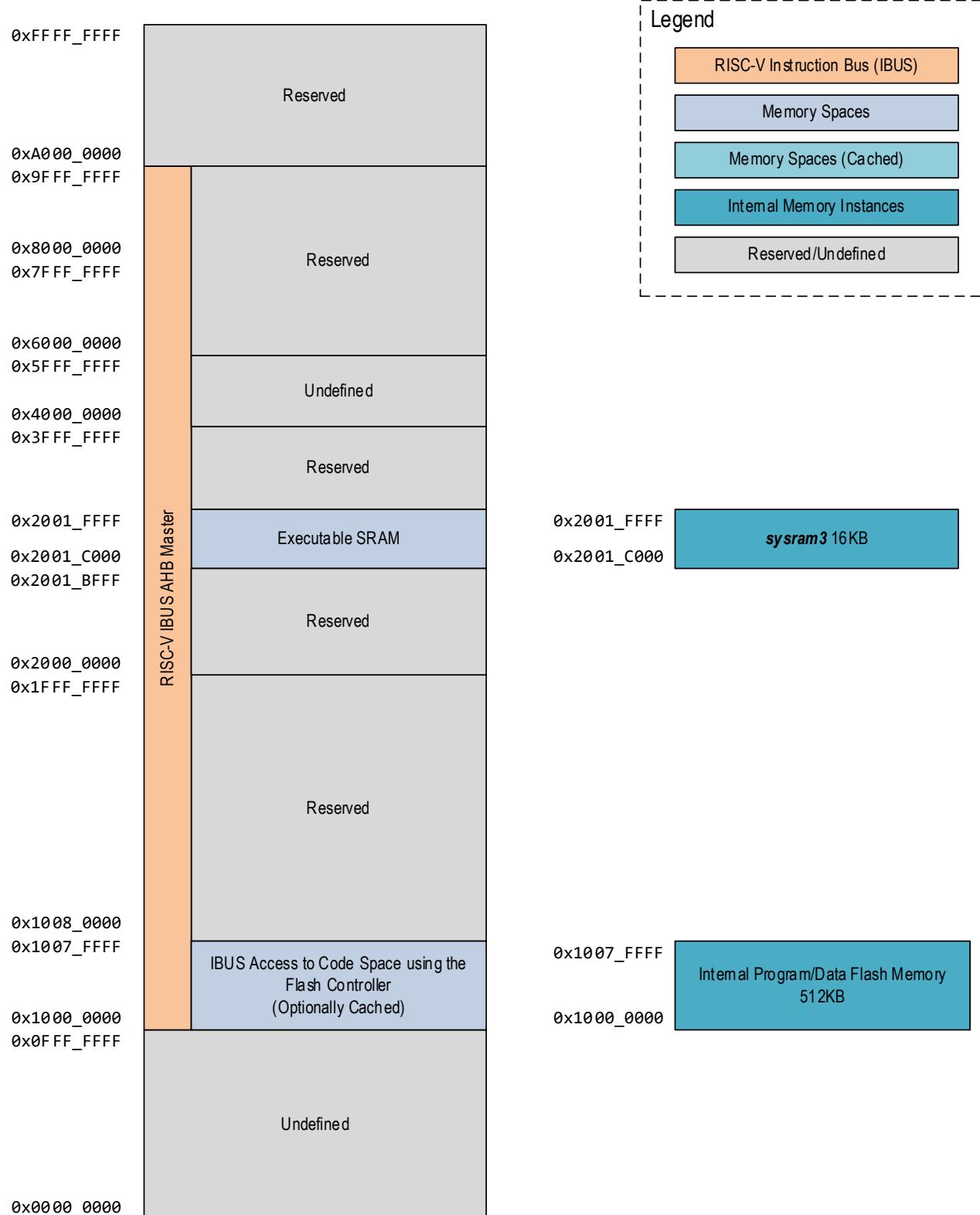


Figure 2-3: CM4 Peripheral and Data Memory Mapping

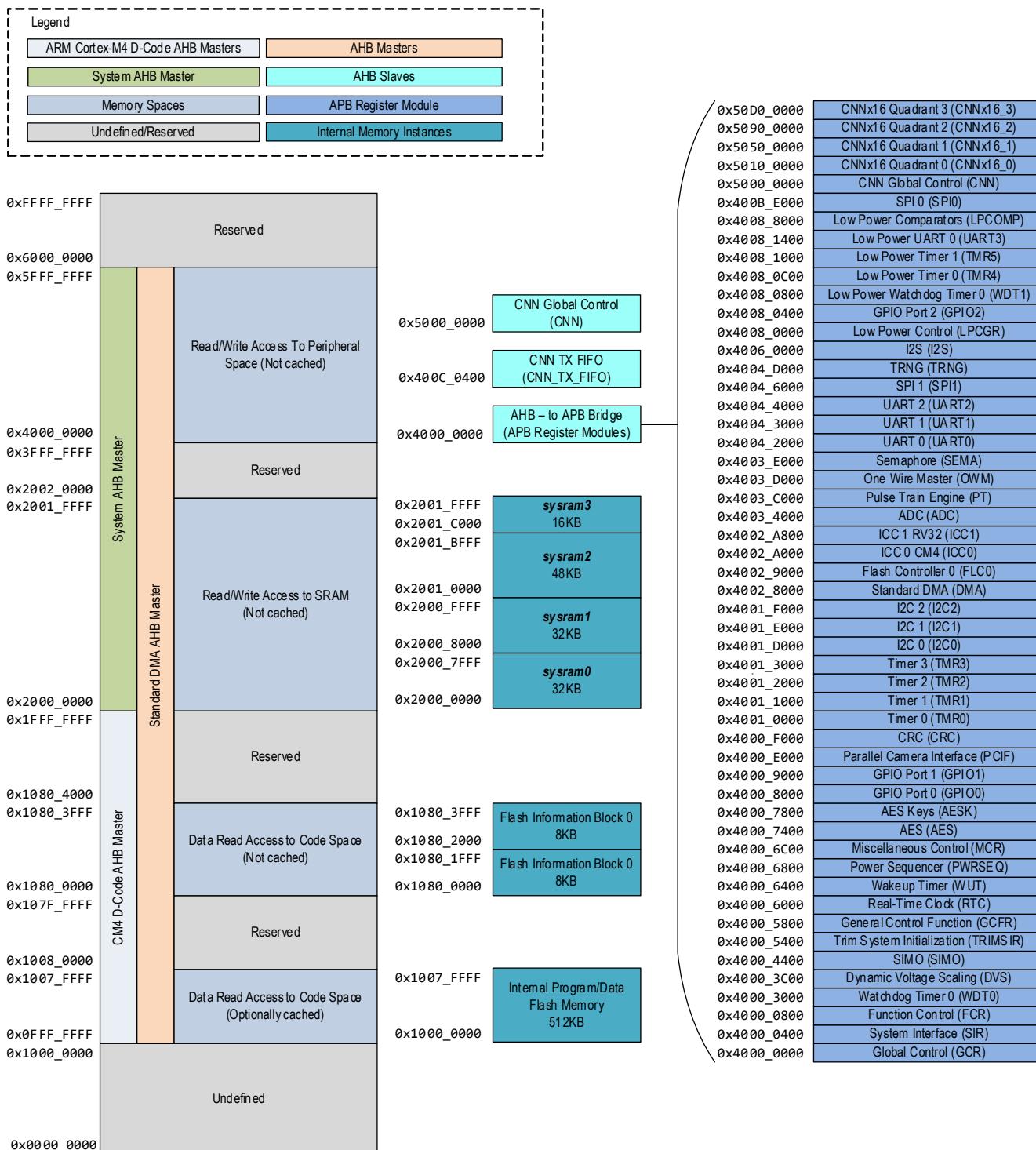
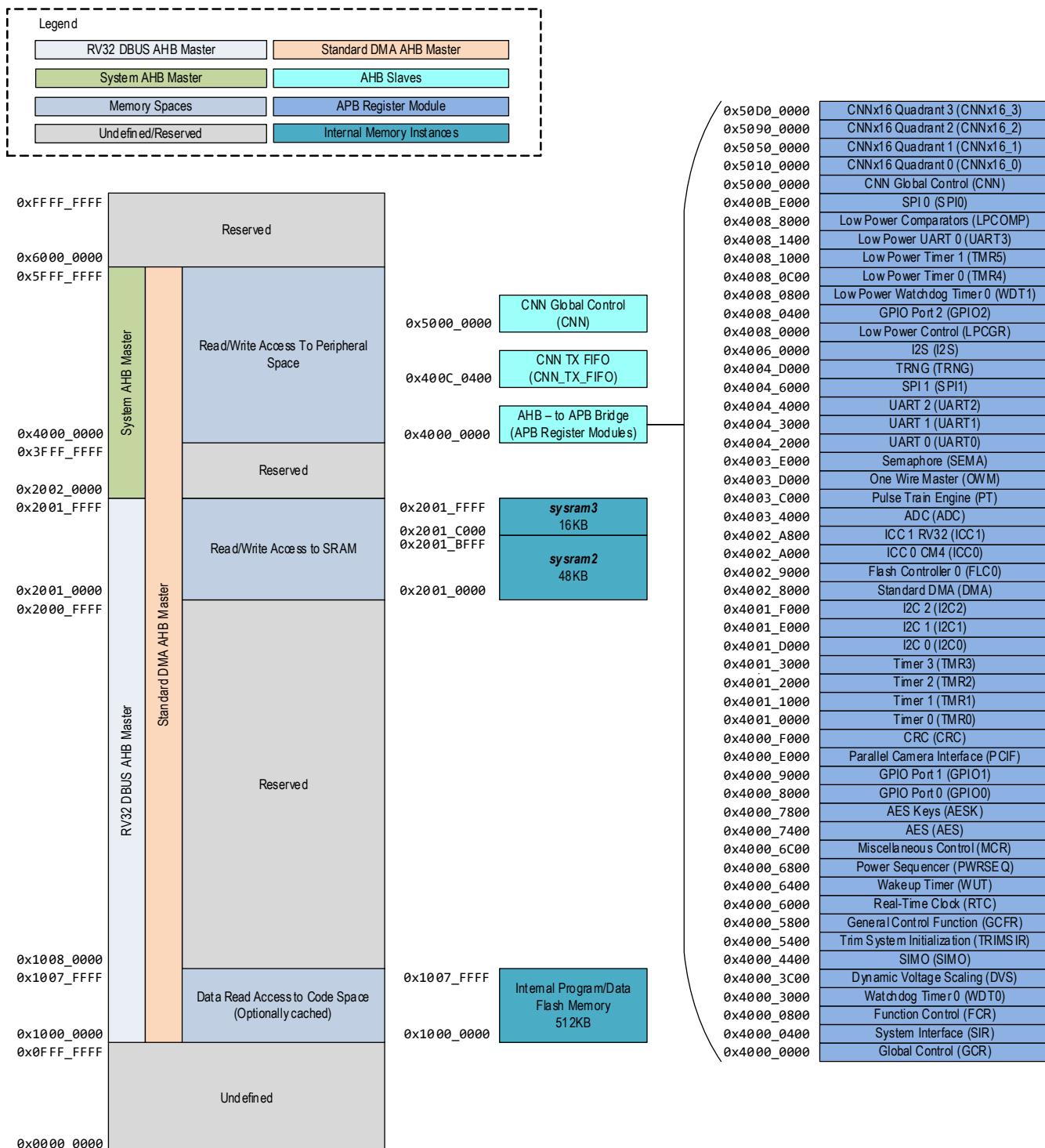


Figure 2-4: RV32 Peripheral and Data Memory Mapping



2.2 Field Access Definitions

All the fields that are accessible by user software have distinct access capabilities. Each register table contained in this user guide has an access type defined for each field. The definition of each field access type is presented in [Table 2-1: Field Access Definitions](#).

Table 2-1: Field Access Definitions

| Access Type | Definition |
|-------------|---|
| RO | Reserved This access type is reserved for static fields. Reads of this field will return the reset value. Writes are ignored. |
| DNM | Reserved. Do Not Modify Reads of this field will return indeterminate values. Software must first read this field and write the same value whenever writing to this register. |
| R | Read Only Reads of this field will return a value. Writes to the field have no effect on device operation. |
| W | Write Only Reads of this field will return indeterminate values. Writes to the field may change the field's state and may affect device operation. |
| R/W | Unrestricted Read/Write Reads of this field will return a value. Writes to the field may change the field's state and may affect device operation. |
| RC | Read-to-Clear Reads of this field may return a value. Any read of this register will clear the field to 0. Writes to the field have no effect on device operation. |
| RS | Read-to-Set Reads of this field may return a value. Any read of this register will set the field to 1. Writes to the field have no effect on device operation. |
| R/W0 | Read-Write-0-Only Reads of this field may return a value. Writing 0 to this field may change the field's state and may affect device operation. Writing 1 to the field has no effect on device operation. |
| R/W1 | Read-Write-1-Only Reads of this field may return a value. Writing 1 to this field may change the field's state and may affect device operation. Writing 0 to the field has no effect on device operation. |
| R/W1C | Read-Write-1-to-Clear Reads of this field may return a value. Writing 1 to this field will clear this field to 0. Writing 0 to the field has no effect on device operation. |
| W1C | Write-1-to-Clear Reads of this field will return indeterminate values. Writing 1 to this field will clear this field to 0. Writing 0 to the field has no effect on device operation. |
| R/W0S | Read-Write-0-to-Set Reads of this field may return a value. Writing 0 to this field will set this field to 1. Writing 1 to the field has no effect on device operation. |

2.3 Standard Memory Regions

Several standard memory regions are defined for the Arm Cortex-M4 and RISC-V architectures; the use of many of these is optional for the system integrator. At a minimum, the MAX78000 must contain some code and data memory for application code and variable/stack use for CPU0, as well as certain components which are part of the instantiated Cortex-M4 core.

2.3.1 Code Space

The code space area of memory is designed to contain the primary memory used for code execution by the device. This memory area is defined from byte address range 0x0000 0000 to 0x1FFF FFFF (0.5GB maximum). Two different standard core bus masters are used by the Cortex-M4 core and Arm debugger to access this memory area. The I-Code AHB bus master is used for instruction decode fetching from code memory, while the D-Code AHB bus master is used for data fetches from code memory. This is arranged so that data fetches avoid interfering with instruction execution. Additionally, the RV32 uses the D-BUS to access code memory in this area and the I-Bus to access data fetches from the code memory.

The MAX78000 code memory mapping is illustrated in [Figure 2-1: CM4 Code Memory Mapping](#) and [Figure 2-2: RISC-V IBUS Code Memory Mapping](#). The code space memory area contains the main internal flash memory, which holds most of the instruction code that will be executed on the device. The internal flash memory is mapped into both code and data space from 0x1000 0000 to 0x1007 FFFF. The main program flash memory is 512KB and consists of 64 logical pages of 8,192 Bytes per page.

This program memory area must also contain the default system vector table and the initial settings for all system exception handlers and interrupt handlers for the CM4 core. The reset vector for the device is 0x0000 0000 and contains the device ROM code which transfers execution to user code at address 0x1000 0000.

The code space memory on the MAX78000 also contains the mapping for the flash information block, from 0x1080 0000 to 0x1080 3FFF. However, this mapping is generally only present during Maxim Integrated production test; it is disabled once the information block has been loaded with valid data and the info block lockout option has been set. This memory is accessible for data reads only and cannot be used for code execution. The flash information block is user read only accessible and contains the Unique Serial Number (USN).

2.3.2 Instruction Cache Memory

The MAX78000 includes a dedicated Unified Instruction Cache Controller with 16,384 Bytes of cache memory for the CM4 core (ICC0). Optionally, [sysram3](#) can be used as a Unified Instruction Cache Controller for the RV32 (ICC1).

The Unified Instruction Cache Memory is used to cache data and instructions fetched via the I-Code bus for the CM4 or the IBUS for the RV32 from the internal flash memory.

Refer to section [Unified Internal Cache Controller](#) for detailed instructions on enabling the Unified Internal Cache Controllers.

2.3.3 Information Block Flash Memory

The information block is a separate area of the Internal Flash Memory and is 16,384 Bytes. The information block is used to store trim settings (option configuration and analog trim) as well as other nonvolatile device-specific information. The information block also contains the device's Unique Serial Number (USN). The USN is a 104-bit field. USN bits 0 thru 7 contain the die revision.

Figure 2-5: Unique Serial Number Format

| | | Bit Position | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---------|------------|------------------|------------------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | |
| Address | 0x10800000 | USN bits 16 - 0 | | | | | | | | | | | | | | | | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | | | |
| | 0x10800004 | x | USN bits 47 - 17 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 0x10800008 | USN bits 64 - 48 | | | | | | | | | | | | | | | | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | | | | |
| | 0x1080000C | x | USN bits 95 - 65 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 0x10800010 | x | x | x | x | x | x | x | x | x | USN bits 103 - 96 | | | | | | | | | | | | | | | | x | x | x | x | x | x | x | x | x | x |

Reading the USN requires the information block to be unlocked. Unlocking the information block does not enable write access to the block but allows the contents of the USN to be read from the block. Unlock the information block using the following steps:

1. Write 0x3A7F 5CA3 to [*FLCn_ACNTL*](#).
2. Write 0xA1E3 4F20 to [*FLCn_ACNTL*](#).
3. Write 0x0x9608 B2C1 to [*FLCn_ACNTL*](#).
4. Information block is now read-only accessible.

To re-lock the information block to prevent access, simply write any 32-bit word to [*FLCn_ACNTL*](#).

2.3.4 SRAM Space

The SRAM area of memory is intended to contain the primary SRAM data memory of the device and is defined from byte address range 0x2000 0000 to 0x3FFF FFFF (0.5GB maximum). This memory can be used for general purpose variable and data storage, code execution, and the CM4 stack as well as the RV32 stack.

The MAX78000 CM4's data memory mapping is illustrated in Figure 2-1: CM4 Code Memory Mapping. The MAX78000 RV32's data memory mapping is illustrated in [*Figure 2-4: RV32 Peripheral and Data Memory Mapping*](#).

The system SRAM configuration is defined in [*Table 2-2: System SRAM Configuration*](#), additionally the CNN memory is covered in the CNN chapter, section [*Memory Configuration*](#).

The SRAM memory area contains the main system SRAM. The size of the internal general-purpose data SRAM is 128KB. The SRAM is divided into four blocks and the contiguous address range is 0x2000 0000 to 0x2001 FFFF.

The SRAM area on the MAX78000 can be used for data storage and code execution by the CM4. The RV32 is limited to use of **sysram2** and **sysram3** for code and data storage in SRAM.

*Note: After a POR, the CM4 has access to all four SRAM regions. sysram2 and sysram3 can be configured to restrict access from the CM4 to prevent unintended modifications of these SRAM instances by the CM4. Set the [*FCR_URVCTRL.memsel*](#) field to 1 to set the RV32 core as the exclusive master for sysram2 and sysram3.*

Code stored in the SRAM is accessed directly for execution (using the system bus) and is not cached. The SRAM is also where the CM4 and RV32 stack must be located, as it is the only general-purpose SRAM memory on the device capable of this function.

Table 2-2: System SRAM Configuration

| System RAM Block # | Size | Start Address | End Address | CM4 Accessible | RV32 Accessible |
|--------------------|------|---------------|-------------|----------------|----------------------|
| sysram0 | 32KB | 0x2000 0000 | 0x2000 7FFF | ✓ | No |
| sysram1 | 32KB | 0x2000 8000 | 0x2000 FFFF | ✓ | No |
| sysram2 | 48KB | 0x2001 0000 | 0x2001 BFFF | Configurable | ✓ |
| sysram3 | 16KB | 0x2001 C000 | 0x2001 FFFF | Configurable | ✓ (Optional ICC1) |

The MAX78000 specific AHB Bus Masters can access the SRAM to use as general storage or working space.

The entirety of the SRAM memory space on the MAX78000 is contained within the dedicated Arm Cortex-M4 SRAM bit-banding region from 0x2000 0000 to 0x200F FFFF (1MB maximum for bit-banding). This means that the CPU can access the entire SRAM either using standard byte/word/doubleword access or using bit-banding operations. The bit-banding mechanism allows any single bit of any given SRAM byte address location to be set, cleared, or read individually by reading from or writing to a corresponding doubleword (32-bit wide) location in the bit-banding alias area.

The alias area for the SRAM bit-banding is located beginning at 0x2200 0000 and is a total of 32MB maximum, which allows the entire 128KB bit banding area to be accessed. Each 32-bit (4 byte aligned) address location in the bit-banding alias area translates into a single bit access (read or write) in the bit-banding primary area. Reading from the location performs a single bit read, while writing either a 1 or 0 to the location performs a single bit set or clear.

Note: The Arm Cortex-M4 core translates the access in the bit-banding alias area into the appropriate read cycle (for a single bit read) or a read-modify-write cycle (for a single bit set or clear) of the bit-banding primary area. This means that bit-banding is a core function (i.e., not a function of the SRAM memory interface layer or the AHB bus layer), and thus is only applicable to accesses generated by the core itself. Reads/writes to the bit-banding alias area by other (non-Arm-core) bus masters will not trigger a bit-banding operation and will instead result in an AHB bus error.

2.3.5 Peripheral Space

The peripheral space area of memory is intended for mapping of control registers, internal buffers/working space, and other features needed for the firmware control of non-core peripherals. It is defined from byte address range 0x4000 0000 to 0x5FFF FFFF (0.5GB maximum). On the MAX78000, all device-specific module registers are mapped to this memory area, as well as any local memory buffers or FIFOs which are required by modules.

As with the SRAM region, there is a dedicated 1MB area at the bottom of this memory region (from 0x4000 0000 to 0x400F FFFF) that is used for bit-banding operations by the Arm core. Four-byte-aligned read/write operations in the peripheral bit-banding alias area (32MB in length, from 0x4200 0000 to 0x43FF FFFF) are translated by the core into read/mask/shift or read/modify/write operation sequences to the appropriate byte location in the bit-banding area.

Note: The bit-banding operation within peripheral memory space is, like bit-banding function in SRAM space, a core remapping function. As such, it is only applicable to operations performed directly by the Arm core. If another memory bus master accesses the peripheral bit-banding alias region, the bit-banding remapping operation will not take place. In this case, the bit-banding alias region will appear to be a non-implemented memory area (causing an AHB bus error).

On the MAX78000, access to the region that contains most peripheral registers (0x4000 0000 to 0x400F FFFF) goes from the AHB bus through an AHB-to-APB bridge. This allows the peripheral modules to operate on the lower power APB bus matrix. This also ensures that peripherals with slower response times do not tie up bandwidth on the AHB bus, which must necessarily have a faster response time since it handles main application instruction and data fetching.

2.3.6 AES Key and Working Space Memory

The AES key memory and working space for AES operations (including input and output parameters) are in a dedicated register file memory tied to the AES engine block. This AES memory is mapped into AHB space for rapid firmware access.

2.3.7 External RAM Space

The external RAM space area of memory is intended for use in mapping off-chip external memory and is defined from byte address range 0x6000 0000 to 0x9FFF FFFF (1GB maximum).

The MAX78000 does not implement this memory area.

2.3.8 External Device Space

The external device space area of memory is intended for use in mapping off-chip device control functions onto the AHB bus. This memory space is defined from byte address range 0xA000 0000 to 0xDFFF FFFF (1GB maximum).

The MAX78000 does not implement this memory area.

2.3.9 System Area (Private Peripheral Bus)

The system area (private peripheral bus) memory space contains register areas for functions that are only accessible by the Arm core itself (and the Arm debugger, in certain instances). It is defined from byte address range 0xE000 0000 to 0xE00F FFFF. This APB bus is restricted and can only be accessed by the Arm core and core-internal functions. It cannot be accessed by other modules which implement AHB memory masters, such as the DMA interface.

In addition to being restricted to the core, application code is only allowed to access this area when running in the privileged execution mode (as opposed to the standard user thread execution mode). This helps ensure that critical system settings controlled in this area are not altered inadvertently or by errant code that should not have access to this area.

Core functions controlled by registers mapped to this area include the SysTick timer, debug and tracing functions, the NVIC (interrupt handler) controller, and the Flash Breakpoint controller.

2.3.10 System Area (Vendor Defined)

The system area (vendor defined) memory space is reserved for vendor (system integrator) specific functions that are not handled by another memory area. It is defined from byte address range 0xE010 0000 to 0xFFFF FFFF.

The MAX78000 does not implement this memory region.

2.4 AHB Interfaces

This section details memory accessibility on the AHB and the organization of AHB master and slave instances.

2.4.1 Core AHB Interfaces

2.4.1.1 I-Code

This AHB master is used by the Arm core for instruction fetching from memory instances located in code space from byte addresses 0x0000 0000 to 0x1FFF FFFF. This bus master is used to fetch instructions from the internal flash memory. Instructions fetched by this bus master are returned by the instruction cache, which in turn triggers a cache line fill cycle to fetch instructions from the internal flash memory when a cache miss occurs.

2.4.1.2 D-Code

This AHB master is used by the Arm core for data fetches from memory instances located in code space from byte addresses 0x0000 0000 to 0x1FFF FFFF. This bus master has access to the internal flash memory and the information block.

2.4.1.3 System

This AHB master is used by the Arm core for all instruction fetches and data read and write operations involving the SRAM data cache. The APB mapped peripherals (through the AHB-to-APB bridge) and AHB mapped peripheral and memory areas are also accessed using this bus master.

2.4.2 AHB Slaves

2.4.2.1 Standard DMA

The Standard DMA AHB Slave has access to all non-core memory areas accessible by the System bus. The Standard DMA does not have access to the internal Flash memory or Information Blocks.

2.4.2.2 CNN and CNN TX FIFO

The CNN and CNN TX FIFO AHB slaves have access to all non-core memory areas accessible by the System bus. They do not have access to the internal Flash memory or Information Blocks.

2.4.2.3 SPI0

The SPI0 AHB Slave has access to all non-core memory areas accessible by the System bus. SPI0 does not have access to the internal Flash memory or Information Blocks.

2.4.3 AHB Slave Base Address Map

Table 2-3, below, contains the base address for each of the AHB slave peripherals. The base address for a given peripheral is the start of the register map for the peripheral. For a given peripheral, the address for a register within the peripheral is defined as the peripheral's AHB Base Address plus the register's offset.

Table 2-3: AHB Slave Base Address Map

| AHB Slave Register Name | Register Prefix | AHB Base Address | AHB End Address |
|-------------------------|-----------------|------------------|-----------------|
| SPI0 | SPI0_ | 0x400B E000 | 0x400B E3FF |
| CNN TX FIFO | CNN_TX_FIFO_ | 0x400C 0400 | 0x400C 0400 |

2.5 Peripheral Register Map

2.5.1 APB Peripheral Base Address Map

Table 2-4, below, contains the base address for each of the APB mapped peripherals. The base address for a given peripheral is the start of the register map for the peripheral. For a given peripheral, the address for a register within the peripheral is defined as the APB peripheral base address plus the registers offset.

Table 2-4: APB Peripheral Base Address Map

| Peripheral Register Name | Register Prefix | APB Base Address | APB End Address |
|---------------------------------------|-----------------|------------------|-----------------|
| Global Control | GCR_ | 0x4000 0000 | 0x4000 03FF |
| System Interface | SI_ | 0x4000 0400 | 0x4000 07FF |
| Function Control | FCR_ | 0x4000 0800 | 0x4000 0BFF |
| Watchdog Timer 0 | WDT0_ | 0x4000 3000 | 0x4000 33FF |
| Dynamic Voltage Scaling Controller | DVS_ | 0x4000 3C00 | 0x4000 3C3F |
| Single Input Multiple Output | SIMO_ | 0x4000 4400 | 0x4000 47FF |
| Trim System Initialization | TRIMSIR_ | 0x4000 5400 | 0x4000 57FF |
| General Control Function | GCFR_ | 0x4000 5800 | 0x4000 5BFF |
| Real time Clock | RTC_ | 0x4000 6000 | 0x4000 63FF |
| Wakeup Timer | WUT_ | 0x4000 6400 | 0x4000 67FF |
| Power Sequencer | PWRSEQ_ | 0x4000 6800 | 0x4000 6BFF |
| Miscellaneous Control | MCR_ | 0x4000 6C00 | 0x4000 6FFF |
| AES | AES_ | 0x4000 7400 | 0x4000 77FF |
| AES Key | AESK_ | 0x4000 7800 | 0x4000 7BFF |
| GPIO Port 0 | GPIO0_ | 0x4000 8000 | 0x4000 8FFF |
| GPIO Port 1 | GPIO1_ | 0x4000 9000 | 0x4000 9FFF |
| Parallel Camera Interface | PCIF_ | 0x4000 E000 | 0x4000 EFFF |
| CRC | CRC_ | 0x4000 F000 | 0x4000 FFFF |
| Timer 0 | TMR0_ | 0x4001 0000 | 0x4001 0FFF |
| Timer 1 | TMR1_ | 0x4001 1000 | 0x4001 1FFF |
| Timer 2 | TMR2_ | 0x4001 2000 | 0x4001 2FFF |
| Timer 3 | TMR3_ | 0x4001 3000 | 0x4001 3FFF |
| I ² C 0 | I2C0_ | 0x4001 D000 | 0x4001 DFFF |
| I ² C 1 | I2C1_ | 0x4001 E000 | 0x4001 EFFF |
| I ² C 2 | I2C2_ | 0x4001 F000 | 0x4001 FFFF |
| Standard DMA | DMA_ | 0x4002 8000 | 0x4002 8FFF |
| Flash Controller 0 | FLCO_ | 0x4002 9000 | 0x4002 93FF |
| Instruction-Cache Controller 0 (CM4) | ICCO_ | 0x4002 A000 | 0x4002 A7FF |
| Instruction Cache Controller 1 (RV32) | ICC1_ | 0x4002 A800 | 0x4002 AFFF |
| ADC | ADC_ | 0x4003 4000 | 0x4003 4FFF |
| Pulse Train Engine | PT_ | 0x4003 C000 | 0x4003 C09F |

| Peripheral Register Name | Register Prefix | APB Base Address | APB End Address |
|-----------------------------------|-----------------|------------------|-----------------|
| One Wire Master | OWM0_ | 0x4003 D000 | 0x4003 DFFF |
| Semaphore | SEMA_ | 0x4003 E000 | 0x4003 EFFF |
| UART 0 | UART0_ | 0x4004 2000 | 0x4004 2FFF |
| UART 1 | UART1_ | 0x4004 3000 | 0x4004 3FFF |
| UART 2 | UART2_ | 0x4004 4000 | 0x4004 4FFF |
| SPI1 | SPI1_ | 0x4004 6000 | 0x4004 7FFF |
| TRNG | TRNG_ | 0x4004 D000 | 0x4004 DFFF |
| I2S | I2S_ | 0x4006 0000 | 0x4006 0FFF |
| Low Power General Control | LPGCR_ | 0x4008 0000 | 0x4008 03FF |
| GPIO Port 2 | GPIO2_ | 0x4008 0400 | 0x4008 05FF |
| Low Power Watchdog Timer 0 (WDT1) | WDT1_ | 0x4008 0800 | 0x4008 0BFF |
| Low Power Timer 4 | TMR4_ | 0x4008 0C00 | 0x4008 0FFF |
| Low Power Timer 5 | TMR5_ | 0x4008 1000 | 0x4008 13FF |
| Low Power UART 0 (UART3) | UART3_ | 0x4008 1400 | 0x4008 17FF |
| Low Power Comparator | LPCOMP_ | 0x4008 8000 | 0x4008 83FF |
| CNN Global Control | CNN_ | 0x5000 0000 | 0x500F FFFF |
| CNNx16 Quadrant 0 | CNNx16_0_ | 0x5010 0000 | 0x504F FFFF |
| CNNx16 Quadrant 1 | CNNx16_1_ | 0x5050 0000 | 0x508F FFFF |
| CNNx16 Quadrant 2 | CNNx16_2_ | 0x5090 0000 | 0x50CF FFFF |
| CNNx16 Quadrant 3 | CNNx16_3_ | 0x50D0 0000 | 0x510F FFFF |

2.6 Error Correction Coding (ECC) Module

This device features an Error Correction Coding (ECC) module which helps ensure data integrity by detecting and correcting bit corruption of the System RAM0 (*sysram0*) memory array. More specific, this feature is Single Error Correcting, Double Error Detecting (SEC-DED). It corrects any single bit flip, detects 2-bit errors, and features a transparent zero wait state operation for reads.

The ECC works by creating check bits for all data written to *sysram0*. These check bits are then stored along with the data. During a read, both the data and check bits are used to determine if one or more bits have become corrupt. If a single bit has been corrupted this can be corrected. If two bits have been corrupted, it will be detected, but not corrected.

If only one bit is determined to be corrupt, reads will contain the “corrected” value. Reading memory does not correct the errored value stored at the read memory location. It is up to the application firmware to determine the appropriate time and method to write the correct data to memory. It is strongly recommended that the application firmware correct the memory as soon as possible to minimize the chance of a second bit from becoming corrupt, resulting in data loss. Since ECC error checking only occurs during a “read” operation, it is recommended that the application periodically “reads” critical memory so that errors can be identified and corrected.

2.6.1 SRAM

A check bit RAM is used to store *sysram0*'s check bits enabling ECC SEC-DED for *sysram0*. The check bit RAM is not mapped to the user memory space and is not available for application usage.

2.6.2 Limitations

Any read from non-initialized memory can trigger an ECC error since the random check bits will most likely not match the random data bits contained in the memory. Writing *sysram0* to all zeroes prior to enabling ECC functionality can prevent this at the expense of the time required. To zeroize *sysram0*, write *GCR_MEMZ.ram0* to 1.

3. System, Power, Clocks, Reset

There are several clocks used by different peripherals and subsystems. These clocks are highly configurable by firmware, allowing developers to select the combination of application performance and power savings required for the target systems. Support for selectable core operating voltage is provided enabling optimal timing access to the internal memories.

3.1 Oscillator Sources

3.1.1 100MHz Internal Primary Oscillator (IPO)

The MAX78000 includes a 100MHz internal high-speed oscillator, referred to in this document as the Internal Primary Oscillator (IPO). This is the fastest frequency oscillator and draws the most power.

The IPO can optionally be powered down in **LPM** by setting the [*GCR_PM.ipo_pd*](#) field to 1.

The IPO can be selected as the SYS_OSC. To use this oscillator as the SYS_OSC, the following steps must be followed:

1. Enable the IPO by setting [*GCR_CLKCTRL.ipo_en*](#) to 1.
2. Wait until the [*GCR_CLKCTRL.ipo_rdy*](#) field reads 1 indicating the IPO is operating.
3. Set [*GCR_CLKCTRL.sysclk_sel*](#) to 4.
4. Wait until the [*GCR_CLKCTRL.sysclk_rdy*](#) field reads 1. The IPO is now operating as the SYS_OSC.

3.1.2 60MHz Internal Secondary Oscillator (ISO)

This is a low-power internal secondary oscillator that is the Power-On Reset default SYS_OSC. This oscillator is automatically selected as SYS_OSC after a System Reset or POR.

The following steps show how to enable the ISO and select it as the SYS_OSC.

1. Enable the ISO by setting [*GCR_CLKCTRL.iso_en*](#) to 1.
2. Wait until the [*GCR_CLKCTRL.iso_rdy*](#) field reads 1 indicating the ISO is operating.
3. Set [*GCR_CLKCTRL.sysclk_sel*](#) to 0.
4. Wait until the [*GCR_CLKCTRL.sysclk_rdy*](#) field reads 1. The ISO is now operating as the SYS_OSC.

3.1.3 8kHz-30kHz Internal Nano-Ring Oscillator (INRO)

This is an ultra-low power internal oscillator that can be selected as the SYS_OSC. This oscillator is always enabled and cannot be disabled by firmware.

The frequency of this oscillator is configurable to 8kHz, 16kHz or 30kHz. Use the [*TRIMSR_INRO.inro_sel*](#) field to select the desired frequency. On a POR or System Reset, the frequency defaults to 30kHz.

The following steps show how to set the INRO as the SYS_OSC.

1. Verify the [*GCR_CLKCTRL.inro_rdy*](#) field reads 1.
2. Set [*GCR_CLKCTRL.sysclk_sel*](#) to 3.
3. Wait until the [*GCR_CLKCTRL.sysclk_rdy*](#) field reads 1. The INRO is now operating as the SYS_OSC.

3.1.4 7.3728MHz Internal Baud Rate Oscillator (IBRO)

This is a very low power internal oscillator that can be selected as SYS_OSC. This clock can optionally be used as a dedicated baud rate clock for the UARTs. This is useful if the SYS_OSC selected does not generate desired UART baud rates.

The following steps show how to enable the IBRO and select it as the SYS_OSC.

1. Wait until the *GCR_CLKCTRL.ibro_rdy* field reads 1 indicating the IBRO is operating.
2. Set *GCR_CLKCTRL.sysclk_sel* to 5.
3. Wait until the *GCR_CLKCTRL.sysclk_rdy* field reads 1. The IBRO is now operating as the SYS_OSC.

3.1.5 32.768kHz External Real-Time Clock Oscillator (ERTCO)

This is an extremely low power internal oscillator that can be selected as the SYS_OSC. The ERTCO can optionally use a 32.768kHz input clock or an 8kHz independent nano-ring oscillator instead of an external crystal. The internal 32.768kHz clock is available as an output on GPIO P3.1 as Alternate Function 1 (SQWOUT).

This oscillator is the default clock for the Real-Time Clock (RTC). If the RTC is enabled the ERTCO is enabled automatically, independent of the selection of the SYS_OSC. This oscillator is disabled on a Power-On Reset or System Reset.

The following steps show how to enable the ERTCO and select it as the SYS_OSC.

1. Enable the ERTCO by setting *GCR_CLKCTRL.ertco_en* to 1.
2. Wait until the *GCR_CLKCTRL.ertco_rdy* field reads 1 indicating the ERTCO is operating.
3. Set *GCR_CLKCTRL.sysclk_sel* to 6.
4. Wait until the *GCR_CLKCTRL.sysclk_rdy* field reads 1. The ERTCO is now operating as the SYS_OSC.

3.2 System Oscillator (SYS_OSC)

The MAX78000 supports multiple clock sources as the System Oscillator (SYS_OSC). The selected System Oscillator (SYS_OSC) is the clock source for most internal blocks. Each oscillator, description and nominal frequency are shown in *Table 3-1, below*. The ERTCO requires an external crystal for operation. An external clock source, EXT_CLK, is supported on P0.3, Alternate Function 1. Each of the oscillator/clocks are described in detail in section [3.1 Oscillator Sources](#).

Table 3-1: Available System Oscillators

| Oscillator/Clock | Description | Nominal Frequency |
|------------------|--|------------------------------------|
| IPO | <i>Internal Primary Oscillator</i> | 100MHz |
| ISO | <i>Internal Secondary Oscillator</i> | 60MHz |
| INRO | <i>Internal Nano-Ring Oscillator</i> | Configurable 8kHz, 16kHz, or 30kHz |
| IBRO | <i>Internal Baud Rate Oscillator</i> | 7.3728MHz |
| ERTCO | <i>External Real-Time Clock Oscillator</i> | 32.768kHz |
| EXT_CLK | External Clock | Up to 80MHz |

3.2.1.1 System Oscillator Selection

Set the system oscillator using the *GCR_CLKCTRL.sysclk_sel* field. Prior to selecting an oscillator as the system oscillator, the oscillator source must first be enabled and ready. Refer to each individual Oscillator Source's detailed description for the required steps to enable the oscillator and select it as the system oscillator.

When the [GCR_CLKCTRL.sysclk_sel](#) is modified, hardware clears the [GCR_CLKCTRL.sysclk_rdy](#) field and there is a delay until the switchover is complete. When the switchover to the selected SYS_OSC is complete, the [GCR_CLKCTRL.sysclk_rdy](#) field is set to 1 by hardware. Application firmware must verify the switchover is complete before continuing operation.

3.2.1.2 System Clock (SYS_CLK)

The selected SYS_OSC is the input to the system oscillator divider to generate the System Clock (SYS_CLK). The system clock divider divides the selected SYS_OSC by the [GCR_CLKCTRL.sysclk_div](#) field as shown in [Equation 3-1](#).

Equation 3-1: System Clock Scaling

$$\text{SYS_CLK} = \frac{\text{SYS_OSC}}{2^{\text{sysclk_div}}}$$

[GCR_CLKCTRL.sysclk_div](#) is selectable from 0 to 7, resulting in divisors of 1, 2, 4, 8, 16, 32, 64 or 128.

SYS_CLK drives the Arm Cortex-M4 with FPU core, the RV32 core and all Advanced High-Performance Bus (AHB) masters in the system. SYS_CLK generates the following internal clocks as shown below:

- Advanced High-Performance Bus (AHB) Clock
 - ◆ $HCLK = \text{SYS_CLK}$
- Advanced Peripheral Bus (APB) Clock,
 - ◆ $PCLK = \frac{\text{SYS_CLK}}{2}$

The Real-Time Clock (RTC) uses the 32.768kHz external real-time clock oscillator (ERTCO) for its clock source. Optionally, the RTC can run using an internal dedicated 8kHz nano-ring oscillator. Refer to the [Real-Time Clock \(RTC\)](#) chapter for details on using this 8kHz nano-ring oscillator for the RTC.

All oscillators are reset to their POR reset default state during:

- Power-On Reset
- System Reset

Oscillator settings are *not* reset during:

- Soft Reset
- Peripheral Reset

[Table 3-2](#) shows each oscillator's enabled state for each type of reset source in the MAX78000.

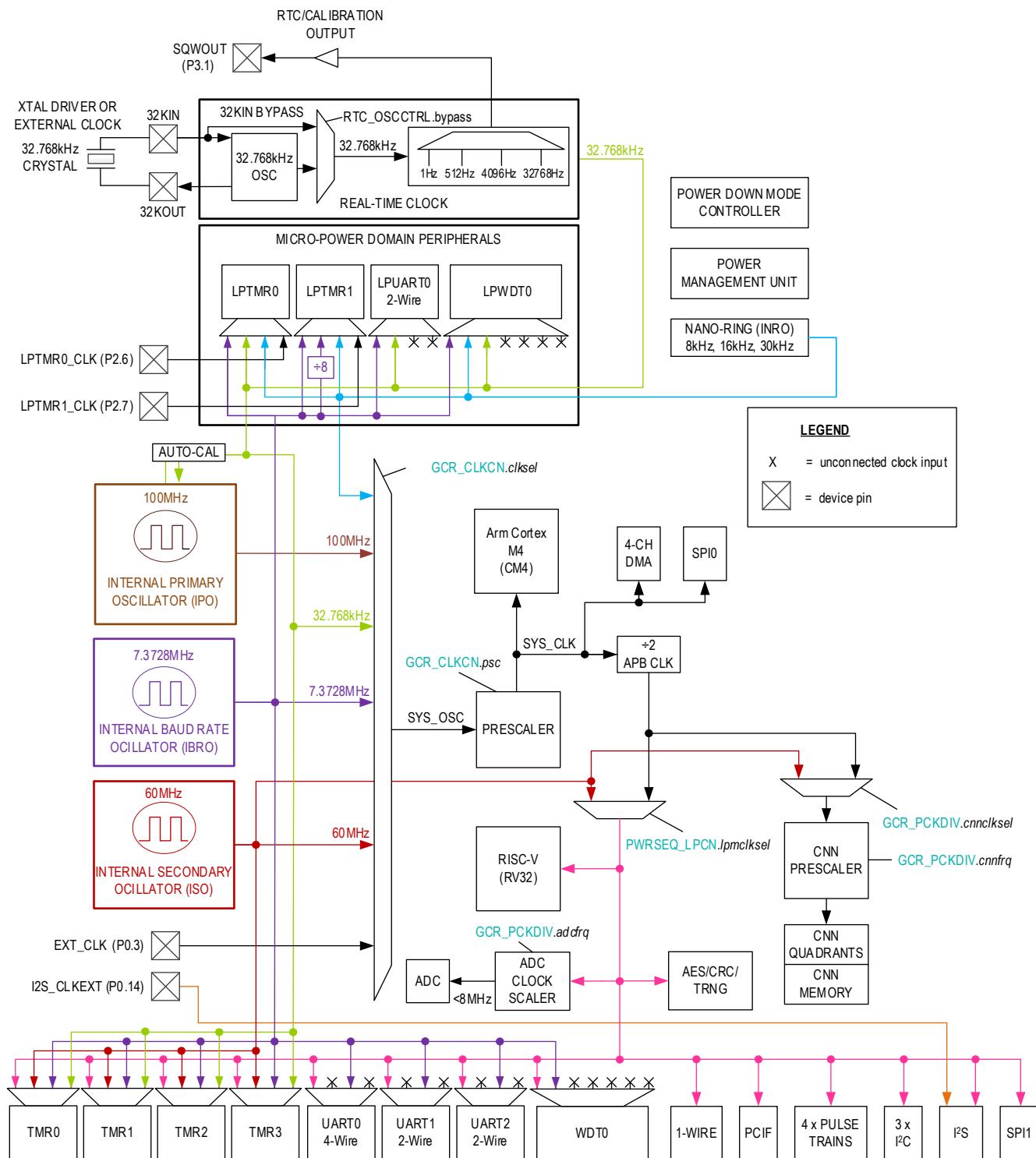
Note: A Watchdog Timer Reset performs a System Reset.

Table 3-2: Reset Sources and Effect on Oscillator and System Clock

| Oscillator | Reset Source | | | |
|-------------------------------|--------------|----------|---------------|---------------|
| | POR | System | Soft | Peripheral |
| IPO | Disabled | Disabled | Retains State | Retains State |
| ISO | Enabled | Enabled | Retains State | Retains State |
| IBRO | Enabled | Enabled | Enabled | Enabled |
| INRO | Enabled | Enabled | Enabled | Enabled |
| ERTCO | Disabled | Disabled | Retains State | Retains State |
| System Clock (SYS_OSC) Source | ISO | ISO | Retains State | Retains State |

[Figure 3-1: MAX78000 Clock Block Diagram](#) shows a high-level diagram of the MAX78000 clock tree.

Figure 3-1: MAX78000 Clock Block Diagram



Preliminary Draft 08/31/2020

3.3 Operating Modes

The MAX78000 includes multiple operating modes and the ability to fine tune power options to optimize performance and power. The system supports the following operating modes:

- **ACTIVE**
- **SLEEP**
- **Low Power Mode (LPM)**
- **Micro Power Mode (UPM)**
- **STANDBY**
- **BACKUP**
- **Power Down Mode (PDM)**

3.3.1 ACTIVE Mode

In this mode, both the CM4 and the RV32 cores can execute application code and all digital and analog peripherals are available on demand. Dynamic clocking disables peripheral not in use, providing the optimal mix of high performance and low power consumption. The CM4 has access to all System RAM by default. The RV32 has access to **sysram2** and **sysram3** and optionally can be configured to have exclusive access to these RAMs. Additionally, **sysram3** can be configured as a unified internal cache controller for the RV32 allowing simultaneous data access and code execution for the CM4 and RV32 from the internal flash memory.

Each of the peripherals can be individually enabled during active mode or powered down. The CNN and each of the four CNNx16_n Processor Arrays and their associated memories can be powered down or set to active mode. See section CNN Configuration for details on enabling the CNNx16_n Processor Arrays and their associated RAM.

3.3.2 SLEEP Mode

This mode consumes less power but wakes faster because the clocks can optionally be enabled.

The device status is as follows:

- The CM4 (CPU0) is sleeping
- The RV32 (CPU1) is sleeping
- The CNN is optionally available for use
 - Each of the four CNNx16_n Quadrants are individually configurable for power down
- Standard DMA is available for use
- All peripherals are on unless explicitly disabled prior to entering **SLEEP**

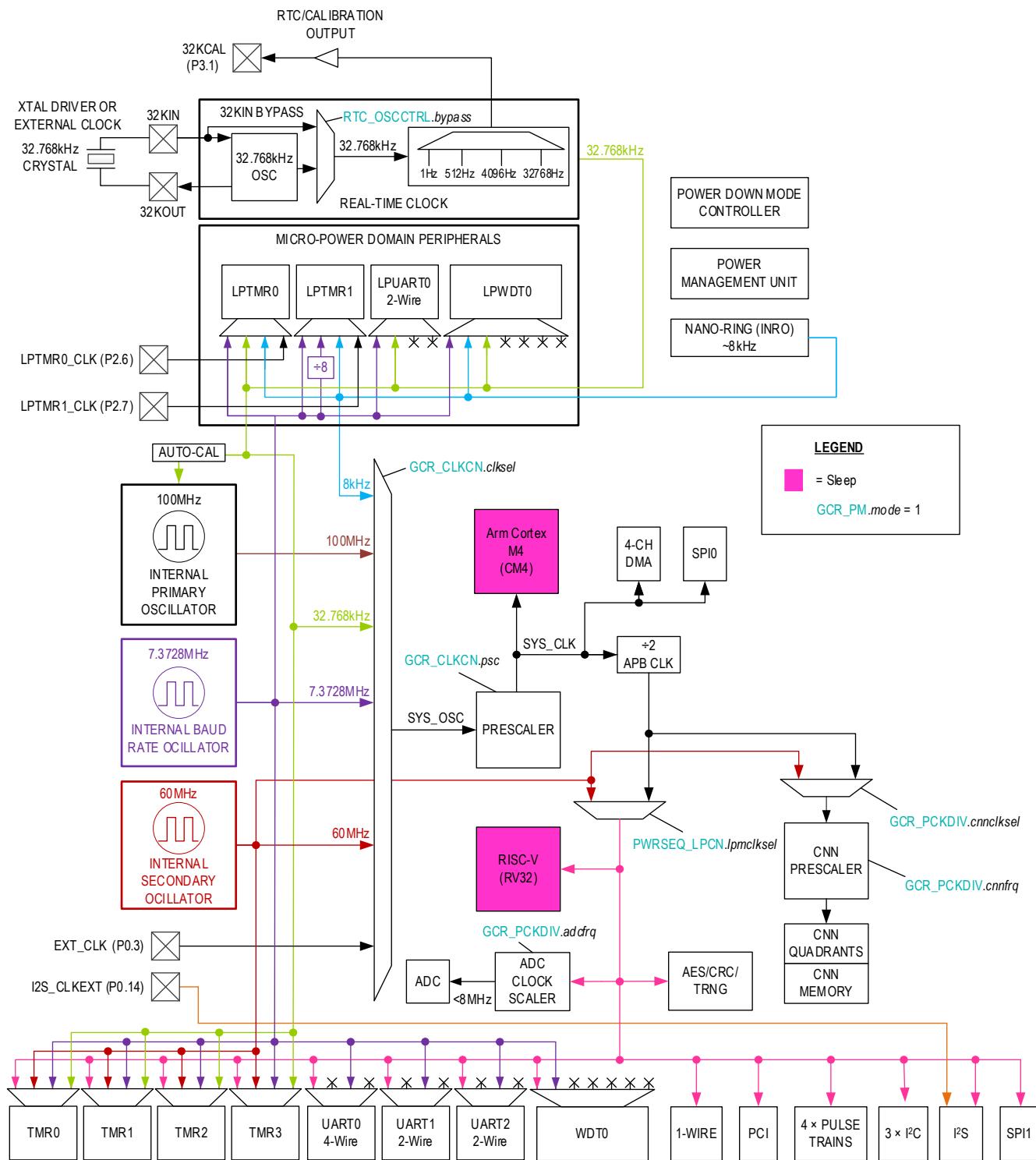
3.3.2.1 Entering SLEEP

Entering **SLEEP** requires both the CM4 and RV32 to cooperate to enter **SLEEP** mode. Synchronization is necessary for deterministic entry into **SLEEP**. Two methods are described below allowing either core to request entry into **SLEEP**. Both methods use the Semaphore peripheral interrupt to communicate between the cores.

If the RV32 is driving entry to **SLEEP**, the RV32 notifies the CM4 of a request to enter **SLEEP** using [Multiprocessor Communications](#). The CM4 receives the notification and then sends a confirmation via the Semaphore peripheral to the RV32. The CM4 should then enter **SLEEP** by setting the SCR.sleepdeep field to 0 and performing a WFI or WFE instruction. The RV32 should then enter **SLEEP** by performing a WFI instruction or by setting [GCR_PM.mode](#) to 1.

Alternatively, the CM4 can initiate the request to enter **SLEEP** by sending the request to the RV32 using [Multiprocessor Communications](#). The RV32 confirms the request via [Multiprocessor Communications](#) and performs a WFI instruction. The CM4 should then enter **SLEEP** by setting SCR.sleepdeep to 0 and performing a WFI or WFE instruction or by setting [GCR_PM.mode](#) to 1.

Figure 3-2: SLEEP Mode Clock Control



Preliminary Draft 08/31/2020

3.3.3 Low Power Mode (LPM)

This mode is suitable for running the RV32 processor to collect and move data from enabled peripherals. The device status is as follows:

- The CM4, *sysram0*, and *sysram1* are in state retention
- The CNN quadrants and memory are active and configurable.
- The RV32 can access the SPI, all UARTS, all Timers, I²C, 1-Wire, Timers, Pulse Train Engine, I²S, CRC, AES, TRNG, Comparators as well as *sysram2* and *sysram3*. *sysram3* can be configured to operate as the RV32 unified instruction cache
- The transition from **LPM** to **ACTIVE** is faster than the transition from **BACKUP** to **ACTIVE** because system initialization is not required
- The DMA is in state retention mode
- *PWRSEQ_GPO* and *PWRSEQ_GP1* registers retain state
- Choose the system PCLK or ISO as the clock source for the RV32 and all Peripherals
 - ◆ *PWRSEQ_LPCN.lpmclksel* defaults to use ISO during LPM, setting this field to 1 uses the PCLK
- The following oscillators are powered down:
 - ◆ IPO
- The following oscillators are enabled:
 - ◆ IBRO
 - ◆ ERTCO
 - ◆ INRO
 - ◆ ISO

3.3.3.1 Entering LPM

Entry into **LPM** should be managed between the two cores using *Multiprocessor Communications* to ensure both cores are in a known state when entering **LPM**.

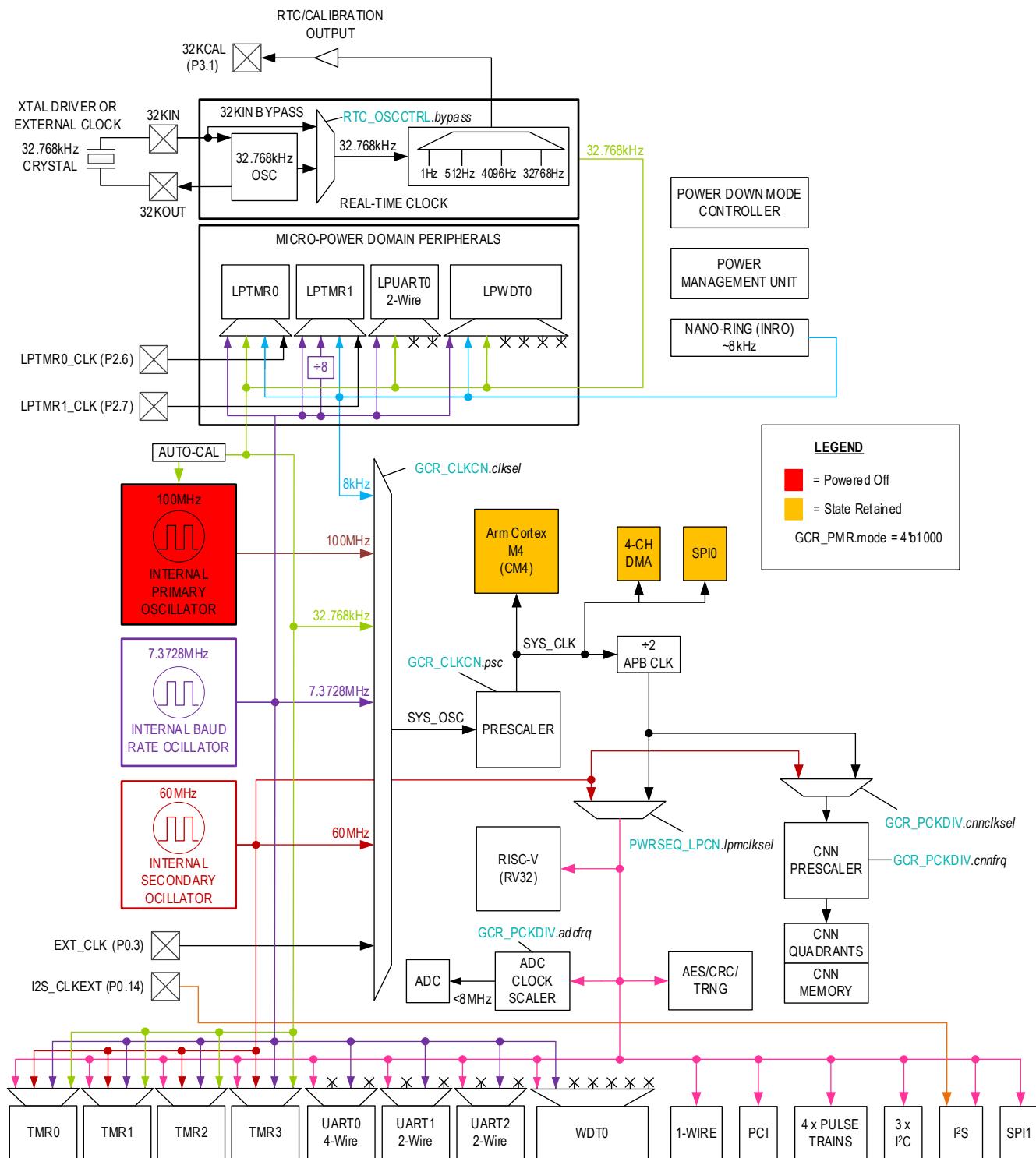
When the CM4 puts itself into *deep sleep*, the device will automatically enter **LPM** and hardware sets the *GCR_PM.mode* to **LPM**. To place the CM4 in **LPM** mode in firmware, perform the following instructions.

```
SCR.sleepdeep = 1; // deep sleep mode enabled  
WFI (or WFE); // Enter deep sleep mode
```

If the RV32 requests the CM4 to enter **LPM** mode via *Multiprocessor Communications* and the CM4 enters **SLEEP** instead, by setting SCR.*sleepdeep* to 0 and performing a WFI or WFE instruction, the RV32 can put the device into **LPM** by directly setting the *GCR_PM.mode* field to **LPM** (8).

Note: The device immediately enters LPM when the GCR_PM.mode field is set to LPM, if the CM4 is not in a known state, issues may occur when exiting LPM.

Figure 3-3: Low Power Mode (LPM) Clock and State Retention Diagram



3.3.4 Micro Power Mode (UPM)

This mode is used for extremely low power consumption while using a minimal set of peripherals to provide wakeup capability. The device status during **UPM** is:

- Both CM4 and RV32 are state retained.
- System state and all System RAM are retained
- CNN quadrants are optionally powered off
- CNN memory provides selectable retention
- The GPIO pins retain their state
- All non-**UPM** peripherals are state retained
- The following oscillators are powered down:
 - ◆ IPO
 - ◆ ISO
- The following oscillators are enabled:
 - ◆ IBRO
 - ◆ ERTCO
 - ◆ INRO
- The following **UPM** peripherals are available for use to wake the device:
 - ◆ LPUART0
 - ◆ LPTMR0
 - ◆ LPTMR1
 - ◆ LPWDT0
 - ◆ LPCOMP0-LPCOMP3
 - ◆ GPIO

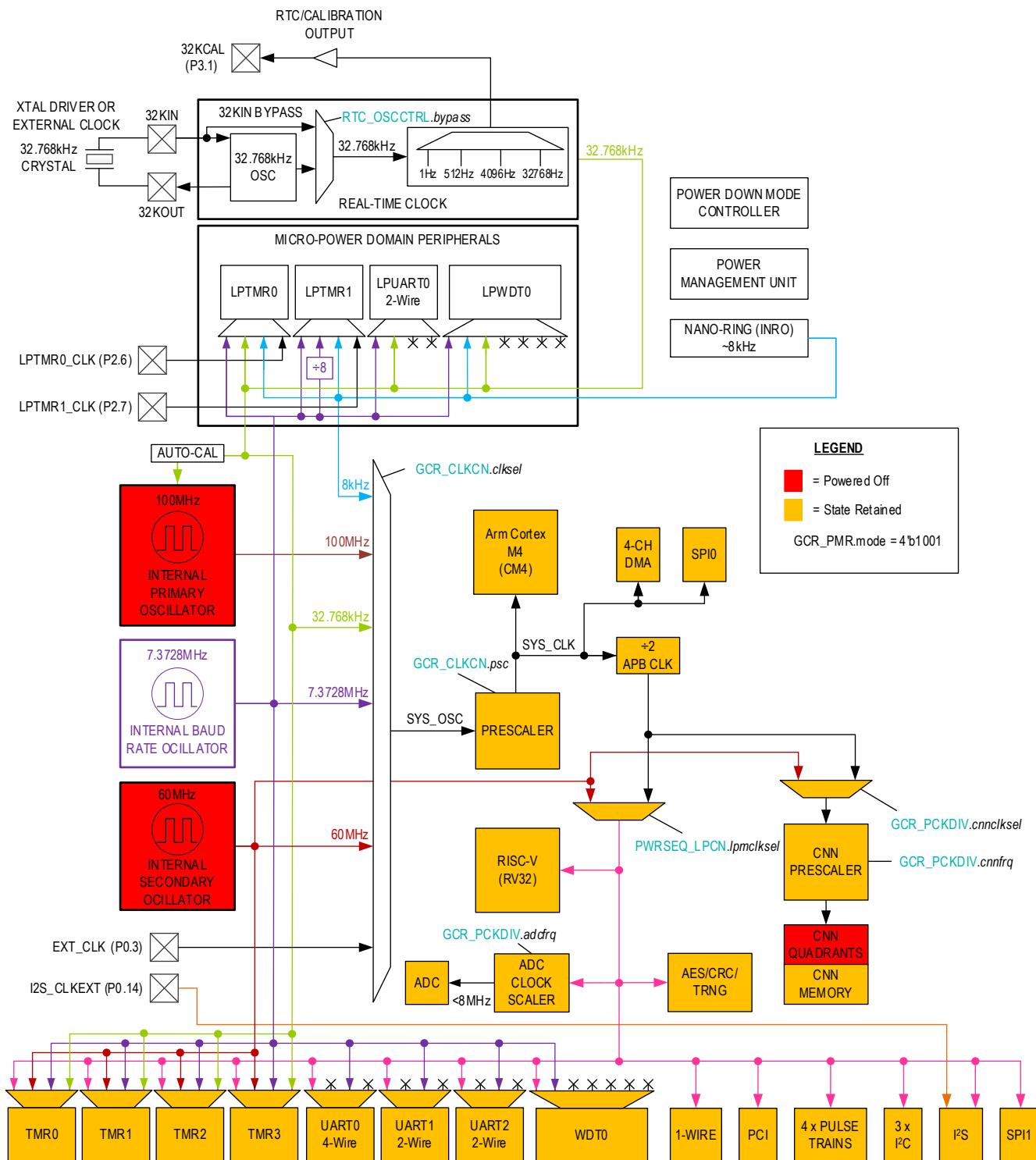
3.3.4.1 Entering UPM

Entering **UPM** mode requires both the CM4 and RV32 to cooperate to enter **UPM** mode. Synchronization is necessary for deterministic entry into **UPM**. Two methods are described below allowing either core to request entry into **UPM** and ensuring deterministic entry. Both methods use the Semaphore peripheral interrupt to communicate between the cores.

If the RV32 is driving entry to **UPM**, the RV32 notifies the CM4 of a request to enter **UPM** using [Multiprocessor Communications](#). The CM4 receives the notification and then sends a confirmation via the Semaphore peripheral to the RV32. The CM4 should then enter **SLEEP** by setting SCR.sleepdeep to 0 and performing a WFI or WFE instruction. The RV32 sets the [*GCR_PM.mode*](#) to **UPM** and the device immediately enters into **UPM**.

Alternatively, the CM4 can initiate the request to enter **UPM** by sending the request to the RV32 using [Multiprocessor Communications](#). The RV32 confirms the request via [Multiprocessor Communications](#) and performs a WFI instruction. The CM4 the sets the [*GCR_PM.mode*](#) to **UPM** and the device immediately enters **UPM**.

Figure 3-4: Micro Power Mode (UPM) Clock and State Retention Block Diagram



Preliminary Draft 08/31/2020

3.3.5 STANDBY Mode

This mode is used to maintain system operation while keeping time with the RTC. The device status is as follows:

- Both CM4 and RV32 are state retained.
- System state and all System RAM is retained
- CNN quadrants are powered off
- CNN memory provides selectable retention (Optional state retention)
- GPIO pins retain their state
- All peripherals retain state
- The following oscillators are powered down:
 - ◆ IPO
 - ◆ ISO
- The following oscillators are enabled:
 - ◆ ERTCO
 - ◆ INRO
 - ◆ IBRO

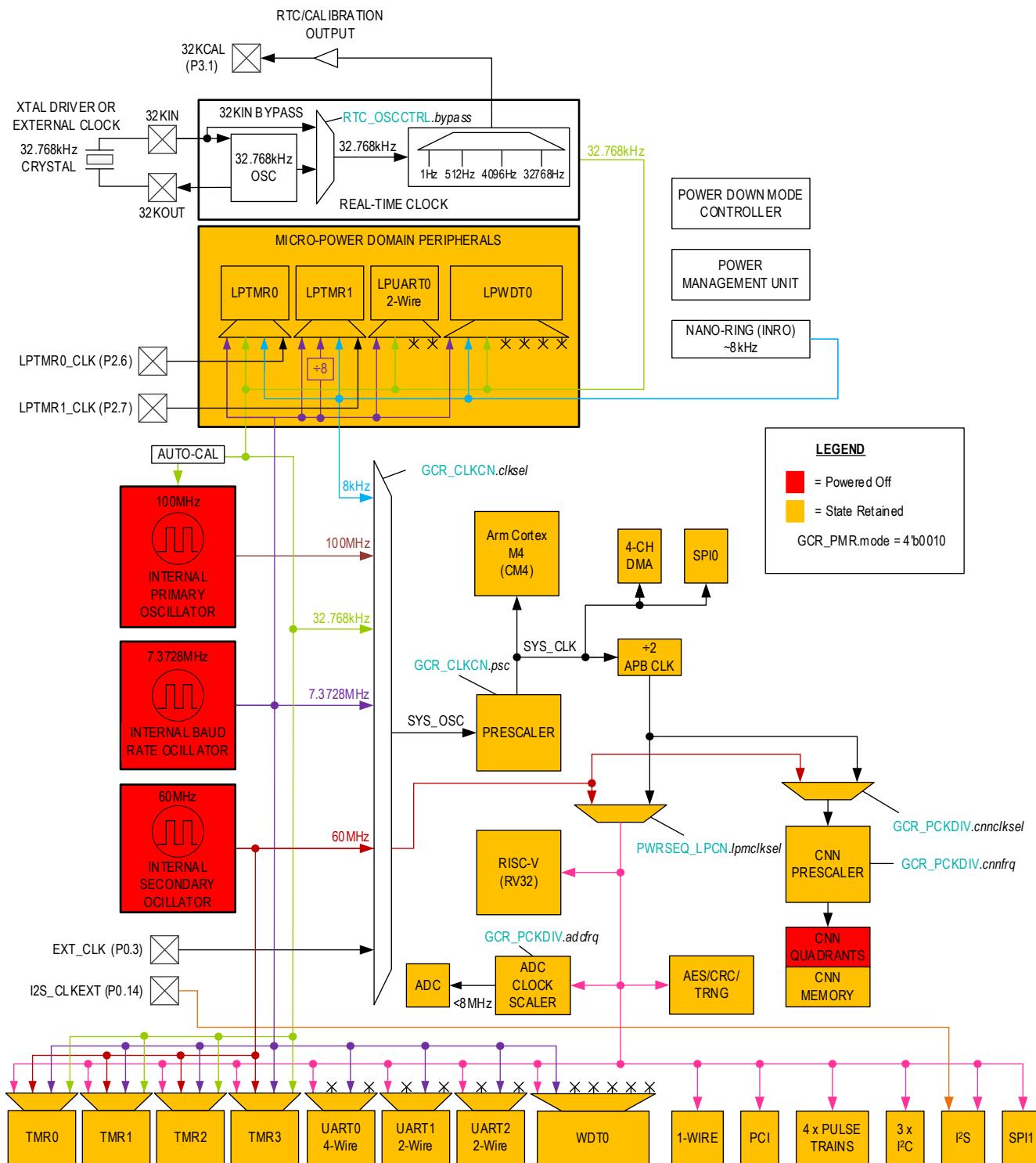
3.3.5.1 Entering STANDBY

Entering **STANDBY** requires both the CM4 and RV32 to enter **STANDBY** mode. Synchronization is necessary for deterministic entry into **STANDBY**. Two methods are described below allowing either core to request entry into **STANDBY** and ensuring deterministic entry. Both methods use the Semaphore peripheral interrupt to communicate between the cores.

If the RV32 is driving entry to **STANDBY**, the RV32 notifies the CM4 of a request to enter **STANDBY** using [Multiprocessor Communications](#). The CM4 receives the notification and then sends a confirmation via the Semaphore peripheral to the RV32. The CM4 should then enter **SLEEP** by setting SCR.sleepdeep to 0 and performing a WFI or WFE instruction. The RV32 sets the [*GCR_PM.mode*](#) to **STANDBY** and the device immediately enters into **STANDBY**.

Alternatively, the CM4 can initiate the request to enter **STANDBY** by sending the request to the RV32 using [Multiprocessor Communications](#). The RV32 confirms the request via [Multiprocessor Communications](#) and performs a WFI instruction. The CM4 then sets the [*GCR_PM.mode*](#) to **STANDBY** and the device immediately enters **STANDBY**.

Figure 3-5: STANDBY Mode Clock and State Retention Block Diagram



Preliminary Draft 08/31/2020

3.3.6 BACKUP Mode

This mode is used to maintain the System RAM. The device status is as follows:

- CM4 and RV32 are powered off.
- sysram0**, **sysram1**, **sysram2** and **sysram3** can be independently configured to be state retained as shown in [Table 3-3](#).
- User configurable CNN memory retention
- All peripherals are powered off
- The following oscillators are powered down:
 - IPO
 - ISO
 - IBRO
 - INRO
- The following oscillators are enabled:
 - ERTCO (The RTC peripheral can be turned off, but not the oscillator)

Table 3-3 System RAM Retention in BACKUP Mode

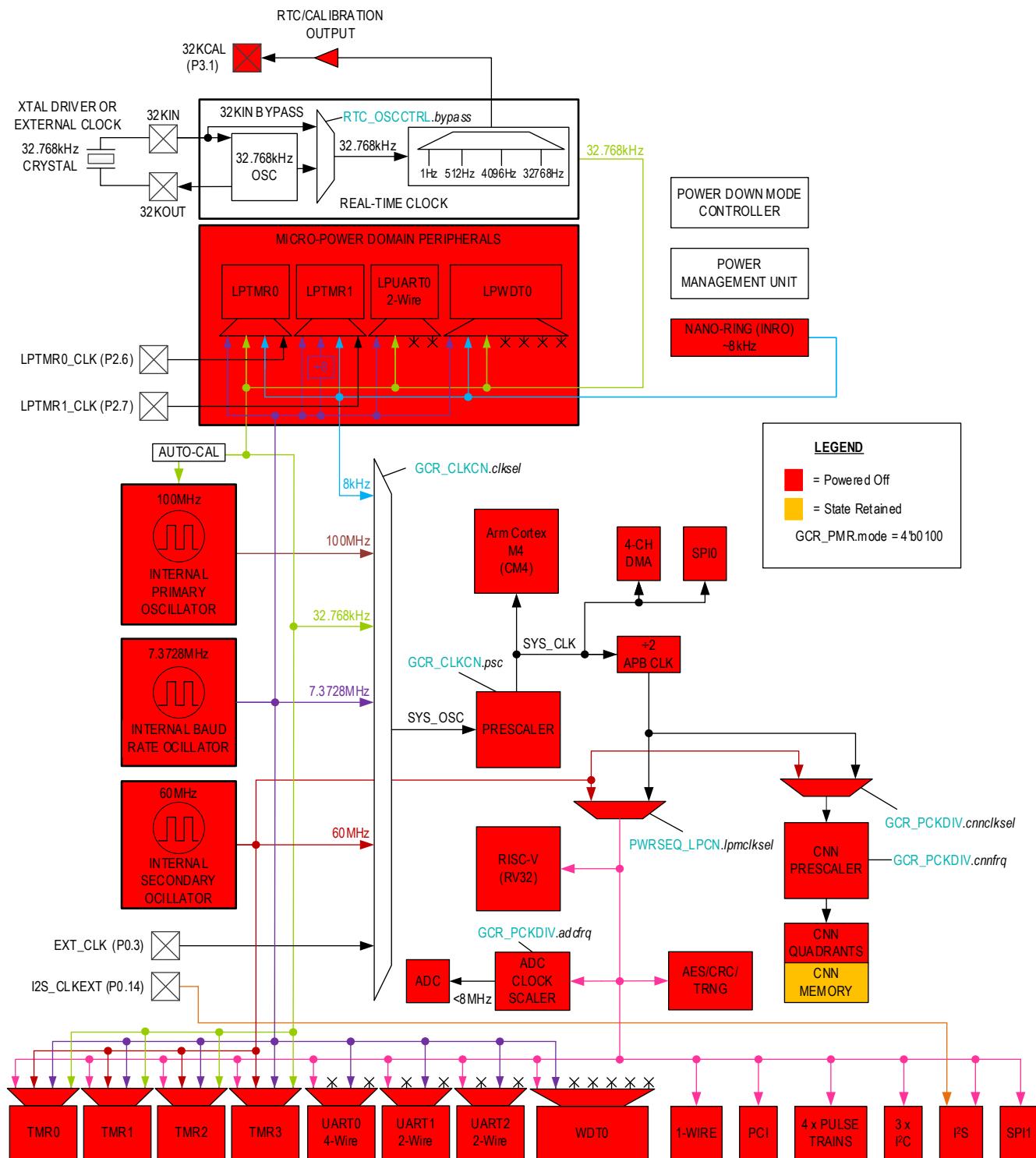
| RAM Block # | Size | State Retention Control |
|----------------|-----------------------|-------------------------------------|
| sysram0 | 32KB + ECC if enabled | PWRSEQ_LPCN.ramret0 |
| sysram1 | 32KB | PWRSEQ_LPCN.ramret1 |
| sysram2 | 48KB | PWRSEQ_LPCN.ramret2 |
| sysram3 | 16KB | PWRSEQ_LPCN.ramret3 |

3.3.6.1 Entering BACKUP

Entering **BACKUP** mode does not require synchronization between the RV32 and CM4 cores. However, it is recommended that [Multiprocessor Communications](#) is used to ensure both cores are aware of entry into BACKUP mode and complete any memory transactions prior to entry.

Either core can set [GCR_PM.mode](#) to **BACKUP** and the device immediately enters **BACKUP**.

Figure 3-6: BACKUP Mode Clock and State Retention Block Diagram



3.3.7 POWER DOWN Mode (PDM)

This mode is used during product level distribution and storage. The device status is as follows:

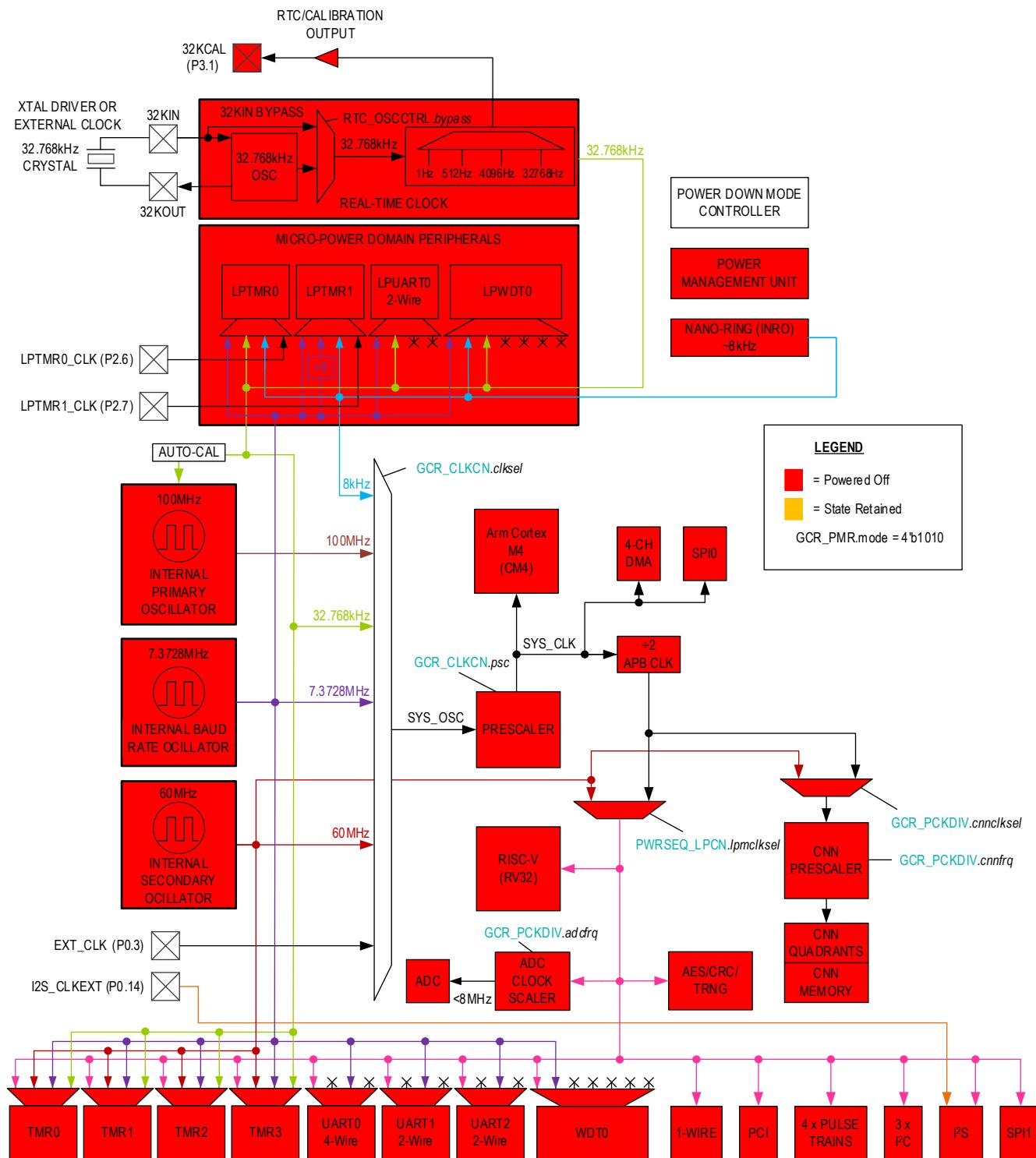
- The CM4 and RV32 are powered off
- All peripherals and all RAMs are powered down
- All oscillators are powered down
- There is no data retention in this mode, but values in the flash are preserved
- V_{REGI} POR voltage monitor is operational.
- Exit from PDM is possible via an external reset (RSTN) or a wake up event on using either P3.0 or P3.1 if configured.

3.3.7.1 Entering PDM

Entering **PDM** does not require synchronization between the RV32 and CM4 cores. However, it is recommended that [Multiprocessor Communications](#) is used to ensure both cores are aware of entry into **PDM** and complete any flash memory transactions.

Either core can set [*GCR_PM.mode*](#) to **PDM** and the device immediately enters **PDM**.

Figure 3-7: Power Down Mode (PDM) Clock and State Retention Block Diagram



3.4 Operating Modes Wakeup Sources

In all operating modes other than **ACTIVE**, wakeup sources are required to re-enter **ACTIVE** operation. The Table below shows available wakeup sources for each operating mode of the MAX78000.

Note: Each wakeup source must be enabled individually except for External Reset, which is hardware controlled.

Table 3-4: Wakeup Sources for Each Operating Mode in the MAX78000

| Operating Mode | Any Peripheral Interrupts | External Reset | RV32 | CNN | CNN FIFO | SPI1 | SPI0 | I2S | I2C2 | I2C1 | I2C0 | LPUART0 (UART3) | UART2 | UART1 | UART0 | LPTMR1 (TMR5) | LPTMR0 (TMR4) | TMR3 | TMR2 | TMR1 | TMR0 | LPWDT0 (WDT1) | WDT0 | LPCMOP3 | LPCMOP2 | LPCMOP1 | COMP0 | RTC | WUT | GPIO3 | GPIO2 | GPIO1 | GPIO0 | |
|----------------|---------------------------|----------------|------|-----|----------|------|------|-----|------|------|------|-----------------|-------|-------|-------|---------------|---------------|------|------|------|------|---------------|------|---------|---------|---------|-------|-----|-----|-------|-------|-------|-------|--|
| SLEEP | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| LPM | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| UPM | | ✓ | | | | | | | | | | ✓ | | | ✓ | ✓ | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| STANDBY | | ✓ | | | | | | | | | | | | | | | | | | | | | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| BACKUP | | ✓ | | | | | | | | | | | | | | | | | | | | | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| PDM | | ✓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

3.5 Device Resets

Four device resets are available:

- Peripheral Reset
- Soft Reset
- System Reset
- Power-On Reset

On completion of any of the four reset cycles, all peripherals are reset. On completion of any reset cycle HCLK and PCLK are operational, the CPU core receives clocks and power, and the device is in **ACTIVE** mode. Program execution begins at the reset vector address.

Contents of the Always-On Domain (AoD) are reset only on power-cycling V_{COREA}, V_{COREB}, V_{DDA}, V_{DDIOH} or V_{REGI}.

Each of the on-chip peripherals can also be reset to their POR default state using the two reset registers [GCR_RST0](#) and [GCR_RST1](#).

Table 3-5: Resets and Low Power Mode Effects shows the effects of each reset type and each of the operating modes.

Table 3-5: Operating Mode and Clock Status

| Clock | Operating Mode | | | | | |
|-------|----------------|-----|-----|---------|--------|-----|
| | SLEEP | LPM | UPM | STANDBY | BACKUP | PDM |
| IPO | CFG | Off | Off | Off | Off | Off |
| ISO | | | | | | Off |
| INRO | | | | | | Off |
| IBRO | | | | | | Off |
| ERTCO | | | | | On | Off |

3.5.1 Peripheral Reset

This resets all peripherals. The CPU retains its state. The GPIO, Watchdog Timers, AoD, RAM retention, and General Control Registers (GCR), including the clock configuration, are unaffected.

To start a Peripheral Reset, set *GCR_RST0.periph* = 1. The reset will be completed immediately upon setting *GCR_RST0* = 1.

3.5.2 Soft Reset

This is the same as a Peripheral Reset except that it also resets the GPIO to its Power-On Reset state.

To start a Soft Reset, set *GCR_RST0.soft* = 1. The reset will be completed immediately upon setting *GCR_RST0.soft* = 1.

3.5.3 System Reset

This is the same as Soft Reset except it also resets all GCR, resetting the clocks to their default state. The CPU state is reset as well as the watchdog timers. The AoD and RAM are unaffected.

A watchdog timer reset event initiates a System Reset. To start a System Reset from firmware, set *GCR_RST0.sys* = 1.

3.5.4 Power-On Reset

A POR resets everything in the device to its default state. A POR results from V_{COREA} , V_{COREB} , V_{DDA} or V_{REGI} falling below their reset voltage level. Refer to the *MAX78000 Datasheet* for details of the reset voltage levels.

3.6 Unified Internal Cache Controllers

The MAX78000 includes two unified internal cache controllers. ICC0 is the cache controller used for the CM4. ICC1, if enabled as a cache controller, is dedicated to the RV32 core. ICC1 uses *sysram3* as the cache memory. If ICC1 is enabled, *sysram3* is not accessible as SRAM (address range 0x2001 C000 to 0x2001 FFFF).

Both caches, ICC0 and ICC1, include a line buffer, tag RAM and a 16KB 2-way set associative RAM when enabled.

3.6.1 Enabling the Internal Cache Controllers

Enabling ICC1 for use as the cache controller for the RV32 requires using *sysram3* as the cache memory.

Note: The contents of sysram3 are lost when ICC1 is enabled and sysram3 is not accessible for data reads or writes as part of the memory map.

Note: Prior to enabling ICC1 as a cache controller, sysram3 should be zeroized.

Perform the following steps to enable ICCn:

1. Set the *ICCn_CTRL.en* to 0, ensuring the cache is invalidated when enabled.
2. Set *ICCn_CTRL.en* to 1.
3. Read *ICCn_CTRL.rdy* until it returns 1.
4. Zeroize the ICC instance by setting *GCR_MEMZ.icc0* or *GCR_MEMZ.icc1* to 1.

3.6.2 Disabling the ICC

Disable an ICC instance by setting *ICCn_CTRL.en* to 0.

To use *sysram3* as data RAM, first disable the ICC1 instance as described above. When ICC1 is disabled, *sysram3* is accessible as data RAM by both the CM4 and RV32 controllers unless *sysram3* is configured for exclusive access by the RV32 core only.

3.6.3 Invalidating the ICC Cache and Tag RAM

The System Configuration Register (*GCR_SYSCTRL*) includes a field for flushing the ICC0 cache. Setting *GCR_SYSCTRL.icc0_flush* to 1 flushes the ICC0 16KB cache and the tag RAM.

Invalidate the contents of a specific ICC instance by setting the *ICCn_INVALIDATE* register to 1. Once invalidated, the system flushes the cache. Read the *ICCn_CTRL.rdy* field until it returns 1 to determine when the flush is completed.

3.6.4 Flushing the ICC

Flush ICC0 using the System Configuration Register (*GCR_SYSCTRL*). Set *GCR_SYSCTRL.icc0_flush* to 1 to immediately flush the contents of the 16KB cache and tag RAM.

Flush ICC1 using the RV32 Control Register (*FCR_URVCTRL*). Set *FCR_URVCTRL.icc1_flush* to 1 to immediately flush the contents of the 16KB cache and tag RAM.

3.6.5 Internal Cache Control Registers (ICC)

See *Table 2-4: APB Peripheral Base Address Map* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in *Table 3-6: Instruction Cache Controller Register Summary*. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register *PERIPHERALn_CTRL* resolves to *PERIPHERAL0_CTRL* and *PERIPHERAL1_CTRL* for instances 0 and 1, respectively.

See *Table 2-1: Field Access Definitions* for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 3-6: Instruction Cache Controller Register Summary

| Offset | Register | Name |
|----------|------------------------|---|
| [0x0000] | <i>ICCn_INFO</i> | <i>Cache ID Register</i> |
| [0x0004] | <i>ICCn_SZ</i> | <i>Cache Memory Size Register</i> |
| [0x0100] | <i>ICCn_CTRL</i> | <i>Instruction Cache Control Register</i> |
| [0x0700] | <i>ICCn_INVALIDATE</i> | <i>Instruction Cache Controller Invalidate Register</i> |

3.6.6 ICC0 Register Details

Table 3-7: ICC0 Cache Information Register

| ICCO Cache Information | | | <i>ICCn_INFO</i> | [0x0000] |
|------------------------|---------|--------|------------------|--|
| Bits | Field | Access | Reset | Description |
| 31:16 | - | RO | 0 | Reserved Do not modify |
| 15:10 | id | R | - | Cache ID Returns the ID for this cache instance. |
| 9:6 | partnum | R | - | Cache Part Number Returns the part number indicator for this cache instance. |

| ICCO Cache Information | | | | ICCn_INFO | [0x0000] |
|------------------------|--------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 5:0 | relnum | R | - | Cache Release Number Returns the release number for this cache instance. | |

Table 3-8: ICCO Memory Size Register

| ICCO Memory Size | | | | ICCn_SZ | [0x0004] |
|------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | mem | R | - | Addressable Memory Size Indicates the size of addressable memory by this cache controller instance in 128KB units. | |
| 15:0 | cch | R | - | Cache Size Returns the size of the cache RAM memory in 1KB units. 16: 16KB Cache RAM | |

Table 3-9: ICCO Cache Control Register

| ICCO Cache Control | | | | ICCn_CTRL | [0x0100] |
|--------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | - | Reserved Do not modify | |
| 16 | rdy | R | - | Ready This field is cleared by hardware anytime the cache as a whole is invalidated (including a POR). Hardware automatically sets this field to 1 when the invalidate operation is complete and the cache is ready. 0: Cache invalidation in process. 1: Cache is ready. <i>Note: While this field reads 0, the cache is bypassed and reads come directly from the line fill buffer.</i> | |
| 15:1 | - | R/W | - | Reserved Do not modify | |
| 0 | en | R/W | 0 | Cache Enable Set this field to 1 to enable the cache. Setting this field to 0 automatically invalidates the cache contents and reads are handled by the line fill buffer. 0: Disable 1: Enable | |

Table 3-10: ICCO Invalidate Register

| ICCO Invalidate | | | | ICCn_INVALIDATE | [0x0700] |
|-----------------|---------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | invalid | W | - | Invalidate Writing any value to this register invalidates the cache. | |

3.7 RAM Memory Management

This device has many features for managing the on-chip RAM. The on-chip RAM includes data RAM, unified cache controller (ICC0) for CPU0 and unified cache controller (ICC1) for CPU1, CNN RAM, and peripheral FIFOs.

3.7.1 On-Chip Cache Management

The MAX78000 includes two unified internal cache controllers for code and data fetches from the flash memory. The caches can be enabled, disabled, zeroized and flushed. Refer to section [Unified Internal Cache Controller](#) for details.

3.7.2 RAM Zeroization

The GCR Memory Zeroize Register, [*GCR_MEMZ*](#), allows clearing memory for firmware or security reasons. Zeroization writes all zeros to the specified memory.

The following SRAM memories can be zeroized:

- Each of the System RAMs can be individually zeroized by setting the respective [*GCR_MEMZ*](#) bit.
 - ◆ [*GCR_MEMZ.ram0*](#)
 - ◆ [*GCR_MEMZ.ram0ecc*](#)
 - ◆ [*GCR_MEMZ.ram1*](#)
 - ◆ [*GCR_MEMZ.ram2*](#)
 - ◆ [*GCR_MEMZ.ram3*](#)
- ICC0 16KB Cache
 - ◆ [*GCR_MEMZ.icc0*](#)
- ICC1 16KB Cache, if enabled
 - ◆ [*GCR_MEMZ.icc1*](#)
- Each of the CNNx16n Processor Arrays support zeroizing the TRAM, MRAM, Bias RAM and SRAM
 - ◆ [*CNNx16_n_TEST.tramz*](#) set to 1 to zero, read [*CNNx16_n_TEST.tallzdone*](#) until 1 for completion
 - ◆ [*CNNx16_n_TEST.mramz*](#) set to 1 to zero, read [*CNNx16_n_TEST.mallzdone*](#) until 1 for completion
 - ◆ [*CNNx16_n_TEST.bramz*](#) set to 1 to zero, read [*CNNx16_n_TEST.ballzdone*](#) until 1 for completion
 - ◆ [*CNNx16_n_TEST.sramz*](#) set to 1 to zero, read [*CNNx16_n_TEST.sallzdone*](#) until 1 for completion

3.8 Miscellaneous Control Registers (MCR)

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-11: Miscellaneous Control Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 3-11: Miscellaneous Control Register Summary

| Offset | Register Name | Access | Description |
|----------|---------------------------------------|--------|--|
| [0x0000] | <i>MCR_ECCEN</i> | R/W | <i>Error Correction Coding Enable Register</i> |
| [0x0004] | <i>MCR_IPO_MTRIM</i> | R/W | <i>IPO Manual Trim Register</i> |
| [0x0008] | <i>MCR_OUTEN</i> | R/W | <i>Miscellaneous Output Enable Register</i> |
| [0x000C] | <i>MCR_CMPO_CTRL</i> | R/W | <i>Comparator Control Register</i> |
| [0x0010] | <i>MCR_CTRL</i> | R/W | <i>Miscellaneous Control Register</i> |
| [0x0020] | <i>MCR_GPIO3_CTRL</i> | R/W | <i>GPIO3 Pin Control Register</i> |

3.8.1 Miscellaneous Control Register Details

Table 3-12: Error Correction Coding Enable Register

| Error Correction Coding Enable | | | MCR_ECCEN | | [0x0000] |
|--------------------------------|------|--------|-----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:1 | - | RO | 0 | Reserved Do not modify | |
| 0 | ram0 | R/W | 0 | System RAM 0 ECC Enable Set this field to 1 to enable ECC for <i>sysram0</i> . 0: Disabled 1: Enabled | |

Table 3-13: IPO Manual Register

| IPO Manual Trim | | | MCR_IPO_MTRIM | | [0x0004] |
|-----------------|------------|--------|---------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:9 | - | RO | 0 | Reserved | |
| 8 | trim_range | R/W | 0 | Trim Range Select If this bit is set to 1, the value loaded into <i>MCR_IPO_MTRIM.mtrim</i> must be greater than the trim setting in <i>TRIMSIR_IPOLO.ipo_limitlo</i> . If this bit is set to 0, the value loaded into <i>MCR_IPO_MTRIM.mtrim</i> must be less than the trim setting in <i>TRIMSIR_CTRL.ipo_limithi</i> . 0: <i>MCR_IPO_MTRIM.mtrim</i> < <i>TRIMSIR_IPOLO.ipo_limitlo</i> 1: <i>MCR_IPO_MTRIM.mtrim</i> > <i>TRIMSIR_CTRL.ipo_limithi</i> | |
| 7:0 | mtrim | R/W | 0x04 | Manual Trim Value Set this value to the desired manual trim based on the value set in <i>MCR_IPO_MTRIM.trim_range</i> . If <i>MCR_IPO_MTRIM.trim_range</i> is 0, the value in this field must be less than the value in <i>TRIMSIR_IPOLO.ipo_limitlo</i> . If <i>MCR_IPO_MTRIM.trim_range</i> is 1, the value in this field must be greater than the value in <i>TRIMSIR_CTRL.ipo_limithi</i> . | |

Table 3-14: Output Enable Register

| Output Enable | | | MCR_OUTEN | | [0x0008] |
|---------------|--------------|--------|-----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:2 | - | RO | 0 | Reserved Do not modify | |
| 1 | pdown_out_en | R/W | 0 | Power Down Output Enable on P3.0 Set this field to 1 to enable the power down output, P3.0 AF1 (PDOWN). PDOWNZ is active in Backup and Standby Mode. 0: PDOWN output not enabled on P3.0 1: PDOWN output is enabled on P3.0 | |
| 0 | sqwout_en | R/W | 0 | Square Wave Output Enable on P3.1 (SQWOUT) Set this field to 1 to enable the square wave output on P3.1 AF1 (SQWOUT). 0: Square wave output not enabled on P3.1 1: Square wave output enabled on P3.1 | |

Table 3-15: Comparator 0 Control Register

| Comparator 0 Control | | | MCR_CMPO_CTRL | | [0x000C] |
|----------------------|--------|--------|---------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved Do not modify | |
| 15 | if | R/W1C | 0 | Comparator 0 Interrupt Flag This field is set to 1 by hardware when the comparator output changes to the active state as set using the <i>MCR_CMPO_CTRL.pol</i> field. Write 1 to clear this flag. 0: No interrupt 1: Interrupt occurred | |
| 14 | out | RO | * | Comparator 0 Output This field is the comparator output state. 0: Output low 1: Output high | |
| 13:7 | - | RO | 0 | Reserved Do not modify | |
| 6 | int_en | R/W | 0 | Comparator 0 Interrupt Enable Set this field to 1 to enable the IRQ for comparator 0. 0: Interrupt disabled 1: Interrupt enabled | |
| 5 | pol | R/W | 0 | Comparator 0 Interrupt Polarity Select Set this field to select the polarity of the output change that generates a comparator 3 interrupt. 0: Interrupt occurs from a transition from low to high 1: Interrupt occurs from a transition from high to low | |
| 4:1 | - | RO | 0 | Reserved Do not modify | |
| 0 | en | R/W | 0 | Comparator 0 Enable Set this field to 1 to enable the comparator 0: Comparator disabled 1: Comparator enable | |

Table 3-16: Miscellaneous Control Register

| Miscellaneous Control | | | MCR_CTRL | | [0x0010] |
|-----------------------|-----------|--------|----------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:10 | - | RO | 0 | Reserved Do not modify | |
| 9 | simo_rstd | R/W | 0 | SIMO System Reset Disable If this field is set, the SIMO is only reset by a Power-On Reset. When this bit is set, the VSET* stay's unchanged when exiting all low power modes. 0: The SIMO is reset by all system resets. 1: The SIMO is only reset by a Power-On Reset. | |

| Miscellaneous Control | | | MCR_CTRL | | [0x0010] |
|-----------------------|----------------|--------|----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 8 | simo_clkscl_en | R/W | 0 | SIMO Clock Scaling Enable Set this field to 1 to enable dynamic clock scaling to the SIMO based on load current. When enabled, the SIMO clock slows down in low power modes, reducing current consumption. 0: SIMO clock scaling disabled 1: SIMO clock scaling enabled | |
| 7:4 | - | DNM | 0x01 | Reserved Do not modify | |
| 3 | ertco_en | R/W | 0 | ERTCO Enable Set this field to 1 to enable the ERTCO. 0: ERTCO disabled 1: ERTCO enabled | |
| 2 | inro_en | R/W | 0 | INRO Enable Set this field to 1 to enable the INRO. 0: INRO disabled 1: INRO enabled | |
| 1:0 | - | RO | 0 | Reserved Do not modify | |

Table 3-17: GPIO3 Pin Control Register

| GPIO3 Pin Control | | | MCR_GPIO3_CTRL | | [0x0020] |
|-------------------|--------|--------|-----------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved Do not modify | |
| 7 | p31_in | RO | See Description | GPIO3 Pin 1 Input Status Read this field to determine the input status of P3.1. 0: Input Low 1: Input High | |
| 6 | p31_pe | R/W | 0 | GPIO3 Pin 1 Pull-up Enable Set this bit to 1 to enable the pull-up resistor for P3.1 0: Pull-up Disabled 1: Pull-up Enabled | |
| 5 | p31_oe | R/W | 0 | GPIO3 Pin 1 Output Enable Set this bit to 1 to enable P3.1 for output mode. 0: Input mode 1: Output mode enabled. | |
| 4 | p31_do | R/W | 0 | GPIO3 Pin 1 Data Output If p31_oe is set to 1, this field is used to control the output state of P3.1. 0: Output low if p31_oe is 1 1: Output high if p31_oe is 1. | |

| GPIO3 Pin Control | | | MCR_GPIO3_CTRL | | [0x0020] |
|-------------------|--------|--------|-----------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 3 | p30_in | RO | See Description | GPIO3 Pin 0 Input Status Read this field to determine the input status of P3.0. 0: Input Low 1: Input High | |
| 2 | p30_pe | R/W | 0 | GPIO3 Pin 0 Pull-up Enable Set this bit to 1 to enable the pull-up resistor for P3.0 0: Pull-up Disabled 1: Pull-up Enabled | |
| 1 | p30_oe | R/W | 0 | GPIO3 Pin 0 Output Enable Set this bit to 1 to enable P3.0 for output mode. 0: Input mode 1: Output mode enabled. | |
| 0 | p30_do | R/W | 0 | GPIO3 Pin 0 Data Output If p30_oe is set to 1, this field is used to control the output state of P3.0. 0: Output low if p30_oe is 1 1: Output high if p30_oe is 1. | |

3.9 Single Inductor Multiple Output (SIMO) Power Supply

The Single Inductor Multiple Output (SIMO) switch mode power supply allows the device to operate autonomously from a single lithium cell. The SIMO provides three buck switching regulators (VREGO_A thru VREGO_C). Each of the three regulator voltages can be controlled by either CPU individually. For the SIMO to operate properly, the three buck regulator outputs must drive the power supply pins of the device as follows in [Table 3-18](#).

3.9.1 Power Supply Monitor

The system also provides a power monitor that monitors the external power supplies relative to the on-chip bandgap voltage. The following power supplies are monitored:

- VCOREA (V_{COREA}) Digital Core Supply Voltage A for the Always-On Domain (AoD)
- VCOREB (V_{COREB}) Digital Core Supply Voltage B
- VDDIO (V_{DDIO}) GPIO Supply Voltage
- VDDIOH (V_{DDIOH}) GPIO High Supply Voltage
- VDDA (V_{DDA}) Always-On Domain Analog Supply Voltage
- VREGI (V_{REGI}) Input Supply Voltage, Battery

If the voltage drops below the trigger threshold, all registers and peripherals in that power domain are reset. This improves reliability and safety by guarding against a low voltage condition corrupting the contents of the registers and the device state.

Refer to the data sheet electrical characteristics for the trigger threshold values and power fail reset voltages.

Table 3-18: SIMO Power Supply Device Pin Connectivity

| SIMO Supply Output Pin | Connection | Device Power Supply Input Pin | Supply Monitor Reset Action |
|------------------------|------------|-------------------------------|-----------------------------|
| V_{REGO_A} | → | V_{DDA} | POR |
| V_{REGO_B} | → | V_{COREB} | POR |
| V_{REGO_C} | → | V_{COREA} | POR |
| - | - | V_{REGI} | POR |

| SIMO Supply Output Pin | Connection | Device Power Supply Input Pin | Supply Monitor Reset Action |
|------------------------|------------|-------------------------------|--|
| - | - | V _{DDIO} Power On | GPIO pad held in reset until the voltage rises above threshold |
| - | - | V _{DDIOH} Power On | GPIO pad held in reset until the voltage rises above threshold |
| - | - | V _{DDIO} | GPIO pad logic enters POR |
| - | - | V _{DDIOH} | GPIO pad logic enters POR |

3.9.2 Single Inductor Multiple Output Registers (SIMO)

See [Table 2-4: APB Peripheral Base Address Map](#) for the SIMO Controller Peripheral Base Address.

Table 3-19: SIMO Controller Register Summary

| Offset | Register | Access | Name |
|----------|---------------------------------------|--------|---|
| [0x0004] | SIMO_VREGO_A | R/W | Buck Voltage Regulator A Control Register |
| [0x0008] | SIMO_VREGO_B | R/W | Buck Voltage Regulator B Control Register |
| [0x000C] | SIMO_VREGO_C | R/W | Buck Voltage Regulator C Control Register |
| [0x0014] | SIMO_IPKA | RO | Reserved. Do not modify this register. |
| [0x0018] | SIMO_IPKB | RO | Reserved. Do not modify this register. |
| [0x001C] | SIMO_MAXTON | RO | Reserved. Do not modify this register. |
| [0x0020] | SIMO_ILOAD_A | RO | Reserved. Do not modify this register. |
| [0x0024] | SIMO_ILOAD_B | RO | Reserved. Do not modify this register. |
| [0x0028] | SIMO_ILOAD_C | RO | Reserved. Do not modify this register. |
| [0x0030] | SIMO_BUCK_ALERT_THR_A | RO | Reserved. Do not modify this register. |
| [0x0034] | SIMO_BUCK_ALERT_THR_B | RO | Reserved. Do not modify this register. |
| [0x0038] | SIMO_BUCK_ALERT_THR_C | RO | Reserved. Do not modify this register. |
| [0x0040] | SIMO_BUCK_OUT_READY | RO | Buck Regulator Output Ready Register |
| [0x0044] | SIMO_ZERO_CROSS_CAL_A | RO | Reserved. Do not modify this register. |
| [0x0048] | SIMO_ZERO_CROSS_CAL_B | RO | Reserved. Do not modify this register. |
| [0x004C] | SIMO_ZERO_CROSS_CAL_C | RO | Reserved. Do not modify this register. |

3.9.3 Single Inductor Multiple Output (SIMO) Registers Details

Table 3-20: SIMO Buck Voltage Regulator A Control Register

| SIMO Buck Voltage Regulator A Control | | | SIMO_VREGO_A | | [0x0004] |
|---------------------------------------|--------|--------|------------------------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | - | Reserved Do not modify this field. | |
| 7 | rangea | R/W | 1 | Regulator Output A Range Selects the Regulator output range for V _{REGO_A} . 0: 0.5V to 1.77 1: 0.6V to 1.87V | |

| SIMO Buck Voltage Regulator A Control | | | SIMO_VREGO_A | | [0x0004] |
|---------------------------------------|-------|--------|--------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 6:0 | vseta | R/W | 0x78h | <p>Regulator Output A Voltage Each bit increment in this field represents 10mV allowing output voltage settings from the minimum to the maximum of the SIMO_VREGO_A.rangea selected.</p> <p><i>SIMO_VREGO_A.rangea</i> = 1: Output Voltage = $0.6V + (10mV \times vseta)$ <i>SIMO_VREGO_A.rangea</i> = 0: Output Voltage = $0.5V + (10mV \times vseta)$</p> <p>Default: 0x78 = <i>SIMO_VREGO_A.rangea</i> = 0, Output Voltage = 1.7V; <i>SIMO_VREGO_A.rangea</i> = 1, Output Voltage = 1.8V</p> <p>Warning: When this regulator is connected as shown in Table 3-18: SIMO Power Supply Device Pin Connectivity, the following apply:</p> <ol style="list-style-type: none"> 1. The maximum setting for this regulator must be followed for V_{DDA} as indicated in the device datasheet. 2. Setting the regulator to a voltage below the Power-Fail Reset Voltage for V_{DDA} will initiate the Power Monitor Reset Action. | |

Table 3-21: SIMO Buck Voltage Regulator B Control Register

| SIMO Buck Voltage Regulator B Control | | | SIMO_VREGO_B | | [0x0008] |
|---------------------------------------|--------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | - | <p>Reserved Do not modify this field.</p> | |
| 7 | rangeb | R/W | 1 | <p>Regulator Output B Range Selects the Regulator output range for V_{REGO_B}.</p> <p>0: 0.5V to 1.77 1: 0.6V to 1.87V</p> | |
| 6:0 | vsetb | R/W | 0x32h | <p>Regulator Output Voltage Each bit increment in this field represents 10mV allowing output voltage settings from the minimum to the maximum of the SIMO_VREGO_B.rangeb selected.</p> <p><i>SIMO_VREGO_B.rangeb</i> = 1: Output Voltage = $0.6V + (10mV \times vsetb)$ <i>SIMO_VREGO_B.rangeb</i> = 0: Output Voltage = $0.5V + (10mV \times vsetb)$</p> <p>Setting this field to 0x7F results in the maximum output voltage per the <i>SIMO_VREGO_B.rangeb</i> selected (1.77V or 1.87V)</p> <p>Default: 0x32 = <i>SIMO_VREGO_B.rangeb</i> = 0, Output Voltage = 1.0V; <i>SIMO_VREGO_B.rangeb</i> = 1, Output Voltage = 1.1V</p> <p>Warning: When this regulator is connected as shown in Table 3-18: SIMO Power Supply Device Pin Connectivity, the following apply:</p> <ol style="list-style-type: none"> 1. The maximum setting for this regulator must be followed for V_{COREB} as indicated in the device datasheet. 2. Setting the regulator to a voltage below the Power-Fail Reset Voltage for V_{COREB} will initiate the Power Monitor Reset Action. | |

Table 3-22: SIMO Buck Voltage Regulator C Control Register

| SIMO Buck Voltage Regulator C Control | | | SIMO_VREGO_C | | [0x000C] |
|---------------------------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | - | <p>Reserved Do not modify this field.</p> | |

| SIMO Buck Voltage Regulator C Control | | | SIMO_VREGO_C | | [0x000C] |
|---------------------------------------|--------|--------|--------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 7 | rangec | R/W | 1 | Regulator Output Range Selects the Regulator output range for V _{REGO_C} . 0: 0.5V to 1.77 1: 0.6V to 1.87V | |
| 6:0 | vsetc | R/W | 0x32h | Regulator Output Voltage Each increment in the register represents 10mV. $SIMO_VREGO_C.rangec = 1; Output\ Voltage = 0.6V + (10mV \times vsetc)$ $SIMO_VREGO_C.rangec = 0; Output\ Voltage = 0.5V + (10mV \times vsetc)$ Setting this field to 0x7F results in the maximum output voltage per the <i>SIMO_VREGO_C.rangec</i> selected (1.77V or 1.87V) Default: 0x32 = <i>SIMO_VREGO_C.rangec</i> = 0, <i>Output Voltage</i> = 1.0V; <i>SIMO_VREGO_C.rangec</i> = 1, <i>Output Voltage</i> = 1.1V Warning: When this regulator is connected as shown in <i>Table 3-18: SIMO Power Supply Device Pin Connectivity</i> , the following apply: <ol style="list-style-type: none"> 1. The maximum setting for this regulator must be followed for V_{COREA} as indicated in the device datasheet. 2. Setting the regulator to a voltage below the Power-Fail Reset Voltage for V_{COREA} will initiate the Power Monitor Reset Action. | |

Table 3-23: SIMO High Side FET Peak Current VREGO_A VREGO_B Register

| SIMO High Side FET Peak Current VREGO_A VREGO_B | | | SIMO_IPKA | | [0x0014] |
|---|---------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | - | Reserved Reserved. Do not modify this field. | |
| 7:4 | ipksetb | R/W | 8 | Reserved Reserved. Do not modify this field. | |
| 3:0 | ipkseta | R/W | 8 | Reserved Reserved. Do not modify this field. | |

Table 3-24: SIMO High Side FET Peak Current VREGO_C Register

| SIMO High Side FET Peak Current VREGO_C VREGO_D | | | SIMO_IPKB | | [0x0018] |
|---|---------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | R/W | - | Reserved Do not modify this field. | |
| 3:0 | ipksetc | R/W | 8 | Reserved Reserved. Do not modify this field. | |

Table 3-25: SIMO Maximum High Side FET Time On Register

| SIMO Maximum High Side FET Time On | | | | SIMO_MAXTON | [0x001C] |
|------------------------------------|--------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | R/W | - | Reserved Do not modify this field. | |
| 3:0 | tonset | R/W | 0x8h | Reserved Do not modify this field. | |

Table 3-26: SIMO Buck Cycle Count VREGO_A Register

| SIMO Buck Cycle Count VREGO_A | | | | SIMO_ILOAD_A | [0x0020] |
|-------------------------------|--------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | iloada | RO | 0 | Reserved Do not modify this field. | |

Table 3-27: SIMO Buck Cycle Count VREGO_B Register

| SIMO Buck Cycle Count VREGO_B | | | | SIMO_ILOAD_B | [0x0024] |
|-------------------------------|--------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | iloadb | RO | 0 | Reserved Do not modify this field. | |

Table 3-28: SIMO Buck Cycle Count VREGO_C Register

| SIMO Buck Cycle Count VREGO_C | | | | SIMO_ILOAD_C | [0x0028] |
|-------------------------------|--------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | iloadc | RO | 0 | Reserved Do not modify this field. | |

Table 3-29: SIMO Buck Cycle Count Alert VREGO_A Register

| SIMO Buck Cycle Count Alert VREGO_A | | | | SIMO_BUCK_ALERT_THR_A | [0x0030] |
|-------------------------------------|----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | buckthra | R/W | 0 | Reserved Do not modify this field. | |

Table 3-30: SIMO Buck Cycle Count Alert VREGO_B Register

| SIMO Buck Cycle Count Alert VREGO_A | | | | SIMO_BUCK_ALERT_THR_B | [0x0034] |
|-------------------------------------|----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | buckthrb | R/W | 0 | Reserved Do not modify this field. | |

Table 3-31: SIMO Buck Cycle Count Alert VREGO_C Register

| SIMO Buck Cycle Count Alert VREGO_A | | | | SIMO_BUCK_ALERT_THR_C | [0x0038] |
|-------------------------------------|----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | buckthrc | R/W | 0 | Reserved Do not modify this field. | |

Table 3-32: SIMO Buck Regulator Output Ready Register

| SIMO Buck Regulator Output Ready | | | | SIMO_BUCK_OUT_READY | [0x0040] |
|----------------------------------|-------------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | - | Reserved Do not modify this field. | |
| 3 | buckoutrdya | RO | 0 | VREGO_A Output Ready When SIMO_VREGO_A.vseta changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready | |
| 2 | buckoutrdyb | RO | 0 | VREGO_B Output Ready When SIMO_VREGO_B.vsetb changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready | |
| 1 | buckoutrdyc | RO | 0 | VREGO_C Output Ready When SIMO_VREGO_C.vsetc changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready | |
| 0 | - | RO | 0 | Reserved Do not modify | |

Table 3-33: SIMO Zero Cross Calibration VREGO_A Register

| SIMO Zero Cross Calibration VREGO_A | | | | SIMO_ZERO_CROSS_CAL_A | [0x0044] |
|-------------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | RO | - | Reserved Do not modify this field. | |
| 4:0 | zxcla | RO | 0 | Reserved Do not modify this field. | |

Table 3-34: SIMO Zero Cross Calibration VREGO_B Register

| SIMO Zero Cross Calibration VREGO_B | | | | SIMO_ZERO_CROSS_CAL_B | [0x0048] |
|-------------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | RO | - | Reserved Do not modify this field. | |
| 4:0 | zxclb | RO | 0 | Reserved Do not modify this field. | |

Table 3-35: SIMO Zero Cross Calibration VREGO_C Register

| SIMO Zero Cross Calibration VREGO_C | | | | SIMO_ZERO_CROSS_CAL_C | [0x004C] |
|-------------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | RO | - | Reserved Do not modify this field. | |
| 4:0 | zxclc | RO | 0 | Reserved Do not modify this field. | |
| 4:0 | zxcld | RO | 0 | Reserved Do not modify this field. | |

3.10 Low-Power General Control Registers (LPGCR)

This set of general control registers provides reset and clock control for the low-power peripherals including:

- LPUART0 (UART3)
- LPTMR0 (TMR4)
- LPTMR1 (TMR5)
- LPWDT0 (WDT1)
- LPCOMP1, LPCOMP2 and LPCOMP3
- GPIO2

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-36 Low-Power Control Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 3-36 Low-Power Control Register Summary

| Offset | Register | Name |
|----------|-------------------------------|------------------------|
| [0x0004] | LPGCR_RST | Reset Control Register |
| [0x0008] | LPGCR_PCLKDIS | Clock Control Register |

3.10.1 Low-Power General Control Registers Details

Table 3-37: Reset Control Register

| Low-Power Reset Control | | | LPGCR_RST | [0x0004] |
|-------------------------|--------|--------|-----------|--|
| Bits | Field | Access | Reset | Description |
| 31:7 | - | RO | 0 | Reserved Do not modify |
| 6 | lpcomp | W1O | 0 | Low Power Comparators Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. See Device Resets for additional information. |
| 5 | - | RO | 0 | Reserved Do not modify |
| 4 | uart3 | W1O | 0 | UART3 (LPUART0) Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. See Device Resets for additional information. |
| 3 | tmr5 | W1O | 0 | TMR5 (LPTMR1) Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. See Device Resets for additional information. |
| 2 | tmr4 | W1O | 0 | TMR4 (LPTMR0) Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. See Device Resets for additional information. |
| 1 | wdt1 | W1O | 0 | WDT1 (LPWDT0) Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. See Device Resets for additional information. |
| 0 | gpio2 | W1O | 0 | GPIO2 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. See Device Resets for additional information. |

Table 3-38: Clock Disable Register

| Clock Disable Register | | | LPGCR_PCLKDIS | [0x008] |
|------------------------|--------|--------|---------------|--|
| Bits | Field | Access | Reset | Description |
| 31:7 | - | RO | 0 | Reserved Do not modify |
| 6 | lpcomp | R/W | 0 | Low Power Comparators Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. <i>Note: This field disables clocks to LPCOMP1, LPCOMP2 and LPCOMP3.</i> 0: Enabled 1: Disabled |

| Clock Disable Register | | | | LPGCR_PCLKDIS | [0x008] |
|------------------------|-------|--------|-------|--|---------|
| Bits | Field | Access | Reset | Description | |
| 5 | - | RO | 0 | Reserved Do not modify | |
| 4 | uart3 | R/W | 0 | UART3 (LPUART0) Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Enabled 1: Disabled | |
| 3 | tmr5 | R/W | 0 | TMR5 (LPTMR1) Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Enabled 1: Disabled | |
| 2 | tmr4 | R/W | 0 | TMR4 (LPTMR0) Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Enabled 1: Disabled | |
| 1 | wdt1 | R/W | 0 | WDT1 (LPWDT0) Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Enabled 1: Disabled | |
| 0 | gpio2 | R/W | 0 | GPIO2 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Enabled 1: Disabled | |

3.11 Power Sequencer Registers (PWRSEQ)

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-39: Power Sequencer Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 3-39: Power Sequencer Register Summary

| Offset | Register | Name |
|----------|--------------------------------|---|
| [0x0000] | PWRSEQ_LPCN | <i>Low Power Control Register</i> |
| [0x0004] | PWRSEQ_LPWKST0 | <i>Low Power GPIO0 Wakeup Status Flags</i> |
| [0x0008] | PWRSEQ_LPWKENO | <i>Low Power GPIO0 Wakeup Enable Register</i> |
| [0x000C] | PWRSEQ_LPWKST1 | <i>Low Power GPIO1 Wakeup Status Flags</i> |
| [0x0010] | PWRSEQ_LPWKEN1 | <i>Low Power GPIO1 Wakeup Enable Register</i> |
| [0x0014] | PWRSEQ_LPWKST2 | <i>Low Power GPIO2 Wakeup Status Flags</i> |

| Offset | Register | Name |
|----------|----------------|---|
| [0x0018] | PWRSEQ_LPWKEN2 | Low Power GPIO2 Wakeup Enable Register |
| [0x001C] | PWRSEQ_LPWKST3 | Low Power GPIO3 Wakeup Status Flags |
| [0x0020] | PWRSEQ_LPWKEN3 | Low Power GPIO3 Wakeup Enable Register |
| [0x0030] | PWRSEQ_LPPWST | Low Power Peripheral Wakeup Status Register |
| [0x0034] | PWRSEQ_LPPWEN | Low Power Peripheral Wakeup Enable Register |
| [0x0048] | PWRSEQ_GPO | General Purpose Register 0 |
| [0x004C] | PWRSEQ_GPI | General Purpose Register 1 |

3.11.1 Power Sequencer Register Details

Table 3-40: Low Power Control Register

| Low Power Control | | PWRSEQ_LPCN | | [0x0000] |
|-------------------|------------|-------------|-------|---|
| Bits | Field | Access | Reset | Description |
| 31 | lpwkst_clr | R/W1 | 0 | Low Power Wakeup Status Register Clear Write 1 to this field to clear the Low Power Wakeup Status registers: <ul style="list-style-type: none"> • PWRSEQ_LPWKST0 • PWRSEQ_LPWKST1 • PWRSEQ_LPWKST2 • PWRSEQ_LPWKST3 • PWRSEQ_LPPWST 1: Write 1 to initiate a clear of all the Low Power Wakeup Status registers. Hardware automatically clears this field when the registers are cleared. |
| 30:12 | - | DNM | 0 | Reserved Do not modify This field must always be set to 0 for future compatibility. |
| 11 | bg_dis | R/W | 1 | Band Gap Disable for LPM and BACKUP Mode Setting this field to 1 (default) disables the Bandgap during LPM and BACKUP mode. 0: System Bandgap is on in LPM and BACKUP modes 1: System Bandgap is off in LPM and BACKUP modes. |
| 10 | - | R/W | 0 | Reserved Do not modify |
| 9 | lpmfast | R/W | 0 | Low Power Mode Clock Select If the ISO is selected (default), fast LPM entry is enabled. Setting the clock to INRO disables fast LPM entry. 0: ISO used for entering LPM (Fast Mode Enable). 1: INRO used for LPM entry (Fast Mode Disabled). |
| 8 | lpmclksel | R/W | 1 | Low Power Mode APB Clock Select This field selects the clock source to use for the RV32 (CPU1) and other APB peripherals during LPM . 0: PCLK is used as the RV32 (CPU1) and APB system clock during LPM . 1: ISO is used as the RV32 (CPU1) and APB system clock during LPM . |
| 7:4 | - | R/W | 0 | Reserved for Future Use Do not modify <i>Note: This field must be set to 0 to maintain future compatibility.</i> |

| Low Power Control | | | | PWRSEQ_LPCN | [0x0000] |
|-------------------|---------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 3 | ramret3 | R/W | 0 | System RAM 3 Data Retention Enable for BACKUP Mode Set this field to 1 to enable Data Retention for System RAM 3. See SRAM Space for System RAM configuration. 0: Disable data retention for System RAM 3 address space in BACKUP . 1: Enable data retention for System RAM 3 address space in BACKUP . | |
| 2 | ramret2 | R/W | 0 | System RAM 2 Data Retention Enable for BACKUP Mode Set this field to 1 to enable Data Retention for System RAM 2. See SRAM Space for System RAM configuration. 0: Disable data retention for System RAM 2 address space in BACKUP . 1: Enable data retention for System RAM 2 address space in BACKUP mode. | |
| 1 | ramret1 | R/W | 0 | System RAM 1 Data Retention Enable for BACKUP Mode Set this field to 1 to enable Data Retention for System RAM 1. See SRAM Space for System RAM configuration. 0: Disable data retention for System RAM 1 address space in BACKUP mode. 1: Enable data retention for System RAM 1 address space in BACKUP mode. | |
| 0 | ramret0 | R/W | 0 | System RAM 0 Data Retention Enable for BACKUP Mode Set this field to 1 to enable Data Retention for System RAM 0. See SRAM Space for System RAM configuration. 0: Disable data retention for System RAM 0 address space in BACKUP mode. 1: Enable data retention for System RAM 0 address space in BACKUP mode. | |

Table 3-41: GPIO0 Low Power Wakeup Status Flags

| GPIO0 Low Power Wakeup Status Flags | | | | PWRSEQ_LPWKST0 | [0x0004] |
|-------------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | st | R/W1C | 0 | GPIO0 Pin Wakeup Status Flag Whenever a GPIO0 pin, in any power mode, transitions from low-to-high or high-to-low, the pin's corresponding bit in this register is set. The device will transition from a low-power mode to ACTIVE mode if the corresponding GPIO pin's interrupt enable bit is set in the PWRSEQ_LPWKENO register. <i>Note: Clear this register before entering any low power mode.</i> | |

Table 3-42: GPIO0 Low Power Wakeup Enable Registers

| GPIO0 Low Power Wakeup Enable | | | | PWRSEQ_LPWKENO | [0x0008] |
|-------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | en | R/W | 0 | GPIO0 Pin Wakeup Interrupt Enable Setting a GPIO0 pin's bit in this register will cause an interrupt to be generated that will wake up the device from any low power mode to ACTIVE mode. A wakeup event sets the corresponding GPIO0's bit in the PWRSEQ_LPWKST0 register, enabling determination of which GPIO pin triggered the wakeup event. Bits corresponding to unimplemented GPIO are ignored. <i>Note: To enable the MAX78000 to wake up from a low power mode on a GPIO pin transition, first set the "GPIO Wakeup Enable" register bit GCR_PM gpio_we = 1.</i> | |

Table 3-43: GPIO1 Low Power Wakeup Status Flags

| GPIO1 Low Power Wakeup Status Flags | | | PWRSEQ_LPWKST1 | | [0x000C] |
|-------------------------------------|-------|--------|----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1C | 0 | Reserved Bits corresponding to unimplemented GPIO are ignored. | |
| 9:0 | st | R/W1C | 0 | GPIO1 Pin Wakeup Status Flag Whenever a GPIO1 pin, in any power mode, transitions from low-to-high or high-to-low, the pin's corresponding bit in this register is set. The device will transition from a low-power to ACTIVE mode if the corresponding interrupt enable bit is set in PWRSEQ_LPWKEN1 . <i>Note: Clear this register before entering any low power mode.</i> | |

Table 3-44: GPIO1 Low Power Wakeup Enable Registers

| GPIO1 Low Power Wakeup Enable | | | PWRSEQ_LPWKEN1 | | [0x0010] |
|-------------------------------|-------|--------|----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:10 | | RO | 0 | Reserved Bits corresponding to unimplemented GPIO are ignored. | |
| 9:0 | en | R/W | 0 | GPIO1 Pin Wakeup Interrupt Enable Setting a GPIO1 pin's bit in this register will cause an interrupt to be generated that will wakeup the device from any low power mode to ACTIVE mode. A wakeup event sets the corresponding GPIO1's bit in the PWRSEQ_LPWKST1 register, enabling determination of which GPIO1 pin triggered the wakeup event. Bits corresponding to unimplemented GPIO are ignored. <i>Note: To enable the MAX78000 to wake up from a low power mode on a GPIO pin transition, first set the "GPIO Wakeup Enable" register bit GCR_PM gpio_we = 1.</i> | |

Table 3-45: GPIO2 Low Power Wakeup Status Flags

| GPIO2 Low Power Wakeup Status Flags | | | PWRSEQ_LPWKST2 | | [0x0014] |
|-------------------------------------|-------|--------|----------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | | R/W1C | 0 | Reserved Bits corresponding to unimplemented GPIO are ignored. | |
| 7:0 | st | R/W1C | 0 | GPIO2 Pin Wakeup Status Flag Whenever a GPIO2 pin, in any power mode, transitions from low-to-high or high-to-low, the corresponding bit in this register is set. Bits corresponding to unimplemented GPIO are ignored. <i>Note: The device will transition from a low-power to ACTIVE mode if the corresponding interrupt enable bit is set in PWRSEQ_LPWKEN. This register should be cleared before entering any low power mode.</i> | |

Table 3-46: GPIO2 Low Power Wakeup Enable Registers

| GPIO2 Low Power Wakeup Enable | | | PWRSEQ_LPWKEN2 | | [0x0018] |
|-------------------------------|-------|--------|----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | | RO | 0 | Reserved Bits corresponding to unimplemented GPIO are ignored. | |
| 7:0 | en | R/W | 0 | GPIO2 Pin Wakeup Interrupt Enable Setting a GPIO2 pin's bit in this register will cause an interrupt to be generated that will wakeup the device from any low power mode to ACTIVE mode. A wakeup event sets the corresponding GPIO2's bit in the PWRSEQ_LPWKST2 register, enabling determination of which GPIO2 pin triggered the wakeup event. Bits corresponding to unimplemented GPIO are ignored. <i>Note: To enable the MAX78000 to wake up from a low power mode on a GPIO pin transition, first set the "GPIO Wakeup Enable" register bit GCR_PM gpio_we = 1.</i> | |

Table 3-47: GPIO3 Low Power Wakeup Status Flags

| GPIO3 Low Power Wakeup Status Flags | | | PWRSEQ_LPWKST3 | | [0x001C] |
|-------------------------------------|-------|--------|----------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:2 | | RO | 0 | Reserved Bits corresponding to unimplemented GPIO are ignored. | |
| 1:0 | st | R/W1C | 0 | GPIO3 Pin Wakeup Status Flag Whenever a GPIO3 pin, in any power mode, transitions from low-to-high or high-to-low, the corresponding bit in this register is set. Bits corresponding to unimplemented GPIO are ignored. <i>Note: The device will transition from a low-power to ACTIVE mode if the corresponding interrupt enable bit is set in PWRSEQ_LPWKEN. This register should be cleared before entering any low power mode.</i> | |

Table 3-48: GPIO3 Low Power Wakeup Enable Registers

| GPIO3 Low Power Wakeup Enable | | | PWRSEQ_LPWKEN3 | | [0x0020] |
|-------------------------------|-------|--------|----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:2 | | RO | 0 | Reserved Do not modify | |
| 1:0 | en | R/W | 0 | GPIO3 Pin Wakeup Interrupt Enable Setting a GPIO3 pin's bit in this register will cause an interrupt to be generated that will wakeup the device from any low power mode to ACTIVE mode. A wakeup event sets the corresponding GPIO3's bit in the PWRSEQ_LPWKST3 register, enabling determination of which GPIO3 pin triggered the wakeup event. Bits corresponding to unimplemented GPIO are ignored. <i>Note: To enable the MAX78000 to wake up from a low power mode on a GPIO pin transition, first set the "GPIO Wakeup Enable" register bit GCR_PM gpio_we = 1.</i> | |

Table 3-49: Low Power Peripheral Wakeup Status Flags

| Low Power Peripheral Wakeup Status Flags | | | PWRSEQ_LPPWST | | [0x0030] |
|--|--------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:18 | | RO | 0 | Reserved Do not modify | |
| 17 | reset | R/W1C | 0 | Reset Detected Wakeup Flag This field is set when an external reset caused the wakeup event. | |
| 16 | backup | R/W1C | 0 | BACKUP Mode Wakeup Flag This field is set when the device wakes up from BACKUP mode. | |
| 15:5 | - | RO | 0 | Reserved Do not modify | |
| 4 | comp0 | R/W1C | 0 | Comparator 0 Wakeup Flag This field is set if the wakeup event was the result of a comparator 0 trigger event. | |
| 3:0 | - | RO | 0 | Reserved Do not modify | |

Table 3-50: Low Power Peripheral Wakeup Enable Registers

| Low Power Peripheral Wakeup Enable | | | PWRSEQ_LPPWEN | | [0x0034] |
|------------------------------------|--------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:27 | | RO | 0 | Reserved Do not modify | |
| 26 | lpcomp | R/W | 0 | Low Power Comparator IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the LPCOMPn IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 25 | spi1 | R/W | 0 | SPI1 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the SPI1 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 24 | i2s | R/W | 0 | I2S IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the I2S IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 23 | i2c2 | R/W | 0 | I2C2 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the I2C2 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 22 | i2c1 | R/W | 0 | I2C1 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the I2C1 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |

| Low Power Peripheral Wakeup Enable | | | PWRSEQ_LPPWEN | | [0x0034] |
|------------------------------------|-------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 21 | i2c0 | R/W | 0 | I2C0 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the I2C0 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 20 | uart3 | R/W | 0 | LPUART0 (UART3) IRQ Wakeup Enable Set this field to 1 to enable wakeup events from LPUART0 (UART3) IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 19 | uart2 | R/W | 0 | UART2 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the UART2 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 18 | uart1 | R/W | 0 | UART1 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the UART1 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 17 | uart0 | R/W | 0 | UART0 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the UART0 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 16 | tmr5 | R/W | 0 | LPTMR1 (TMR5) IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the LPTMR1 (TMR5) IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 15 | tmr4 | R/W | 0 | LPTMR0 (TMR4) IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the LPTMR0 (TMR4) IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 14 | tmr3 | R/W | 0 | TMR3 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the TMR3 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 13 | tmr2 | R/W | 0 | TMR2 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the TMR2 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 12 | tmr1 | R/W | 0 | TMR1 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the TMR1 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 11 | tmr0 | R/W | 0 | TMR0 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the TMRO IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |

| Low Power Peripheral Wakeup Enable | | | PWRSEQ_LPPWEN | | [0x0034] |
|------------------------------------|-------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 10 | cpu1 | R/W | 0 | CPU1 (RV32) IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the RV32 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 9 | wdt1 | R/W | 0 | WDT1 (LPWDTO) IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the WDT1 (LPWDTO) IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 8 | wdt0 | R/W | 0 | WDT0 IRQ Wakeup Enable Set this field to 1 to enable wakeup events from the WDT0 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 7:5 | - | RO | 0 | Reserved Do not modify | |
| 4 | comp0 | R/W | 0 | Comparator 0 Wakeup Enable Set this field to 1 to enable wakeup events from Comparator 0. Comparator 0 can wake the device up from SLEEP , LPM , UPM , STANDBY and BACKUP . 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ. | |
| 3:0 | - | RO | 0 | Reserved Do not modify | |

Table 3-51: Low Power General Purpose Register 0

| Low Power General Purpose Register 0 | | | PWRSEQ_GPO | | [0x0048] |
|--------------------------------------|-------|--------|------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W | 0 | General Purpose Field This register can be used as a general-purpose register by software and retains the contents during SLEEP , LPM , UPM , STANDBY and BACKUP modes. | |

Table 3-52: Low Power General Purpose Register 1

| Low Power General Purpose Register 1 | | | PWRSEQ_GP1 | | [0x004C] |
|--------------------------------------|-------|--------|------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W | 0 | General Purpose Field This register can be used as a general-purpose register by software and retains the contents during SLEEP , LPM , UPM , STANDBY and BACKUP modes. | |

3.12 Trim System Initialization Registers (TRIMSIR)

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-53: Trim System Initialization Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Note: The TRIMSIR registers are reset only on a Power-On-Reset. System Reset, Soft Reset and Peripheral Reset do not affect the TRIMSIR register values.

Table 3-53: Trim System Initialization Register Summary

| Offset | Register Name | Access | Description |
|----------|-------------------------------|--------|--|
| [0x0008] | TRIMSIR_RTC | R/W | RTC Trim System Initialization Register |
| [0x0034] | TRIMSIR_SIMO | R/W | System Initialization Register |
| [0x003C] | TRIMSIR_IPOLO | R/W | System initialization Function Status Register |
| [0x0040] | TRIMSIR_CTRL | R/W | Control Trim System Initialization Register |
| [0x0044] | TRIMSIR_INRO | R/W | INRO Trim System Initialization Register |

3.12.1 TRIM System Initialization Register Details

Table 3-54: RTC Trim System Initialization Register

| RTC Trim System Initialization | | | TRIMSIR_RTC | | [0x0008] |
|--------------------------------|--------|--------|-------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31 | lock | RO | 0 | Lock If this field is set to 1, this register is read-only and the RTC X1 and RTC X2 fields cannot be modified. | |
| 30:26 | - | RO | 0 | Reserved for Future Use Do not modify | |
| 25:21 | x2trim | R/W* | 0 | RTC X2 Trim The X2 trim setting for the RTC. <i>Note: If TRIMSIR_RTC.lock is set to 1, this field is read-only.</i> | |
| 20:16 | x1trim | R/W* | 0 | RTC X1 Trim The X1 trim setting for the RTC. <i>Note: If TRIMSIR_RTC.lock is set to 1, this field is read-only.</i> | |
| 15:0 | - | RO | 0 | Reserved for Future Use Do not modify | |

Table 3-55: SIMO Trim System Initialization Register

| SIMO System Initialization | | | TRIMSIR_SIMO | | [0x0034] |
|----------------------------|------|--------|--------------|----------------------------------|----------|
| Bits | Name | Access | Reset | Description | |
| 31:3 | - | RO | 0 | Reserved Do not modify | |

| SIMO System Initialization | | | TRIMSir_SIMO | | [0x0034] |
|----------------------------|--------|--------|--------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 2:0 | clkdiv | R/W | 1 | SIMO Clock Divide This field selects the SIMO clock divisor. The SIMO uses the INRO as its input clock. 0: $\frac{INRO}{1}$ 1: $\frac{INRO}{16}$ 2: Reserved for Future Use 3: $\frac{INRO}{32}$ 4: Reserved for Future Use 5: $\frac{INRO}{64}$ 6: Reserved for Future Use 7: $\frac{INRO}{128}$ | |

Table 3-56: IPO Low Trim System Initialization Register

| IPO Trim Low System Initialization | | | TRIMSir_IPOLO | | [0x003C] |
|------------------------------------|-------------|--------|-----------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved Do not modify | |
| 7:0 | ipo_limitlo | RO | See Description | IPO Low Trim Limit This field contains the low trim limit for the IPO. | |

Table 3-57: Control Trim System Initialization Register

| Control System Initialization | | | TRIMSir_CTRL | | [0x0040] |
|-------------------------------|--------------|--------|-----------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:29 | inro_trim | R/W | See Description | INRO Clock Trim This field contains the trim for the INRO when set to 8KHz. | |
| 28:26 | - | RO | 0 | Reserved Do not modify | |
| 25:24 | inro_sel | R/W | 2 | INRO Clock Select Selects the INRO frequency. 0: 8KHz 1: 16KHz 2: 30KHz (Power-On Reset default) 3: Reserved for Future Use | |
| 23:15 | ipo_limithi | R/W | 0x1FF | IPO High Trim Limit This field contains the high limit for the IPO. | |
| 14:8 | vdda_limithi | R/W | 0x78 | VDDA High Trim Limit High trim limit for VDDA. | |
| 7 | - | RO | 0 | Reserved Do not modify | |

| Control System Initialization | | | TRIMSI_R_CTRL | | [0x0040] |
|-------------------------------|--------------|--------|---------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 6:0 | vdda_limitlo | R/W | 0x64 | V_{DDA} Low Trim Limit Low trim limit for V _{DDA} | |

Table 3-58: INRO Trim System Initialization Register

| INRO System Initialization | | | TRIMSI_R_INRO | | [0x0044] |
|----------------------------|----------|--------|---------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved Do not modify | |
| 7:6 | lpclksel | R/W | 2 | INRO Low Power Mode Clock Select Selects the INRO clock frequency for LPM operation. 0: 8KHz 1: 16KHz 2: 30KHz (POR default) 3: Reserved for Future Use | |
| 5:3 | trim30k | R/W | 0 | INRO 30KHz Trim This field contains the trim for INRO when set to 30KHz. | |
| 2:0 | trim16k | R/W | 0 | INRO 16KHz Trim This field contains the trim for INRO when set to 16KHz. | |

3.13 Global Control Registers (GCR)

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-59: Global Control Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Note: The General Control Registers are only reset on a System Reset or Power-On Reset. A Soft Reset or Peripheral Reset does not affect these registers.

Table 3-59: Global Control Register Summary

| Offset | Register | Description |
|----------|------------------------------|-----------------------------|
| [0x0000] | GCR_SYSCTRL | System Control Register |
| [0x0004] | GCR_RST0 | Reset Register 0 |
| [0x0008] | GCR_CLKCTRL | Clock Control Register |
| [0x000C] | GCR_PM | Power Management Register |
| [0x0018] | GCR_PCLKDIV | Peripheral Clocks Divisor |
| [0x0024] | GCR_PCLKDIS0 | Peripheral Clocks Disable 0 |
| [0x0028] | GCR_MEMCTRL | Memory Clock Control |
| [0x002C] | GCR_MEMZ | Memory Zeroize Register |
| [0x0040] | GCR_SYST | System Status Flags |
| [0x0044] | GCR_RST1 | Reset Register 1 |

| Offset | Register | Description |
|----------|------------------------------|---|
| [0x0048] | GCR_PCLKDIS1 | <i>Peripheral Clocks Disable 1</i> |
| [0x004C] | GCR_EVENTEN | <i>Event Enable Register</i> |
| [0x0050] | GCR_REVISION | <i>Revision Register</i> |
| [0x0054] | GCR_SYSIE | <i>System Status Interrupt Enable</i> |
| [0x0064] | GCR_ECCERR | <i>Error Correction Coding Error Register</i> |
| [0x0068] | GCR_ECCCED | <i>Error Correction Coding Correctable Error Detected</i> |
| [0x006C] | GCR_ECCIE | <i>Error Correction Coding Interrupt Enable Register</i> |
| [0x0070] | GCR_ECCADDR | <i>Error Correction Coding Error Address Register</i> |
| [0x0080] | GCR_GPRO | <i>General Purpose Register 0</i> |

3.13.1 Global Control Register Details (GCR)

Table 3-60: System Control Register

| System Control | | | GCR_SYSCTRL | | [0x0000] |
|----------------|--------|--------|-----------------------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:18 | - | RO | 0 | Reserved Do not modify | |
| 17:16 | ovr | R/W | 0b10 | Operating Voltage Range Set this field to match the V _{COREA} voltage to enable the on-chip RAM to operate at the optimal timing range. 0b00: 0.9V ± 10% 0b01: 1.0V ± 10% 0b10: 1.1V ± 10% 0b11: Reserved for Future Use | |
| 15 | chkres | R | 0 | ROM Checksum Calculation Pass/Fail This is the result after setting bit GCR_SYSCTRL.cchk . This bit is only valid after the ROM checksum is complete and GCR_SYSCTRL.cchk is cleared. 0: Pass 1: Fail | |
| 14 | sw_dis | R/W | 0 | Serial Wire Debug Disable This bit is used to disable the serial wire debug interface. 0: SWD Disabled 1: SWD Enabled <i>Note: This bit is only writeable if the flash is not factory locked or if the GCR_SYSST.icelock bit is 0 and the GCR_SYSCTRL.rom_done bit is 1.</i> | |
| 13 | cchk | R/W | 0 | Calculate ROM Checksum This bit is self-clearing when the ROM checksum calculation is complete, and the result is available at bit GCR_SYSCTRL.chkres . Writing a 0 has no effect. 0: No operation 1: Start ROM checksum calculation | |

| System Control | | | | GCR_SYSCTRL | [0x0000] |
|----------------|-----------------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 12 | romdone | R/W | 0 | ROM Start Code Status Used to disable SWD interface during system initialization procedure. 0: ROM Start Code not completed 1: ROM Start Code not completed <i>Note: If the flash is factory locked, software can only set this bit. Subsequently, this bit cannot be cleared by software.</i> | |
| 11:7 | - | RO | 0 | Reserved Do not modify | |
| 6 | icc0_flush | R/W | 0 | ICCO Cache Flush Write 1 to flush the code cache and the instruction buffer for the M4. This bit is automatically cleared to 0 when the flush is complete. Writing 0 has no effect and does not stop a cache flush in progress. 0: ICCO flush complete. 1: Flush the contents of the ICCO cache. | |
| 5 | - | RO | 0 | Reserved Do not modify | |
| 4 | flash_page_flip | R/* | 0 | Flash Page Flip Flag Flips the bottom and top halves of Flash memory. This bit is controlled by hardware. <i>Note: Software/Firmware should not change the state of this bit during normal operation. Any change to this bit also flushes both code and data caches.</i> 0: Physical layout matches logical layout 1: Top and Bottom halves flipped | |
| 3:1 | - | RO | 1 | Reserved Do not modify | |
| 0 | bstapen | RO | * | Boundary Scan Tap Enable This field's reset value matches GCR_SYSST.icelock . Do not modify | |

Table 3-61: Reset Register 0

| Reset 0 | | | | GCR_RST0 | [0x0004] |
|---------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31 | sys | R/W | 0 | System Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. See System Reset for additional information. | |
| 30 | periph | R/W | 0 | Peripheral Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. <i>Note: Watchdog Timers, GPIO Ports, the AoD, RAM Retention and the General Control Registers (GCR) are unaffected. See Peripheral Reset for additional information.</i> | |
| 29 | soft | R/W | 0 | Soft Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. See Soft Reset for additional information. | |
| 28 | uart2 | R/W | 0 | UART2 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |

| Reset 0 | | | GCR_RST0 | | [0x0004] |
|---------|--------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 27 | - | R/W | 0 | Reserved Do not modify | |
| 26 | adc | R/W | 0 | ADC Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 25 | cnn | R/W | 0 | CNN Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 24 | trng | R/W | 0 | TRNG Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 23 | - | R/W | 0 | Reserved Do not modify | |
| 22 | smpthr | R/W | 0 | Semaphore Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 21:18 | - | R/W | 0 | Reserved Do not modify | |
| 17 | rtc | R/W | 0 | RTC Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 16 | i2c0 | R/W | 0 | I2C0 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 15:14 | - | RO | 0 | Reserved Do not modify | |
| 13 | spi1 | R/W | 0 | SPI1 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 12 | uart1 | R/W | 0 | UART1 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 11 | uart0 | R/W | 0 | UART0 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 10:9 | - | R/W | 0 | Reserved Do not modify | |
| 8 | tmr3 | R/W | 0 | TMR3 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 7 | tmr2 | R/W | 0 | TMR2 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 6 | tmr1 | R/W | 0 | TMR1 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 5 | tmr0 | R/W | 0 | TMR0 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |

| Reset 0 | | | | GCR_RST0 | [0x0004] |
|---------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 4 | - | RO | - | Reserved Do not modify | |
| 3 | gpio1 | R/W | 0 | GPIO1 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 2 | gpio0 | R/W | 0 | GPIO0 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 1 | wdt0 | R/W | 0 | Watchdog Timer 0 Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |
| 0 | dma | R/W | 0 | DMA Access Block Reset Write 1 to reset. This field is cleared by hardware when the reset is complete. | |

Table 3-62: Clock Control Register

| Clock Control | | | | GCR_CLKCTRL | [0x0008] |
|---------------|-----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:30 | - | RO | 0b10 | Reserved Do not modify | |
| 29 | inro_rdy | | 0 | 8kHz Internal Nano-Ring Oscillator (INRO) Ready Status 0: Not ready or not enabled. 1: Oscillator ready. | |
| 28 | ibro_rdy | R | 0 | 7.3728MHz Internal Baud Rate Oscillator (IBRO) Ready Status 0: Not ready. 1: Oscillator ready. | |
| 27 | ipo_rdy | R | 0 | 100MHz Internal Primary Oscillator (IPO) Ready Status 0: Not ready or not enabled. 1: Oscillator ready. | |
| 26 | iso_rdy | R | 0 | 60MHz Internal Secondary Oscillator (ISO) Ready Status 0: Not ready or not enabled. 1: Oscillator ready. | |
| 25 | ertco_rdy | R | 0 | 32.768kHz External RTC Oscillator (ERTCO) Ready Status 0: Not ready or not enabled. 1: Oscillator ready. | |
| 24:22 | - | RO | 0 | Reserved Do not modify | |
| 21 | ibro_vs | R/W | 0 | 7.3728MHz IBRO Power Supply Select 0: IBRO is powered from V _{COREA} 1: IBRO is powered using a dedicated 1V regulated internal supply | |
| 20 | ibro_en | RO | 1 | 7.3728MHz IBRO Enable The IBRO is always enabled. 1: Enabled and ready when GCR_CLKCTRL.ibro_rdy = 1. | |
| 19 | ipo_en | R/W | 0 | 100MHz IPO Enable 0: Disabled 1: Enabled and ready when GCR_CLKCTRL.ipot_rdy = 1. | |

| Clock Control | | | GCR_CLKCTRL | | [0x0008] |
|---------------|------------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 18 | iso_en | R/W | 1 | 60MHz ISO Enable Set this field to 0 to disable the ISO. The ISO is the System Oscillator (SYS_OSC) after a POR or System Reset. 0: Disabled 1: Enabled and ready when GCR_CLKCTRL.iso_rdy = 1 | |
| 17 | ertco_en | R/W | 0 | 32.768kHz ERTCO Enable 0: Disabled if the RTC_CTRL.en field is also set to 0. 1: Enabled and ready when GCR_CLKCTRL.ertco_rdy = 1, regardless of the state of the RTC_CTRL.en field. | |
| 16:14 | - | RO | 0 | Reserved Do not modify | |
| 13 | sysclk_rdy | R | 0 | SYS_OSC Select Ready When SYS_OSC is changed by modifying GCR_CLKCTRL.sysclk_sel , there is a delay until the switchover is complete. This bit is cleared until the switchover is complete. 0: Switch to new clock source not yet complete. 1: SYS_OSC is clock source selected in GCR_CLKCTRL.sysclk_sel . | |
| 12 | - | RO | 0 | Reserved Do not modify | |
| 11:9 | sysclk_sel | R/W | 0 | System Clock Source Select Selects the system oscillator (SYS_OSC) used as the system clock (SYS_CLK) source. Modifying this field immediately clears GCR_CLKCTRL.sysclk_rdy . 0: ISO (POR and System Reset default) 1: Reserved 2: Reserved 3: INRO 4: IPO 5: IBRO 6: ERTCO 7: External Clock, EXT_CLK, P0.3, AF1 | |
| 8:6 | sysclk_div | R/W | 0 | System Clock Prescaler Sets the divider for generating SYS_CLK from the selected SYS_OSC as shown in the following equation: $\text{SYS_CLK} = \frac{\text{SYS_OSC}}{2^{\text{sysclk_div}}}$ Note: Valid values are from 0 to 7 for sysclk_div. | |
| 5:0 | - | RO | 8 | Reserved Do not modify | |

Table 3-63: Power Management Register

| Power Management | | | GCR_PM | | 0x000C |
|------------------|-------|--------|--------|----------------------------------|--------|
| Bits | Field | Access | Reset | Description | |
| 31:18 | - | RO | 0 | Reserved Do not modify | |

| Power Management | | | | GCR_PM | 0x000C |
|------------------|------------|--------|---------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 17 | ibro_pd | R/W | 1 | IBRO Power Down LPM Set this field to 1 to power down the IBRO when entering LPM . 0: IBRO is powered on during LPM 1: IBRO is powered off during LPM | |
| 16 | ipo_pd | R/W | 1 | IPO Power Down LPM Set this field to 1 to power down the IPO when entering LPM . 0: IPO is powered on during LPM 1: IPO is powered off during LPM | |
| 15 | iso_pd | R/W | 1 | ISO Power Down LPM Set this field to 1 to power down the ISO when entering LPM . 0: ISO is powered on during LPM 1: ISO is powered off during LPM | |
| 14:10 | - | DNM | 0b11100 | Reserved Do not modify | |
| 9 | aincomp_we | R/W | 0 | Analog Input Comparator Wakeup Enable This bit enables the Analog Input Comparator IRQ to wake the device from SLEEP , LPM , or BACKUP modes. | |
| 8 | - | RO | 0 | Reserved Do not modify | |
| 7 | wut_we | R/W | 0 | Wakeup Timer Enable Set this field to 1 to enable the Wakeup Timer as a wakeup source. The Wakeup Timer will wake the device from SLEEP , LPM or BACKUP modes. 0: Wakeup source disabled 1: Wakeup source enabled. | |
| 6 | - | RO | 0 | Reserved Do not modify | |
| 5 | rtc_we | R/W | 0 | RTC Alarm Wakeup Enable Set this field to 1 to enable an RTC alarm to wake the device. The RTC alarm will wake the device from SLEEP , LPM or BACKUP modes. 0: Wakeup source disabled 1: Wakeup source enabled | |
| 4 | gpio_we | R/W | 0 | GPIO Wakeup Enable Set this field to 1 to enable all GPIO pins as potential wakeup sources. Any GPIO configured for wakeup will wake the device from SLEEP , LPM or BACKUP modes. 0: Wakeup source disabled 1: Wakeup source enabled | |
| 3:0 | mode | R/W | 0 | Operating Mode This field controls the operating mode of the device. 0b0000: ACTIVE 0b0001: SLEEP 0b0010: STANDBY 0b0100: BACKUP 0b1000: LPM (CM4 deep sleep) 0b1001: UPM 0b1010: PDM | |

Table 3-64: Peripheral Clock Divisor Register

| Peripheral Clocks Divisor | | | GCR_PCLKDIV | | [0x0018] |
|---------------------------|-----------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:18 | - | RO | - | Reserved Do not modify | |
| 17 | cnnclksel | R/W | 0 | CNN Peripheral Clock Select Set this field to select the clock source for the CNN peripheral clock, f_{CNN_Clock} . 0: PCLK 1: ISO | |
| 16:14 | cnnclkdiv | R/W | | CNN Peripheral Clock Frequency Divider This field is used as a divider of the CNN Peripheral Clock. The CNN Peripheral Clock, f_{CNN_Clock} , is selected using the field GCR_PCLKDIV.cnnclksel . 0: $\frac{CNN_Clock}{2}$ 1: $\frac{CNN_Clock}{4}$ 2: $\frac{CNN_Clock}{8}$ 3: $\frac{CNN_Clock}{16}$ 4-7: $\frac{CNN_Clock}{1}$ | |
| 13:10 | adcfreq | R/W | 0 | ADC Peripheral Clock Frequency Select Configures the frequency of the ADC peripheral clock from the PCLK. 0: Reserved 1: Reserved 2 – 15: $f_{adc_clk} = f_{PCLK}/adcfreq$ | |
| 9:0 | - | RO | - | Reserved Do not modify | |

Table 3-65: Peripheral Clock Disable Register 0

| Peripheral Clocks Disable 0 | | | GCR_PCLKDIS0 | | [0x0024] |
|-----------------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:30 | - | R/W | 1 | Reserved Do not modify | |
| 29 | pt | R/W | 1 | Pulse Train Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled. 1: Clock disabled | |
| 28 | i2c1 | R/W | 1 | I2C1 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 27:26 | - | RO | 1 | Reserved Do not modify | |

| Peripheral Clocks Disable 0 | | | GCR_PCLKDIS0 | | [0x0024] |
|-----------------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 25 | cnn | R/W | 1 | CNN Clock Disable Disabling a clock disables functionality while also saving power. Read and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 24 | - | RO | 1 | Reserved Do not modify | |
| 23 | adc | R/W | 1 | ADC Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 22:19 | - | RO | 1 | Reserved Do not modify | |
| 18 | tmr3 | R/W | 1 | TMR3 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 17 | tmr2 | R/W | 1 | TMR2 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 16 | tmr1 | R/W | 1 | TMR1 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 15 | tmr0 | R/W | 1 | TMRO Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 14 | - | RO | 1 | Reserved Do not modify | |
| 13 | i2c0 | R/W | 1 | I2C0 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 12:11 | - | RO | 1 | Reserved Do not modify | |

| Peripheral Clocks Disable 0 | | | GCR_PCLKDIS0 | | [0x0024] |
|-----------------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 10 | uart1 | R/W | 1 | UART1 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 9 | uart0 | R/W | 1 | UART0 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 8:7 | - | RO | 0b11 | Reserved Do not modify | |
| 6 | spi1 | R/W | 1 | SPI1 Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 5 | dma | R/W | 1 | DMA Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 4:2 | - | RO | 0b11 | Reserved Do not modify | |
| 1 | gpio1 | R/W | 1 | GPIO1 Port and Pad Logic Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |
| 0 | gpio0 | R/W | 1 | GPIO0 Port and Pad Logic Clock Disable Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled | |

Table 3-66: Memory Clock Control Register

| Memory Clock Control | | | GCR_MEMCTRL | | [0x0028] |
|----------------------|------------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | RO | 0 | Reserved Do not modify | |
| 16 | sysram0ecc | R/W | 0 | sysram0 ECC Enable Set this field to 1 to enable ECC for sysram0 . 0: sysram0 Active, ECC disabled. 1: sysram0 Active, ECC enabled. | |

| Memory Clock Control | | | GCR_MEMCTRL | | [0x0028] |
|----------------------|-------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 15:3 | - | RO | 0 | Reserved Do not modify | |
| 2:0 | fws | R/W | 5 | Program Flash Wait States Number of wait-state cycles per Flash code read access. 0: Invalid 1 – 7: Number of Flash code access wait states <i>Note: For the IPO and ISO clocks the minimum wait state is 2.</i> <i>Note: For all other clock sources the minimum wait state is 1.</i> | |

Table 3-67: Memory Zeroize Control Register

| Memory Zeroize | | | GCR_MEMZ | | [0x002C] |
|----------------|------------|--------|----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:7 | - | RO | - | Reserved Do not modify | |
| 6 | icc1 | R/W1O | 0 | ICC1 Zeroization Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress. | |
| 5 | icc0 | R/W1O | 0 | ICC0 Zeroization Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress. | |
| 4 | sysram0ecc | R/W1O | 0 | sysram0 ECC Zeroization Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress. | |
| 3 | ram3 | R/W1O | 0 | sysram3 Zeroization Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress. | |
| 2 | ram2 | R/W1O | 0 | sysram2 Zeroization Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress. | |
| 1 | ram1 | R/W1O | 0 | sysram1 Zeroization Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress. | |

| Memory Zeroize | | | GCR_MEMZ | | [0x002C] |
|----------------|-------|--------|----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 0 | ram0 | R/W1O | 0 | sysram0 Zeroization Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress. | |

Table 3-68: System Status Flag Register

| System Status Flag | | | GCR_SYSST | | [0x0040] |
|--------------------|---------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | RO | 0 | Reserved Do not modify | |
| 0 | icelock | R | 0 | Arm ICE Lock Status Flag 0: Arm ICE is unlocked (enabled) 1: Arm ICE is locked (disabled) | |

Table 3-69: Reset Register 1

| Reset 1 | | | GCR_RST1 | | [0x0044] |
|---------|-------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31 | cpu1 | RO | 0 | CPU1 (RV32) Reset Write 1 to initiate the reset operation. 0: Operation complete 1: Operation in progress | |
| 30:26 | - | RO | 0 | Reserved Do not modify | |
| 25 | simo | R/W | 0 | Single Inductor Multiple Output Block Reset Write 1 to initiate the reset operation. 0: Operation complete 1: Operation in progress | |
| 24 | dvs | R/W | 0 | Dynamic Voltage Scaling Controller Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |
| 23:21 | - | RO | 0 | Reserved Do not modify | |
| 20 | i2c2 | R/W | 0 | I²C2 Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |
| 19 | i2s | R/W | 0 | Audio Interface Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |

| Reset 1 | | | GCR_RST1 | | [0x0044] |
|---------|-------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 18:17 | - | R/W | 0 | Reserved Do not modify | |
| 16 | smphr | R/W | 0 | Semaphore Block Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |
| 15:12 | - | R/W | - | Reserved Do not modify | |
| 11 | spi0 | R/W | 0 | SPI0 Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |
| 10 | aes | R/W | 0 | AES Block Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |
| 9 | crc | R/W | 0 | CRC Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |
| 8 | - | R/W | 0 | Reserved Do not modify | |
| 7 | owm | R/W | 0 | 1-Wire Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |
| 6:2 | - | RO | 0 | Reserved Do not modify | |
| 1 | pt | R/W | 0 | Pulse Train Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |
| 0 | i2c1 | R/W | 0 | I²C1 Reset Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress. | |

Table 3-70: Peripheral Clock Disable Register 1

| Peripheral Clock Disable 1 | | | GCR_PCLKDIS1 | | [0x0048] |
|----------------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31 | cpu1 | R/W | 1 | CPU1 (RV32 Clock Disable) Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled 1: Disabled | |
| 30:28 | - | R/W | 1 | Reserved Do not modify | |
| 27 | wdt0 | R/W | 1 | Watchdog Timer 0 Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled 1: Disabled | |
| 26:25 | - | R/W | 1 | Reserved Do not modify | |
| 24 | i2c2 | R/W | 1 | I²C2 Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled 1: Disabled | |
| 23 | i2s0 | R/W | 1 | I²S Audio Interface Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled 1: Disabled | |
| 22:17 | - | R/W | 1 | Reserved Do not modify. | |
| 16 | spi0 | R/W | 1 | SPI0 Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled. | |
| 15 | aes | R/W | 1 | AES Block Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled. | |
| 14 | crc | R/W | 1 | CRC Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled. | |

| Peripheral Clock Disable 1 | | | | GCR_PCLKDIS1 | [0x0048] |
|----------------------------|--------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 13 | owm | R/W | 1 | 1-Wire Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled. | |
| 12:10 | - | R/W1 | 1 | Reserved Do not modify | |
| 9 | smpthr | R/W | 1 | Semaphore Block Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled. | |
| 8:3 | - | R/W1 | 1 | Reserved Do not modify | |
| 2 | trng | R/W | 1 | TRNG Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled. | |
| 1 | uart2 | R/W | 1 | UART2 Clock Disable Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled. | |
| 0 | - | R/W1 | 1 | Reserved Do not modify | |

Table 3-71: Event Enable Register

| Event Enable | | | | GCR_EVENTEN | [0x004C] |
|--------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:3 | - | RO | 0 | Reserved Do not modify | |
| 2 | tx | R/W | 0 | CPU0 (CM4) TXEV Event Enable When this bit is set, the TXEV event will wake the CM4 from a low power mode entered with a WFE instruction. 0: Disabled. 1: Enabled. | |
| 1 | - | RO | 0 | Reserved Do not modify | |

| Event Enable | | | | GCR_EVENTEN | [0x004C] |
|--------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 0 | dma | R/W | 0 | CPU0 (CM4) DMA CTZ Wake-Up Enable Enables a DMA CTZ event to generate an RXEV interrupt to wake the CM4 from a low power mode entered with a WFE instruction. 0: Disabled. 1: Enabled. | |

Table 3-72: Revision Register

| Revision | | | | GCR_REVISION | [0x0050] |
|----------|----------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved Do not modify | |
| 15:0 | revision | R | * | Device Revision Returns the chip revision ID as packed BCD. For example, 0xA1 would indicate the device is revision A1. | |

Table 3-73: System Status Interrupt Enable Register

| System Status Interrupt Enable | | | | GCR_SYSIE | [0x0054] |
|--------------------------------|-----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | RO | - | Reserved Do not modify | |
| 0 | iceunlock | R/W | 0 | Arm ICE Unlocked Interrupt Enable Set this field to generate an interrupt if the GCR_SYSST.iceclock is set. 0: Interrupt disabled 1: Interrupt enabled | |

Table 3-74: Error Correction Coding Error Register

| Error Correction Coding Error | | | | GCR_ECCERR | [0x0064] |
|-------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | RO | 0 | Reserved Do not modify | |
| 0 | ram0 | R/W1C | 0 | sysram0 ECC Error This flag is set if an ECC error occurs in sysram0 . Write to 1 to clear the flag. 0: No error 1: Error | |

Table 3-75: Error Correction Coding Correctable Error Detected Register

| Error Correction Coding Correctable Error Detected | | | | GCR_ECCCED | [0x0068] |
|--|-------|--------|-------|----------------------------------|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | RO | 0 | Reserved Do not modify | |

| Error Correction Coding Correctable Error Detected | | | | GCR_ECCCED | [0x0068] |
|--|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 0 | ram0 | R/W1C | 0 | sysram0 Correctable ECC Error Detected When this bit is set it indicates that there is a single correctable error in the sysram0 block. Write to 1 to clear the flag. 0: No error or uncorrectable error if GCR_ECCERR.ram0 is set to 1. 1: Correctable error detected. | |

Table 3-76: Error Correction Coding Interrupt Enable Register

| Error Correction Coding Interrupt Enable | | | | GCR_ECCIE | [0x006C] |
|--|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | RO | 0 | Reserved Do not modify | |
| 0 | ram0 | R/W | 0 | sysram0 ECC Error Interrupt Enable Set this field to 1 to generate an interrupt if an ECC error condition occurs for sysram0 . 0: Interrupt disabled 1: Interrupt enabled | |

Table 3-77: Error Correction Coding Error Address Register

| Error Correction Coding Error Address | | | | GCR_ECCADDR | [0x0070] |
|---------------------------------------|------------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31 | tagramerr | R | 0 | ECC Error Address/TAG RAM Error Data depends on which block has reported the error. If sysram0 then this bit represents the bit of the AMBA address of read which produced the error. If the error is the cache, then this bit is set as shown below: 0: No error 1: Tag Error. The error is in the TAG RAM | |
| 30 | tagrambank | R | 0 | ECC Error Address/TAG RAM Error Bank Data depends on which block has reported the error. If sysram0 then this bit represents the bit of the AMBA address of read which produced the error. If the error is from the cache, then this bit is set as shown below: 0: Error is in TAG RAM Bank 0 1: Error is in TAG RAM Bank 1 | |
| 29:16 | tagramaddr | R | 0 | ECC Error Address/TAG RAM Error Address Data depends on which block has reported the error. If sysram0 these bits represents the bits of the AMBA address of read which produced the error. If the error is from the cache, then this field is set as shown below: [TAG ADDRESS]: Represents the TAG RAM Address | |
| 15 | dataramerr | R | 0 | ECC Error Address/DATA RAM Error Address Data depends on which block has reported the error. If sysram0 then this bit represents the bit of the AMBA address of read which produced the error. If the error is from the cache, then this bit is set as shown below: 0: No error 1: DATA RAM Error. The error is in the Data RAM | |

| Error Correction Coding Error Address | | | | GCR_ECCADDR | [0x0070] |
|---------------------------------------|-------------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 14 | datarambank | R | 0 | ECC Error Address/DATA RAM Error Bank Data depends on which block has reported the error. If sysram0 then this bit represents the bits of the AMBA address of read which produced the error. If the error is from the cache, then this bit is set as shown below: 0: Error is in sysram0 1: Error is in DATA RAM Bank 1 | |
| 13:0 | dataramaddr | R | 0 | ECC Error Address/TAG RAM Error Address Data depends on which block has reported the error. This field represents the bits of the AMBA address of read which produced the error. If the error is from the caches, then this field is set as shown below: [DATA ADDRESS]: Represents the DATA RAM Error Address | |

Table 3-78: General Purpose Register 0

| General Purpose Register 0 | | | | GCR_GPRO | [0x0080] |
|----------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W | 0 | General Purpose Register General purpose register | |

3.14 Error Correction Coding (ECC)

See [Table 2-4: APP Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-79: Error Correction Coding Enable Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 3-79: Error Correction Coding Enable Register Summary

| Offset | Register Name | Access | Description |
|----------|---------------|--------|--------------------------------|
| [0x0000] | ECC_EN | R/W | Error Correction Coding Enable |

3.14.1 Error Correction Coding Enable Register Details

Table 3-80: Error Correction Coding Enable Register

| Error Correction Coding Enable | | | ECC_EN | | [0x0000] |
|--------------------------------|------|--------|--------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:9 | - | RO | 0 | Reserved Do not modify | |
| 8 | ram0 | R/W | 0 | System RAM ECC Enable Set this field to 1 to enable ECC for sysram0 . 0: Disabled 1: Enabled | |

| Error Correction Coding Enable | | | ECC_EN | [0x0000] |
|--------------------------------|------|--------|--------|----------------------------------|
| Bits | Name | Access | Reset | Description |
| 7:0 | - | RO | 0 | Reserved Do not modify |

3.15 System Initialization Registers (SIR)

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-81: System Initialization Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 3-81: System Initialization Register Summary

| Offset | Register Name | Access | Description |
|----------|----------------------------|--------|--|
| [0x0000] | SIR_STAT | RO | <i>System Initialization Status Register</i> |
| [0x0004] | SIR_ADDR | RO | <i>System Initialization Address Error Register</i> |
| [0x0100] | SIR_FSTAT | RO | <i>System initialization Function Status Register</i> |
| [0x0104] | SIR_SFSTAT | RO | <i>System initialization Security Function Status Register</i> |

3.15.1 System Initialization Register Details

Table 3-82: System Initialization Status Register

| System Initialization Status | | | SIR_STAT | [0x0000] |
|------------------------------|--------|--------|-----------------|---|
| Bits | Name | Access | Reset | Description |
| 31:2 | - | RO | 0 | Reserved for Future Use Do not modify |
| 1 | crcerr | RO | See Description | CRC Configuration Error Flag This field is set by hardware during reset if an error in the device configuration is detected in the OTP memory. 0: Configuration valid. 1: Configuration invalid, the address of the configuration error is stored in the SIR_ADDR register. <i>Note: If this field reads 1 a device error has occurred. Please contact Maxim Integrated technical support for additional assistance providing the address contained in SIR_ADDR.erraddr.</i> |
| 0 | magic | RO | See Description | Configuration Valid Flag This field is set to 1 by hardware during reset if the device configuration is valid and the magic word is set by the factory. 0: OTP is not configured correctly 1: OTP Configuration Valid <i>Note: If this field reads 0 the device configuration is invalid, and a device error has occurred during system initialization. Please contact Maxim Integrated technical support for additional assistance.</i> |

Table 3-83: System Initialization Address Error Register

| System Initialization Status | | | SIR_ADDR | | [0x0004] |
|------------------------------|---------|--------|----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | erraddr | RO | 0 | Configuration Error Address If the SIR_STAT.crcerr field is set to 1, the value in this register is the address of the configuration failure. | |

Table 3-84: System Initialization Function Status Register

| System Initialization Function Status | | | SIR_FSTAT | | [0x0100] |
|---------------------------------------|-------|--------|-----------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved Do not modify | |
| 7 | smphr | RO | See Description | Semaphore Block This field indicates if the device includes the Semaphore Block. 0: Block is not available. 1: Block is available. | |
| 6:3 | -- | RO | 0 | Reserved Do not modify | |
| 2 | adc | RO | See Description | ADC This field indicates if the device includes the ADC. 0: Block is not available. 1: Block is available. | |
| 1 | - | RO | 0 | Reserved Do not modify | |
| 0 | fpu | RO | See Description | FPU This field indicates if the device includes the FPU. 0: Block is not available. 1: Block is available. | |

Table 3-85: System Initialization Security Function Status Register

| System Initialization Security Function Status | | | SIR_SFSTAT | | [0x0104] |
|--|------|--------|-----------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved Do not modify | |
| 3 | aes | RO | See Description | AES This field indicates if the device includes the AES block. 0: Block is not available. 1: Block is available. | |
| 2 | trng | RO | See Description | TRNG This field indicates if the device includes the TRNG block. 0: Block is not available. 1: Block is available. | |
| 1:0 | - | RO | 0 | Reserved Do not modify | |

3.16 Function Control Registers (FCR)

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-86: Function Control Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 3-86: Function Control Register Summary

| Offset | Register | Description |
|----------|---------------------------------|---|
| [0x0000] | FCR_FCTRL0 | <i>Function Control 0 Register (I²C Glitch Filter Control)</i> |
| [0x0004] | FCR_AUTOCAL0 | <i>IPO Automatic Calibration 0 Register</i> |
| [0x0008] | FCR_AUTOCAL1 | <i>IPO Automatic Calibration 1 Register</i> |
| [0x000C] | FCR_AUTOCAL2 | <i>IPO Automatic Calibration 2 Register</i> |
| [0x0010] | FCR_URVBOOTADDR | <i>RV32 Boot Address Register</i> |
| [0x0014] | FCR_URVCTRL | <i>RV32 Control Register</i> |

3.16.1 Function Control Register Details

Table 3-87: Function Control 0 Register

| Function Control 0 | | | FCR_FCTRL0 | | [0x0000] |
|--------------------|--------------------|--------|------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:26 | - | RO | 0 | Reserved Do not modify | |
| 25 | i2c2_scl_filter_en | R/W | 0 | I²C2 SCL Glitch Filter Enable 0: Disabled 1: Enabled | |
| 24 | i2c2_sda_filter_en | R/W | 0 | I²C2 SDA Glitch Filter Enable 0: Disabled 1: Enabled | |
| 23 | i2c1_scl_filter_en | R/W | 0 | I²C1 SCL Glitch Filter Enable 0: Disabled 1: Enabled | |
| 22 | i2c1_sda_filter_en | R/W | 0 | I²C1 SDA Glitch Filter Enable 0: Disabled 1: Enabled | |
| 21 | i2c0_scl_filter_en | R/W | 0 | I²C0 SCL Glitch Filter Enable 0: Disabled 1: Enabled | |
| 20 | i2c0_sda_filter_en | R/W | 0 | I²C0 SDA Glitch Filter Enable 0: Disabled 1: Enabled | |
| 19:0 | - | RO | 0 | Reserved Do not modify | |

Table 3-88: IPO Automatic Calibration 0 Register

| IPO Automatic Calibration 0 | | | FCR_AUTOCAL0 | | [0x0004] |
|-----------------------------|--------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:23 | trim | RO | 0 | IPO Trim Value Initial factory trim value for the IPO. | |
| 22:20 | - | RO | | Reserved Do not modify | |
| 19:8 | gain | R/W | 0 | IPO Trim Adaptation Gain | |
| 7:5 | - | RO | 0 | Reserved Do not modify | |
| 4 | atomic | R/W1 | 0 | IPO Trim Atomic Start Set this bit to start an atomic automatic calibration of the IPO. The calibration will run for FCR_.runtime milliseconds. This bit is automatically cleared by hardware when the calibration is complete. | |
| 3 | invert | R/W | 0 | IPO Trim Step Invert 0: IPO trim step is not inverted 1: IPO trim step is inverted | |
| 2 | load | R/* | 0 | IPO Initial Trim Load Set this bit to load the initial trim value for the IPO from FCR_.initial . This bit will be cleared by hardware once the load is complete. | |
| 1 | en | R/W | 0 | IPO Automatic Calibration Continuous Mode Enable 0: Disabled 1: Enabled | |
| 0 | sel | R/W | 0 | IPO Trim Select 0: Use default trim 1: Use automatic calibration trim values | |

Table 3-89: IPO Automatic Calibration 1 Register

| IPO Automatic Calibration 1 | | | FCR_AUTOCAL1 | | [0x0008] |
|-----------------------------|---------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:9 | - | RO | 0 | Reserved Do not modify | |
| 8:0 | initial | R/W | 0 | IPO Trim Automatic Calibration Initial Trim This field contains the initial trim setting for the IPO. | |

Table 3-90: IPO Automatic Calibration 2 Register

| IPO Automatic Calibration 2 | | | FCR_AUTOCAL2 | | [0x000C] |
|-----------------------------|-------|--------|--------------|----------------------------------|----------|
| Bits | Field | Access | Reset | Description | |
| 31:21 | - | RO | 0 | Reserved Do not modify | |

| IPO Automatic Calibration 2 | | | FCR_AUTOCAL2 | | [0x000C] |
|-----------------------------|---------|--------|--------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 20:8 | div | R/W | 0 | IPO Trim Automatic Calibration Divide Factor Target trim frequency for the IPO: $f_{IPO} = \text{div} \times 32768$ <i>Note: Setting div to 0 is equivalent to setting div to 1.</i> | |
| 7:0 | runtime | R/W | 0 | IPO Trim Automatic Calibration Run Time Atomic Run Time = donecnt milliseconds | |

Table 3-91: RV32 Boot Address Register

| RV32 Boot Address | | | FCR_URVBOOTADDR | | [0x0010] |
|-------------------|-------|--------|-----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W | 0x2000_C000 | RV32 Boot Address Set this field to the boot address for the RV32 core. The reset value for this register is 0x2001_C000, sysram3 . | |

Table 3-92: RV32 Control Register

| RV32 Boot Address | | | FCR_URVCTRL | | [0x0014] |
|-------------------|----------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:2 | - | RO | 0 | Reserved Do not modify | |
| 1 | iflushen | R/W | 0 | ICC1 Cache Flush Enable Write 1 to flush the cache and the instruction buffer for the RV32 core. This bit is automatically cleared to 0 when the flush is complete. Writing 0 has no effect and does not stop a cache flush in progress. 0: ICC1 flush complete 1: Flush the contents of the ICC1 cache | |
| 0 | memsel | R/W | 0 | RV32 Memory Select This field determines if sysram2 and sysram3 are shared between the CM4 and RV32 cores. Set this field to 1 to set the RV32 core as the exclusive master for sysram2 and sysram3 . 0: sysram2 and sysram3 are shared and accessible by both the CM4 and RV32 cores. 1: sysram2 and sysram3 are accessible by the RV32 core only. <i>Note: The application firmware must ensure that no accesses are occurring in sysram2 or sysram3 prior to setting this field to 1. Refer to section 8.2 Multiprocessor Communications for information on using the Semaphore peripheral for communication between the RV32 and CM4 cores.</i> | |

3.17 General Control Function Registers (GCFR)

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 3-93: General Control Function Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 3-93: General Control Function Register Summary

| Offset | Register | Description |
|----------|---------------------------|-------------------------------------|
| [0x0000] | GCFR_REG0 | General Control Function Register 0 |
| [0x0004] | GCFR_REG1 | General Control Function Register 1 |
| [0x0008] | GCFR_REG2 | General Control Function Register 2 |
| [0x000C] | GCFR_REG3 | General Control Function Register 3 |

3.17.1 General Control Function Register Details

Table 3-94: General Control Function Register 0

| General Control Function 0 | | GCFR_REG0 | | [0x0000] |
|----------------------------|-----------------|-----------|-------|--|
| Bits | Field | Access | Reset | Description |
| 31:4 | - | RO | 0 | Reserved Do not modify |
| 3 | cnnx16_3_pwr_en | R/W | 0 | CNNx16_3 Power Domain Enable 0: Disabled 1: Enabled |
| 2 | cnnx16_2_pwr_en | R/W | 0 | CNNx16_2 Power Domain Enable 0: Disabled 1: Enabled |
| 1 | cnnx16_1_pwr_en | R/W | 0 | CNNx16_1 Power Domain Enable 0: Disabled 1: Enabled |
| 0 | cnnx16_0_pwr_en | R/W | 0 | CNNx16_0 Power Domain Enable 0: Disabled 1: Enabled |

Table 3-95: General Control Function Register 1

| General Control Function Register 1 | | GCFR_REG1 | | [0x0004] |
|-------------------------------------|-----------------|-----------|-------|---|
| Bits | Field | Access | Reset | Description |
| 31:4 | - | RO | 0 | Reserved Do not modify |
| 3 | cnnx16_3_ram_en | R/W | 0 | CNNx16_3 RAM Power Enable 0: Disabled 1: Enabled |
| 2 | cnnx16_2_ram_en | R/W | 0 | CNNx16_2 RAM Power Enable 0: Disabled 1: Enabled |
| 1 | cnnx16_1_ram_en | R/W | 0 | CNNx16_1 RAM Power Enable 0: Disabled 1: Enabled |
| 0 | cnnx16_0_ram_en | R/W | 0 | CNNx16_0 RAM Power Enable 0: Disabled 1: Enabled |

Table 3-96: General Control Function Register 2

| General Control Function Register 2 | | | GCFR_REG2 | | [0x0008] |
|-------------------------------------|--------------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved Do not modify | |
| 3 | cnnx16_3_iso | R/W | 0 | CNNx16_3 Power Domain Isolation 0: Disabled 1: Enabled | |
| 2 | cnnx16_2_iso | R/W | 0 | CNNx16_2 Power Domain Isolation 0: Disabled 1: Enabled | |
| 1 | cnnx16_1_iso | R/W | 0 | CNNx16_1 Power Domain Isolation 0: Disabled 1: Enabled | |
| 0 | cnnx16_0_iso | R/W | 0 | CNNx16_0 Power Domain Isolation 0: Disabled 1: Enabled | |

Table 3-97: General Control Function Register 3

| General Control Function Register 3 | | | GCFR_REG3 | | [0x000C] |
|-------------------------------------|--------------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved Do not modify | |
| 3 | cnnx16_3_RST | R/W | 0 | CNNx16+3 Power Domain Reset 0: Disabled 1: Enabled | |
| 2 | cnnx16_2_RST | R/W | 0 | CNNx16_2 Power Domain Reset 0: Disabled 1: Enabled | |
| 1 | cnnx16_1_RST | R/W | 0 | CNNx16_1 Power Domain Reset 0: Disabled 1: Enabled | |
| 0 | cnnx16_0_RST | R/W | 0 | CNNx16_0 Power Domain Reset 0: Disabled 1: Enabled | |

4. Interrupts and Exceptions

Interrupts and exceptions are managed by the Arm Cortex-M4 with FPU Nested Vector Interrupt Controller (NVIC) or the RV32 interrupt controller. The NVIC manages the interrupts, exceptions, priorities, and masking. [Table 4-1: MAX78000 CM4 Interrupt Vector Table](#) and [Table 4-2: MAX78000 RV32 Interrupt Vector Table](#) details the MAX78000's interrupt vector tables for the CM4 and RV32 processors respectively and describes each exception and interrupt.

4.1 CM4 Interrupt and Exception Features

- 8 programmable priority levels
- Nested exception and interrupt support
- Interrupt masking

4.2 CM4 Interrupt Vector Table

[Table 4-1](#) lists the interrupt and exception table for the MAX78000's CM4 core. There are 119 interrupt entries for the MAX78000, including reserved for future use interrupt place holders. Including the 15 system exceptions for the Arm Cortex-M4 with FPU, the total number of entries is 134.

Table 4-1: MAX78000 CM4 Interrupt Vector Table

| Exception (Interrupt) Number | Offset | Name | Description |
|------------------------------|-------------------|--------------------|--|
| 1 | [0x0004] | Reset_Handler | Reset |
| 2 | [0x0008] | NMI_Handler | Non-Maskable Interrupt |
| 3 | [0x000C] | HardFault_Handler | Hard Fault |
| 4 | [0x0010] | MemManage_Handler | Memory Management Fault |
| 5 | [0x0014] | BusFault_Handler | Bus Fault |
| 6 | [0x0018] | UsageFault_Handler | Usage Fault |
| 7:10 | [0x001C]-[0x0028] | - | Reserved |
| 11 | [0x002C] | SVC_Handler | Supervisor Call Exception |
| 12 | [0x0030] | DebugMon_Handler | Debug Monitor Exception |
| 13 | [0x0034] | - | Reserved |
| 14 | [0x0038] | PendSV_Handler | Request Pending for System Service |
| 15 | [0x003C] | SysTick_Handler | System Tick Timer |
| 16 | [0x0040] | PF_IRQHandler | Power Fail interrupt |
| 17 | [0x0044] | WDTO_IRQHandler | Windowed Watchdog Timer 0 Interrupt |
| 18 | [0x0048] | - | Reserved |
| 19 | [0x004C] | RTC_IRQHandler | Reserved |
| 20 | [0x0050] | TRNG_IRQHandler | True Random Number Generator Interrupt |
| 21 | [0x0054] | TMR0_IRQHandler | Timer 0 Interrupt |
| 22 | [0x0058] | TMR1_IRQHandler | Timer 1 Interrupt |
| 23 | [0x005C] | TMR2_IRQHandler | Timer 2 Interrupt |
| 24 | [0x0060] | TMR3_IRQHandler | Timer 3 Interrupt |
| 25 | [0x0064] | TMR4_IRQHandler | Timer 4 (LPTMR0) Interrupt |
| 26 | [0x0068] | TMR5_IRQHandler | Timer 5 (LPTMR1) Interrupt |
| 27:28 | [0x006C]:[0x0070] | - | Reserved |
| 29 | [0x0074] | I2C0_IRQHandler | I ² C Port 0 Interrupt |
| 30 | [0x0078] | UART0_IRQHandler | UART Port 0 Interrupt |

| Exception (Interrupt) Number | Offset | Name | Description |
|---------------------------------|-------------------|---------------------|---|
| 31 | [0x007C] | UART1_IRQHandler | UART Port 1 Interrupt |
| 32 | [0x0080] | SPI1_IRQHandler | SPI Port 1 Interrupt |
| 33:35 | [0x0084]:[0x008C] | - | Reserved |
| 36 | [0x90] | ADC_IRQHandler | ADC Interrupt |
| 37:38 | [0x0094]:[0x0098] | - | Reserved |
| 39 | [0x009C] | FLCO_IRQHandler | Flash Controller 0 Interrupt |
| 40 | [0x00A0] | GPIO0_IRQHandler | GPIO Port 0 Interrupt |
| 41 | [0x00A4] | GPIO1_IRQHandler | GPIO Port 1 Interrupt |
| 42 | [0x00A8] | GPIO2_IRQHandler | GPIO Port 2 Interrupt |
| 43 | [0x00AC] | - | Reserved |
| 44 | [0x00B0] | DMA0_IRQHandler | DMA0 Interrupt |
| 45 | [0x00B4] | DMA1_IRQHandler | DMA1 Interrupt |
| 46 | [0x00B8] | DMA2_IRQHandler | DMA2 Interrupt |
| 47 | [0x00BC] | DMA3_IRQHandler | DMA3 Interrupt |
| 48:49 | [0x00C0 : 0x00C4] | - | Reserved |
| 50 | [0x00C8] | UART2_IRQHandler | UART Port 2 Interrupt |
| 51 | [0x00CC] | - | Reserved |
| 52 | [0x00D0] | I2C1_IRQHandler | I ² C Port 1 Interrupt |
| 53:68 | [0x00D4]:[0x0110] | - | Reserved |
| 69 | [0x0114] | WUT_IRQHandler | Wakeup Timer Interrupt |
| 70 | [0x0118] | GPIOWake_IRQHandler | GPIO Wakeup Interrupt |
| 71 | [0x011C] | - | Reserved |
| 72 | [0x0120] | SPI0_IRQHandler | SPI Port 0 Interrupt |
| 73 | [0x0124] | WDT1_IRQHandler | Low Power Watchdog Timer 0 (WDT1) Interrupt |
| 74 | [0x0128] | - | Reserved |
| 75 | [0x012C] | PT_IRQHandler | Pulse Train Interrupt |
| 76:77 | [0x0130]:[0x0134] | - | Reserved |
| 78 | [0x0138] | I2C2_IRQHandler | I ² C Port 2 Interrupt |
| 79 | [0x013C] | RISCV_IRQHandler | CPU1 (RV32) Interrupt |
| 80:82 | [0x0140]:[0x0148] | - | Reserved |
| 83 | [0x014C] | OWM_IRQHandler | 1-Wire Master Interrupt |
| 84:97 | [0x0150]:[0x0184] | - | Reserved |
| 98 | [0x0188] | ECC_IRQHandler | Error Correction Coding Block Interrupt |
| 99 | [0x018C] | DVS_IRQHandler | Digital Voltage Scaling Interrupt |
| 100 | [0x0190] | SIMO_IRQHandler | Single Input Multiple Output Interrupt |
| 101:103 | [0x0194]:[0x019C] | - | Reserved |
| 104 | [0x01A0] | UART3_IRQHandler | UART3 (LPUART0) Interrupt |
| 105:106 | [0x01A4]:[0x01A8] | - | Reserved |
| 107 | [0x01AC] | PCIF_IRQHandler | Parallel Camera Interface Interrupt |
| 108:112 | [0x01B0]:[0x01C0] | - | Reserved |
| 113 | [0x01C4] | AES_IRQHandler | AES Interrupt |
| 114 | [0x01C8] | - | Reserved |
| 115 | [0x01CC] | I2S_IRQHandler | I2S Interrupt |

| Exception (Interrupt) Number | Offset | Name | Description |
|---------------------------------|----------|---------------------|--------------------------------|
| 116 | [0x01D0] | CNN_FIFO_IRQHandler | CNN Transmit FIFO Interrupt |
| 117 | [0x01D4] | CNN_IRQHandler | CNN Interrupt |
| 118 | [0x01D8] | - | Reserved |
| 119 | [0x01DC] | LPCOMP_IRQHandler | Low Power Comparator Interrupt |

4.3 RV32 Interrupt Vector Table

Table 4-2 lists the interrupt and exception table for the MAX78000's RV32 core.

Table 4-2: MAX78000 RV32 Interrupt Vector Table

| Exception (Interrupt) Number | Name | Description |
|---------------------------------|--------------------|-------------------------------------|
| 4 | PF_IRQHandler | System Fault interrupt |
| 5 | WDT0_IRQHandler | Windowed Watchdog Timer 0 Interrupt |
| 6 | GPIOAKE_IRQHandler | GPIO Wakeup Interrupt |
| 7 | RTC_IRQHandler | RTC Interrupt |
| 8 | TMRO_IRQHandler | Timer 0 Interrupt |
| 9 | TMR1_IRQHandler | Timer 1 Interrupt |
| 10 | TMR2_IRQHandler | Timer 2 Interrupt |
| 11 | TMR3_IRQHandler | Timer 3 Interrupt |
| 12 | TMR4_IRQHandler | Timer 4 (LPTMR0) Interrupt |
| 13 | TMR5_IRQHandler | Timer 5 (LPTMR1) Interrupt |
| 14 | I2C0_IRQHandler | I ² C Port 0 Interrupt |
| 15 | UART0_IRQHandler | UART Port 0 Interrupt |
| 16 | - | Reserved |
| 17 | I2C1_IRQHandler | I ² C Port 1 Interrupt |
| 18 | UART1_IRQHandler | UART Port 1 Interrupt |
| 19 | UART2_IRQHandler | UART Port 2 Interrupt |
| 20 | I2C2_IRQHandler | I ² C Port 2 Interrupt |
| 21 | UART3_IRQHandler | UART3 (LPUART0) Interrupt |
| 22 | SPI1_IRQHandler | SPI Port 1 Interrupt |
| 23 | WUT_IRQHandler | Wakeups Timer Interrupt |
| 24 | FLC0_IRQHandler | Flash Controller 0 Interrupt |
| 25 | GPIO0_IRQHandler | GPIO Port 0 Interrupt |
| 26 | GPIO1_IRQHandler | GPIO Port 1 Interrupt |
| 27 | GPIO2_IRQHandler | GPIO Port 2 Interrupt |
| 28 | DMA0_IRQHandler | DMA0 Interrupt |
| 29 | DMA1_IRQHandler | DMA1 Interrupt |
| 30 | DMA2_IRQHandler | DMA2 Interrupt |
| 31 | DMA3_IRQHandler | DMA3 Interrupt |
| 32:45 | - | Reserved |
| 46 | AES_IRQHandler | AES Interrupt |
| 47 | TRNG_IRQHandler | TRNG Interrupt |
| 48 | WDT1_IRQHandler | Watchdog Timer 1 (LPWDT0) Interrupt |

| Exception (Interrupt) Number | Name | Description |
|---------------------------------|---------------------|--|
| 49 | DVS_IRQHandler | Digital Voltage Scaling Interrupt |
| 50 | SIMO_IRQHandler | Single Input Multiple Output Interrupt |
| 51 | - | Reserved |
| 52 | PT_IRQHandler | Pulse Train Interrupt |
| 53 | ADC_IRQHandler | ADC Interrupt |
| 54 | OWM_IRQHandler | 1-Wire Master Interrupt |
| 55 | I2S_IRQHandler | I2S Interrupt |
| 56 | CNN_FIFO_IRQHandler | CNN TX FIFO Interrupt |
| 57 | CNN_IRQHandler | CNN Interrupt |
| 58 | - | Reserved |
| 59 | PCIF_IRQHandler | Parallel Camera Interface Interrupt |

5. General-Purpose I/O and Alternate Function Pins (GPIO)

General-purpose I/O (GPIO) pins can be individually configured to operate in a digital I/O mode or in an alternate function (AF) mode which maps a signal associated with an enabled peripheral to that GPIO. The GPIO support dynamic switching between I/O mode and alternate function mode. Configuring a pin for an alternate function supersedes its use as a digital I/O, however the state of the GPIO can still be read via the [GPIOn_IN](#) register.

The electrical characteristics of a GPIO pin are identical whether the pin is configured as an I/O or alternate function, except where explicitly noted in the data sheet electrical characteristics tables.

GPIO are logically divided into ports of 32 pins. Package variants may not implement all pins of a specific 32-bit GPIO port.

Each pin of a port has an interrupt function that can be independently enabled and configured as a level- or edge-sensitive interrupt. All GPIOs of a given port share the same interrupt vector as detailed in the section [GPIO Interrupt Handling](#).

Note: GPIO3 is controlled differently and has different features than the other GPIO ports in the MAX78000. Details for using GPIO3 are covered in the system chapter.

The features for each GPIO pin include:

- Full CMOS outputs with configurable drive strength settings.
- Input modes/options:
 - ◆ High impedance
 - ◆ Weak pullup/pulldown
 - ◆ Strong pullup/pulldown
- Output data can be from [GPIOn_OUT](#) register or an enabled peripheral
- Input data can be read from [GPIOn_IN](#) input register or the enabled peripheral
- Bit set and clear registers for efficient bit-wise write access to the pins and configuration registers
- Wake from low-power modes using edge triggered inputs
- Selectable GPIO voltage supply for GPIO0, GPIO1 and GPIO2:
 - ◆ VDDIO
 - ◆ VDDIOH
- Selectable interrupt events:
 - ◆ Level triggered low
 - ◆ Level triggered high
 - ◆ Edge triggered rising edge
 - ◆ Edge triggered falling edge
 - ◆ Edge triggered rising or falling edge
- All GPIO pins default to input mode with weak-pullup during power-on-reset events

5.1 Instances

Table 5-1: MAX78000 GPIO Pin Count shows the number of GPIO available on each IC package. Some packages and part numbers do not implement all bits of a 32-bit GPIO port. Register fields corresponding to unimplemented GPIO contain indeterminate values and should not be modified.

Table 5-1: MAX78000 GPIO Pin Count

| Package | GPIO | PINS |
|----------|-------------|------|
| 81-CTBGA | GPIO0[30:0] | 31 |
| | GPIO1[9:0] | 10 |
| | GPIO2[7:0] | 8 |
| | GPIO3[1:0] | 2 |

Note: Refer to the Single Inductor Multiple Output (SIMO) Power Supply

The Single Inductor Multiple Output (SIMO) switch mode power supply allows the device to operate autonomously from a single lithium cell. The SIMO provides three buck switching regulators (VREGO_A thru VREGO_C). Each of the three regulator voltages can be controlled by either CPU individually. For the SIMO to operate properly, the three buck regulator outputs must drive the power supply pins of the device as follows in *Table 3-18*.

5.1.1 Power Supply Monitor

The system also provides a power monitor that monitors the external power supplies relative to the on-chip bandgap voltage. The following power supplies are monitored:

- VCOREA (VCOREA) Digital Core Supply Voltage A for the Always-On Domain (AoD)
- VCOREB (VCOREB) Digital Core Supply Voltage B
- VDDIO (VDDIO) GPIO Supply Voltage
- VDDIOH (VDDIOH) GPIO High Supply Voltage
- VDDA (VDDA) Always-On Domain Analog Supply Voltage
- VREGI (VREGI) Input Supply Voltage, Battery

If the voltage drops below the trigger threshold, all registers and peripherals in that power domain are reset. This improves reliability and safety by guarding against a low voltage condition corrupting the contents of the registers and the device state.

Refer to the data sheet electrical characteristics for the trigger threshold values and power fail reset voltages.

Table 3-18: SIMO Power Supply Device Pin Connectivity

| SIMO Supply Output Pin | Connection | Device Power Supply Input Pin | Supply Monitor Reset Action |
|------------------------|------------|-------------------------------|--|
| VREGO_A | → | VDDA | POR |
| VREGO_B | → | VCOREB | POR |
| VREGO_C | → | VCOREA | POR |
| - | - | VREGI | POR |
| - | - | VDDIO Power On | GPIO pad held in reset until the voltage rises above threshold |
| - | - | VDDIOH Power On | GPIO pad held in reset until the voltage rises above threshold |
| - | - | VDDIO | GPIO pad logic enters POR |
| - | - | VDDIOH | GPIO pad logic enters POR |

Single Inductor Multiple Output Registers (SIMO)

See *Table 2-4: APB Peripheral Base Address Map* for the SIMO Controller Peripheral Base Address.

Table 3-19: SIMO Controller Register Summary

| Offset | Register | Access | Name |
|----------|---------------------|--------|---|
| [0x0004] | <i>SIMO_VREGO_A</i> | R/W | Buck Voltage Regulator A Control Register |
| [0x0008] | <i>SIMO_VREGO_B</i> | R/W | Buck Voltage Regulator B Control Register |
| [0x000C] | <i>SIMO_VREGO_C</i> | R/W | Buck Voltage Regulator C Control Register |
| [0x0014] | <i>SIMO_IPKA</i> | RO | Reserved. Do not modify this register. |
| [0x0018] | <i>SIMO_IPKB</i> | RO | Reserved. Do not modify this register. |
| [0x001C] | <i>SIMO_MAXTON</i> | RO | Reserved. Do not modify this register. |
| [0x0020] | <i>SIMO_ILOAD_A</i> | RO | Reserved. Do not modify this register. |
| [0x0024] | <i>SIMO_ILOAD_B</i> | RO | Reserved. Do not modify this register. |
| [0x0028] | <i>SIMO_ILOAD_C</i> | RO | Reserved. Do not modify this register. |

| Offset | Register | Access | Name |
|----------|------------------------------|--------|--|
| [0x0030] | <i>SIMO_BUCK_ALERT_THR_A</i> | RO | Reserved. Do not modify this register. |
| [0x0034] | <i>SIMO_BUCK_ALERT_THR_B</i> | RO | Reserved. Do not modify this register. |
| [0x0038] | <i>SIMO_BUCK_ALERT_THR_C</i> | RO | Reserved. Do not modify this register. |
| [0x0040] | <i>SIMO_BUCK_OUT_READY</i> | RO | Buck Regulator Output Ready Register |
| [0x0044] | <i>SIMO_ZERO_CROSS_CAL_A</i> | RO | Reserved. Do not modify this register. |
| [0x0048] | <i>SIMO_ZERO_CROSS_CAL_B</i> | RO | Reserved. Do not modify this register. |
| [0x004C] | <i>SIMO_ZERO_CROSS_CAL_C</i> | RO | Reserved. Do not modify this register. |

5.1.2 Single Inductor Multiple Output (SIMO) Registers Details

Table 3-20: SIMO Buck Voltage Regulator A Control Register

| SIMO Buck Voltage Regulator A Control | | | SIMO_VREGO_A | | [0x0004] |
|---------------------------------------|--------|--------|--------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | - | Reserved Do not modify this field. | |
| 7 | rangea | R/W | 1 | Regulator Output A Range Selects the Regulator output range for VREGO_A. 0: 0.5V to 1.77 1: 0.6V to 1.87V | |
| 6:0 | vseta | R/W | 0x78h | Regulator Output A Voltage Each bit increment in this field represents 10mV allowing output voltage settings from the minimum to the maximum of the <i>SIMO_VREGO_A.rangea</i> selected. $SIMO_VREGO_A.rangea = 1: Output\ Voltage = 0.6V + (10mV \times vseta)$ $SIMO_VREGO_A.rangea = 0: Output\ Voltage = 0.5V + (10mV \times vseta)$ Default: 0x78 = <i>SIMO_VREGO_A.rangea</i> = 0, Output Voltage = 1.7V; <i>SIMO_VREGO_A.rangea</i> = 1, Output Voltage = 1.8V Warning: When this regulator is connected as shown in Table 3-18: SIMO Power Supply Device Pin Connectivity, the following apply: <ol style="list-style-type: none"> The maximum setting for this regulator must be followed for VDDA as indicated in the device datasheet. Setting the regulator to a voltage below the Power-Fail Reset Voltage for VDDA will initiate the Power Monitor Reset Action. | |

Table 3-21: SIMO Buck Voltage Regulator B Control Register

| SIMO Buck Voltage Regulator B Control | | | SIMO_VREGO_B | | [0x0008] |
|---------------------------------------|--------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | - | Reserved Do not modify this field. | |
| 7 | rangeb | R/W | 1 | Regulator Output B Range Selects the Regulator output range for VREGO_B. 0: 0.5V to 1.77 1: 0.6V to 1.87V | |

| SIMO Buck Voltage Regulator B Control | | | SIMO_VREGO_B | | [0x0008] |
|---------------------------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 6:0 | vsetb | R/W | 0x32h | Regulator Output Voltage Each bit increment in this field represents 10mV allowing output voltage settings from the minimum to the maximum of the <i>SIMO_VREGO_B.rangeb</i> selected. $SIMO_VREGO_B.rangeb = 1$; Output Voltage = $0.6V + (10mV \times vsetb)$ $SIMO_VREGO_B.rangeb = 0$; Output Voltage = $0.5V + (10mV \times vsetb)$ Setting this field to 0x7F results in the maximum output voltage per the <i>SIMO_VREGO_B.rangeb</i> selected (1.77V or 1.87V) Default: 0x32 = <i>SIMO_VREGO_B.rangeb</i> = 0, Output Voltage = 1.0V; <i>SIMO_VREGO_B.rangeb</i> = 1, Output Voltage = 1.1V Warning: When this regulator is connected as shown in Table 3-18: SIMO Power Supply Device Pin Connectivity, the following apply: <ol style="list-style-type: none"> 1. The maximum setting for this regulator must be followed for VCOREB as indicated in the device datasheet. 2. Setting the regulator to a voltage below the Power-Fail Reset Voltage for VCOREB will initiate the Power Monitor Reset Action. | |

Table 3-22: SIMO Buck Voltage Regulator C Control Register

| SIMO Buck Voltage Regulator C Control | | | SIMO_VREGO_C | | [0x000C] |
|---------------------------------------|--------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | - | Reserved Do not modify this field. | |
| 7 | rangec | R/W | 1 | Regulator Output Range Selects the Regulator output range for VREGO_C. 0: 0.5V to 1.77 1: 0.6V to 1.87V | |
| 6:0 | vsetc | R/W | 0x32h | Regulator Output Voltage Each increment in the register represents 10mV. $SIMO_VREGO_C.rangec = 1$; Output Voltage = $0.6V + (10mV \times vsetc)$ $SIMO_VREGO_C.rangec = 0$; Output Voltage = $0.5V + (10mV \times vsetc)$ Setting this field to 0x7F results in the maximum output voltage per the <i>SIMO_VREGO_C.rangec</i> selected (1.77V or 1.87V) Default: 0x32 = <i>SIMO_VREGO_C.rangec</i> = 0, Output Voltage = 1.0V; <i>SIMO_VREGO_C.rangec</i> = 1, Output Voltage = 1.1V Warning: When this regulator is connected as shown in Table 3-18: SIMO Power Supply Device Pin Connectivity, the following apply: <ol style="list-style-type: none"> 1. The maximum setting for this regulator must be followed for VCOREA as indicated in the device datasheet. 2. Setting the regulator to a voltage below the Power-Fail Reset Voltage for VCOREA will initiate the Power Monitor Reset Action. | |

Table 3-23: SIMO High Side FET Peak Current VREGO_A VREGO_B Register

| SIMO High Side FET Peak Current VREGO_A VREGO_B | | | SIMO_IPKA | | [0x0014] |
|---|---------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | - | Reserved Reserved. Do not modify this field. | |
| 7:4 | ipksetb | R/W | 8 | Reserved Reserved. Do not modify this field. | |
| 3:0 | ipkseta | R/W | 8 | Reserved Reserved. Do not modify this field. | |

Table 3-24: SIMO High Side FET Peak Current VREGO_C Register

| SIMO High Side FET Peak Current VREGO_C VREGO_D | | | SIMO_IPKB | | [0x0018] |
|---|---------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | R/W | - | Reserved Do not modify this field. | |
| 3:0 | ipksetc | R/W | 8 | Reserved Reserved. Do not modify this field. | |

Table 3-25: SIMO Maximum High Side FET Time On Register

| SIMO Maximum High Side FET Time On | | | SIMO_MAXTON | | [0x001C] |
|------------------------------------|--------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | R/W | - | Reserved Do not modify this field. | |
| 3:0 | tonset | R/W | 0x8h | Reserved Do not modify this field. | |

Table 3-26: SIMO Buck Cycle Count VREGO_A Register

| SIMO Buck Cycle Count VREGO_A | | | SIMO_ILOAD_A | | [0x0020] |
|-------------------------------|--------|--------|--------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | iloada | RO | 0 | Reserved Do not modify this field. | |

Table 3-27: SIMO Buck Cycle Count VREGO_B Register

| SIMO Buck Cycle Count VREGO_B | | | SIMO_ILOAD_B | | [0x0024] |
|-------------------------------|-------|--------|--------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |

| SIMO Buck Cycle Count VREGO_B | | | | SIMO_ILOAD_B | [0x0024] |
|-------------------------------|--------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 7:0 | iloadb | RO | 0 | Reserved Do not modify this field. | |

Table 3-28: SIMO Buck Cycle Count VREGO_C Register

| SIMO Buck Cycle Count VREGO_C | | | | SIMO_ILOAD_C | [0x0028] |
|-------------------------------|--------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | iloadc | RO | 0 | Reserved Do not modify this field. | |

Table 3-29: SIMO Buck Cycle Count Alert VREGO_A Register

| SIMO Buck Cycle Count Alert VREGO_A | | | | SIMO_BUCK_ALERT_THR_A | [0x0030] |
|-------------------------------------|----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | buckthra | R/W | 0 | Reserved Do not modify this field. | |

Table 3-30: SIMO Buck Cycle Count Alert VREGO_B Register

| SIMO Buck Cycle Count Alert VREGO_A | | | | SIMO_BUCK_ALERT_THR_B | [0x0034] |
|-------------------------------------|----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | buckthrb | R/W | 0 | Reserved Do not modify this field. | |

Table 3-31: SIMO Buck Cycle Count Alert VREGO_C Register

| SIMO Buck Cycle Count Alert VREGO_A | | | | SIMO_BUCK_ALERT_THR_C | [0x0038] |
|-------------------------------------|----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify this field. | |
| 7:0 | buckthrc | R/W | 0 | Reserved Do not modify this field. | |

Table 3-32: SIMO Buck Regulator Output Ready Register

| SIMO Buck Regulator Output Ready | | | | SIMO_BUCK_OUT_READY | [0x0040] |
|----------------------------------|-------------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | - | Reserved Do not modify this field. | |
| 3 | buckoutrdya | RO | 0 | VREGO_A Output Ready When <i>SIMO_VREGO_A.vseta</i> changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready | |
| 2 | buckoutrdyb | RO | 0 | VREGO_B Output Ready When <i>SIMO_VREGO_B.vsetb</i> changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready | |
| 1 | buckoutrdyc | RO | 0 | VREGO_C Output Ready When <i>SIMO_VREGO_C.vsetc</i> changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready | |
| 0 | - | RO | 0 | Reserved Do not modify | |

Table 3-33: SIMO Zero Cross Calibration VREGO_A Register

| SIMO Zero Cross Calibration VREGO_A | | | | SIMO_ZERO_CROSS_CAL_A | [0x0044] |
|-------------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | RO | - | Reserved Do not modify this field. | |
| 4:0 | zxcla | RO | 0 | Reserved Do not modify this field. | |

Table 3-34: SIMO Zero Cross Calibration VREGO_B Register

| SIMO Zero Cross Calibration VREGO_B | | | | SIMO_ZERO_CROSS_CAL_B | [0x0048] |
|-------------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | RO | - | Reserved Do not modify this field. | |
| 4:0 | zxclb | RO | 0 | Reserved Do not modify this field. | |

Table 3-35: SIMO Zero Cross Calibration VREGO_C Register

| SIMO Zero Cross Calibration VREGO_C | | | SIMO_ZERO_CROSS_CAL_C | | [0x004C] |
|-------------------------------------|--------|--------|-----------------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | RO | - | Reserved Do not modify this field. | |
| 4:0 | zxclc | RO | 0 | Reserved Do not modify this field. | |
| 4:0 | zxclid | RO | 0 | Reserved Do not modify this field. | |

[Low-Power General Control Registers \(LPGCR\)](#) and [Power Sequencer Registers \(PWRSEQ\)](#) for details on using GPIO3.

Note: Refer to the MAX78000 device datasheet for a description of alternate functions for each GPIO port pin.

5.2 Configuration

Each device pin can be individually configured as a GPIO or an alternate function. The correct alternate function setting must be selected for each pin of a given multi-pin peripheral for proper operation.

5.2.1 Pin Function Configuration

Table 5-2: MAX78000 GPIO Pin Function Configuration depicts the bit settings for the `GPIOn_EN0`, `GPIOn_EN1`, and `GPIOn_EN2` registers to configure the function of the GPIOn port pins. Each of the bits within these registers represents the configuration of a single pin on the GPIO port. For example, `GPIO0_EN0.config[25]`, `GPIO0_EN1.config[25]`, and `GPIO0_EN2.config[25]` all represent configuration for device pin P0.25. See [Table 5-6: MAX78000 GPIO Alternate Function Configuration](#) for a detailed example of how each of these bits applies to each of the GPIO device pins.

Table 5-2: MAX78000 GPIO Pin Function Configuration

| MODE | <code>GPIOn_EN0.config[pin]</code> | <code>GPIOn_EN1.config[pin]</code> | <code>GPIOn_EN2.config[pin]</code> |
|-------------------------|------------------------------------|------------------------------------|------------------------------------|
| AF1 | 0 | 0 | 0 |
| AF2 | 0 | 1 | 0 |
| I/O (transition to AF1) | 1 | 0 | 0 |
| I/O (transition to AF2) | 1 | 1 | 0 |

5.2.2 Input Mode Configuration

Table 5-3: MAX78000 Input Mode Configuration depicts the bit settings for the digital I/O input mode. Each of the bits within these registers represents the configuration of a single pin on the GPIO port. For example, `GPIO0_PAD_CFG2.config[25]`, `GPIO0_PAD_CFG1.config[25]`, `GPIO0_PS.str[25]`, and `GPIO0_VSEL.vlvl[25]` all represent configuration for device pin P0.25. See [Table 5-9: MAX78000 GPIO Pullup/Pulldown/Drive Strength/Voltage Configuration Reference](#) for a detailed example of how each of these bits applies to each of the GPIO device pins. Refer to the data sheet for details of specific electrical characteristics.

Table 5-3: MAX78000 Input Mode Configuration

| Input Mode | Mode Select | | Pullup/Pulldown Strength | Power Supply |
|----------------|---|---|--------------------------------|-----------------------------------|
| | <code>GPIOn_PAD_CFG2.config[pin]</code> | <code>GPIOn_PAD_CFG1.config[pin]</code> | <code>GPIOn_PS.str[pin]</code> | <code>GPIOn_VSEL.vlvl[pin]</code> |
| High-impedance | 0 | 0 | N/A | N/A |

| Input Mode | Mode Select | | Pullup/Pulldown Strength | Power Supply |
|----------------------------------|------------------------------------|------------------------------------|---------------------------|-----------------------------|
| | <i>GPIOOn_PAD_CFG2.config[pin]</i> | <i>GPIOOn_PAD_CFG1.config[pin]</i> | <i>GPIOOn_PS.str[pin]</i> | <i>GPIOOn_VSEL.vlv[pin]</i> |
| Weak Pull-up to VDDIO (1MΩ) | 0 | 1 | 0 | 0 |
| Strong Pull-up to VDDIO (25KΩ) | 0 | 1 | 1 | 0 |
| Weak Pull-down to VDDIO (1MΩ) | 1 | 0 | 0 | 1 |
| Strong Pull-down to VDDIO (25KΩ) | 1 | 0 | 1 | 1 |
| Reserved | 1 | 1 | N/A | N/A |

Table 5-4: MAX78000 Output Mode Configuration shows the configuration options for digital I/O in output mode. Each of the bits within these registers represents the configuration of a single pin on the GPIO port. For example, *GPIO0_DS.str[25], GPIO0_DS1.str[25], and GPIO0_VSEL.vlv[25]* all represent configuration for device pin P0.25. See *Table 5-9: MAX78000 GPIO Pullup/Pulldown/Drive Strength/Voltage Configuration Reference* for a detailed example of how each of these bits applies to each of the GPIO device pins. Refer to the data sheet for details of specific electrical characteristics.

Table 5-4: MAX78000 Output Mode Configuration

| Input Mode | Drive Strength | | Power Supply |
|--|----------------------------|---------------------------|-----------------------------|
| | <i>GPIOOn_DS1.str[pin]</i> | <i>GPIOOn_DS.str[pin]</i> | <i>GPIOOn_VSEL.vlv[pin]</i> |
| Output Drive Strength 0, VDDIO Supply | 0 | 0 | 0 |
| Output Drive Strength 1, VDDIO Supply | 0 | 1 | 0 |
| Output Drive Strength 2, VDDIO Supply | 1 | 0 | 0 |
| Output Drive Strength 3, VDDIO Supply | 1 | 1 | 0 |
| Output Drive Strength 0, VDDIOH Supply | 0 | 0 | 1 |
| Output Drive Strength 1, VDDIOH Supply | 0 | 1 | 1 |
| Output Drive Strength 2, VDDIOH Supply | 1 | 0 | 1 |
| Output Drive Strength 3, VDDIOH Supply | 1 | 1 | 1 |

Each GPIO port is assigned a dedicated interrupt vector as shown in *Table 5-5: MAX78000 GPIO Port Interrupt Vector Mapping*.

Table 5-5: MAX78000 GPIO Port Interrupt Vector Mapping

| GPIO Interrupt Source | GPIO Interrupt Status Register | CM4 Interrupt Vector Number | RV32 Interrupt Vector Number | GPIO Interrupt Vector |
|-----------------------|--------------------------------|-----------------------------|------------------------------|-----------------------|
| GPIO0[31:0] | <i>GPIOOn_INT_STAT</i> | 40 | 25 | GPIO0_IRQHandler |
| GPIO1[9:0] | <i>GPIOOn_INT_STAT</i> | 41 | 26 | GPIO1_IRQHandler |
| GPIO2[7:0] | <i>GPIOOn_INT_STAT</i> | 42 | 27 | GPIO2_IRQHandler |



5.3 Reference Tables

The tables in this section provide example references for register bit assignment to configure a device's GPIO pins.

Table 5-6: MAX78000 GPIO Alternate Function Configuration Reference

| Device Pin | Alternate Function Configuration Bits | | |
|------------|---------------------------------------|-----------------------------|-----------------------------|
| P0.0 | <i>GPIO0_EN0.config[0]</i> | <i>GPIO0_EN1.config[0]</i> | <i>GPIO0_EN2.config[0]</i> |
| P0.1 | <i>GPIO0_EN0.config[1]</i> | <i>GPIO0_EN1.config[1]</i> | <i>GPIO0_EN2.config[1]</i> |
| ... | ... | ... | ... |
| P0.30 | <i>GPIO0_EN0.config[30]</i> | <i>GPIO0_EN1.config[30]</i> | <i>GPIO0_EN2.config[30]</i> |
| P0.31 | <i>GPIO0_EN0.config[31]</i> | <i>GPIO0_EN1.config[31]</i> | <i>GPIO0_EN2.config[31]</i> |

Table 5-7: MAX78000 GPIO Output/Input Configuration Reference

| Device Pin | GPIO Output Enable | GPIO Output Write | GPIO Input Enable | GPIO Input Read |
|------------|----------------------------|--------------------------|---------------------------|---------------------------|
| P0.0 | <i>GPIO0_OUT_EN.en[0]</i> | <i>GPIO0_OUT.lvl[0]</i> | <i>GPIO0_IN_EN.en[0]</i> | <i>GPIO0_IN.input[0]</i> |
| P0.1 | <i>GPIO0_OUT_EN.en[1]</i> | <i>GPIO0_OUT.lvl[1]</i> | <i>GPIO0_IN_EN.en[1]</i> | <i>GPIO0_IN.input[1]</i> |
| ... | ... | ... | ... | ... |
| P0.30 | <i>GPIO0_OUT_EN.en[30]</i> | <i>GPIO0_OUT.lvl[30]</i> | <i>GPIO0_IN_EN.en[30]</i> | <i>GPIO0_IN.input[30]</i> |
| P0.31 | <i>GPIO0_OUT_EN.en[31]</i> | <i>GPIO0_OUT.lvl[31]</i> | <i>GPIO0_IN_EN.en[31]</i> | <i>GPIO0_IN.input[31]</i> |

Table 5-8: MAX78000 GPIO Interrupt Configuration Reference

| Device Pin | Enable | Status | Dual Edge | Polarity | Trigger | Wakeup |
|------------|----------------------------|-------------------------------|-----------------------------------|------------------------------|----------------------------------|-----------------------------|
| P0.0 | <i>GPIO0_INT_EN.en[0]</i> | <i>GPIO0_INT_STAT.int[0]</i> | <i>GPIO0_INT_DUAL_EDGE.en[0]</i> | <i>GPIO0_INT_POL.pol[0]</i> | <i>GPIO0_INT_MOD.trigger[0]</i> | <i>GPIO0_WAKE_EN.en[0]</i> |
| P0.1 | <i>GPIO0_INT_EN.en[1]</i> | <i>GPIO0_INT_STAT.int[1]</i> | <i>GPIO0_INT_DUAL_EDGE.en[1]</i> | <i>GPIO0_INT_POL.pol[1]</i> | <i>GPIO0_INT_MOD.trigger[1]</i> | <i>GPIO0_WAKE_EN.en[1]</i> |
| ... | ... | ... | ... | ... | ... | ... |
| P0.30 | <i>GPIO0_INT_EN.en[30]</i> | <i>GPIO0_INT_STAT.int[30]</i> | <i>GPIO0_INT_DUAL_EDGE.en[30]</i> | <i>GPIO0_INT_POL.pol[30]</i> | <i>GPIO0_INT_MOD.trigger[30]</i> | <i>GPIO0_WAKE_EN.en[30]</i> |
| P0.31 | <i>GPIO0_INT_EN.en[31]</i> | <i>GPIO0_INT_STAT.int[31]</i> | <i>GPIO0_INT_DUAL_EDGE.en[31]</i> | <i>GPIO0_INT_POL.pol[31]</i> | <i>GPIO0_INT_MOD.trigger[31]</i> | <i>GPIO0_WAKE_EN.en[31]</i> |

Table 5-9: MAX78000 GPIO Pullup/Pulldown/Drive Strength/Voltage Configuration Reference

| Device Pin | Pullup/Pulldown/Strength Select | | | Drive Strength | | Voltage |
|------------|---------------------------------|---------------------------------|------------------------|------------------------|-------------------------|---------------------------|
| P0.0 | <i>GPIO0_PAD_CFG1.config[0]</i> | <i>GPIO0_PAD_CFG2.config[0]</i> | <i>GPIO0_PS.str[0]</i> | <i>GPIO0_DS.str[0]</i> | <i>GPIO0_DS1.str[0]</i> | <i>GPIO_VSSEL.vlvl[0]</i> |
| P0.1 | <i>GPIO0_PAD_CFG1.config[1]</i> | <i>GPIO0_PAD_CFG2.config[1]</i> | <i>GPIO0_PS.str[1]</i> | <i>GPIO0_DS.str[1]</i> | <i>GPIO0_DS1.str[1]</i> | <i>GPIO_VSSEL.vlvl[0]</i> |

| Device Pin | Pullup/Pulldown/Strength Select | | | Drive Strength | | Voltage |
|------------|----------------------------------|----------------------------------|-------------------------|-------------------------|--------------------------|----------------------------|
| ... | ... | ... | ... | ... | ... | ... |
| P0.30 | <i>GPIO0_PAD_CFG1.config[30]</i> | <i>GPIO0_PAD_CFG2.config[30]</i> | <i>GPIO0_PS.str[30]</i> | <i>GPIO0_DS.str[30]</i> | <i>GPIO0_DS1.str[30]</i> | <i>GPIOn_VSEL.vlvl[31]</i> |
| P0.31 | <i>GPIO0_PAD_CFG1.config[31]</i> | <i>GPIO0_PAD_CFG2.config[31]</i> | <i>GPIO0_PS.str[31]</i> | <i>GPIO0_DS.str[31]</i> | <i>GPIO0_DS1.str[31]</i> | <i>GPIOn_VSEL.vlvl[31]</i> |

5.4 Usage

5.4.1 Reset State

During a power-on-reset event, each GPIO is reset to the default input mode with the weak pullup resistor enabled as follows:

1. The GPIO Configuration Enable bits shown in [Table 5-2](#) are set to I/O (transition to AF1) mode.
2. Input mode is enabled (*GPIOn_IN_EN.en[pin]* = 1).
3. High impedance mode enabled (*GPIOn_PAD_CFG1.config[pin]* = 0, *GPIOn_PAD_CFG2.config[pin]* = 0), pullup and pulldown disabled.
4. Output mode disabled (*GPIOn_OUT_EN.en[pin]* = 0)
5. Interrupt disabled (*GPIOn_INT_EN.en[pin]* = 0)

5.4.2 Input Mode Configuration

Perform the following steps to configure one or more pins for input mode:

1. Set the GPIO Configuration Enable bits shown in [Table 5-2](#) to any one of the I/O mode settings.
2. Configure the electrical characteristics of the pin as desired as shown in [Table 5-3](#).
3. Enable the input buffer connection to the GPIO pin by setting *GPIOn_IN_EN.en[pin]* to 1.
4. Read the input state of the pin using the *GPIOn_IN.input[pin]* field.

5.4.3 Output Mode Configuration

Perform the following steps to configure a pin for output mode:

1. Set the GPIO Configuration Enable bits shown in [Table 5-2](#) to any one of the I/O mode settings.
2. Configure the electrical characteristics of the pin as desired as shown in [Table 5-4](#).
3. Set the output logic high or logic low using the *GPIOn_OUT.lvl[pin]* bit.
4. Enable the output buffer for the pin by setting *GPIOn_OUT_EN.en[pin]* to 1.

5.4.4 Alternate Function Configuration

Most GPIO support one or more alternate functions which are selected with the GPIO Configuration Enable bits shown in [Table 5-2](#). The bits that select the AF must only be changed while the pin is in one of the I/O modes (*GPIOn_EN0* = 1). The

specific I/O mode must match the desired AF. For example, if a transition to AF1 is desired, first select the setting corresponding to I/O (transition to AF1). Then enable the desired mode by selecting the AF1 mode.

1. Set the GPIO Configuration Enable bits shown in [Table 5-2](#) to the I/O mode that corresponds with the desired new AF setting. For example, select “I/O (transition to AF1)” if switching to AF1. Switching between different I/O mode settings will not affect the state or electrical characteristics of the pin.
2. Configure the electrical characteristics of the pin. See [Table 5-3](#) if the assigned alternate function will use the pin as an input. See [Table 5-4](#) if the assigned alternate function will use the pin as an output.
3. Set the GPIO Configuration Enable bits shown in [Table 5-2](#) to the desired alternate function.

5.5 Configuring GPIO (External) Interrupts

Each GPIO pin supports external interrupt events when the GPIO is configured for I/O mode and the input mode is enabled. If the GPIO is configured as a peripheral alternate function, the interrupts are peripheral controlled.

GPIO interrupts can be individually enabled and configured as an edge or level triggered independently on a pin-by-pin basis. The edge trigger can be a rising, falling, or both transitions.

Each GPIO pin has a dedicated status bit in its corresponding [`GPIOn_INT_STAT`](#) register. A GPIO interrupt will occur when the a status bit transitions from 0 to 1 if the corresponding bit is set in the corresponding [`GPIOn_INT_EN`](#) register. Note that the interrupt status bit will always be set when the current interrupt configuration event occurs, but an interrupt will only be generated if explicitly enabled.

The following procedure details the steps for enabling ACTIVE mode interrupt events for a GPIO pin:

1. Disable interrupts by setting the [`GPIOn_INT_EN.en\[pin\]`](#) field to 0. This will prevent any new interrupts on the pin from triggering but will not clear previously triggered (pending) interrupts. The application can disable all interrupts for a GPIO port by writing 0 to the [`GPIOn_IN_EN`](#) register. To maintain previously enabled interrupts, read the [`GPIOn_IN_EN`](#) register and save the state prior to setting the register to 0.
2. Clear pending interrupts by writing 1 to the [`GPIOn_INT_CLR.clr\[pin\]`](#) bit.
3. Configure the pin for the desired interrupt event
4. Set [`GPIOn_INT_MOD.trigger\[pin\]`](#) to select the desired interrupt.
5. For level triggered interrupts, the interrupt triggers on an input high ([`GPIOn_INT_POL.pol\[pin\] = 0`](#)) or input low level.
6. For edge triggered interrupts, the interrupt triggers on a transition from low to high ([`GPIOn_INT_POL.pol\[pin\] = 0`](#)) or high to low ([`GPIOn_INT_POL.pol\[pin\] = 1`](#)).
7. Optionally set [`GPIOn_INT_DUAL_EDGE.en\[pin\]`](#) to 1 to trigger on both the rising and falling edges of the input signal.
 - a. Set [`GPIOn_INT_EN.en\[pin\]`](#) to 1 to enable the interrupt for the pin.

5.5.1 GPIO Interrupt Handling

Each GPIO port is assigned its own dedicated interrupt vector as shown in [Table 5-5: MAX78000 GPIO Port Interrupt Vector Mapping](#).

Handle GPIO interrupts in a firmware interrupt vector, complete the following steps:

1. Read the [*GPIOn_INT_STAT*](#) register to determine the GPIO pin that triggered the interrupt.
2. Complete interrupt tasks associated with the interrupt source pin (application defined).
3. Clear the interrupt flag in the [*GPIOn_INT_STAT*](#) register by writing a 1 to the [*GPIOn_INT_CLR*](#) bit position that triggered the interrupt. This also clears and rearms the edge detectors for edge triggered interrupts.
4. Return from the interrupt vector handler.

5.5.2 Using GPIO for Wakeup from Low Power Modes

Low-power modes support an asynchronous wakeup from edge triggered interrupts on the GPIO ports. Level triggered interrupts are not supported for wakeup because the system clock must be active to detect levels.

A single wakeup interrupt vector, `GPIOWAKE_IRQHandler`, is assigned for all pins of all GPIO ports. When the GPIO wakeup event occurs, the application software must interrogate each [*GPIOn_INT_STAT*](#) register to determine which external port pin caused the wake-up event.

Table 5-10: MAX78000 GPIO Wakeup Interrupt Vector

| GPIO Wake Interrupt Source | GPIO Wake Interrupt Status Register | CM4 Interrupt Vector Number | RV32 Interrupt Vector Number | GPIO Wakeup Interrupt Vector |
|----------------------------|---------------------------------------|-----------------------------|------------------------------|----------------------------------|
| GPIO0 | <i>GPIOn_INT_STAT</i> | 70 | 6 | <code>GPIOWAKE_IRQHandler</code> |
| GPIO1 | <i>GPIOn_INT_STAT</i> | 70 | 6 | <code>GPIOWAKE_IRQHandler</code> |
| GPIO2 | <i>GPIOn_INT_STAT</i> | 70 | 6 | <code>GPIOWAKE_IRQHandler</code> |

To enable low power mode (**SLEEP**, **LPM**, **UPM**, **STANDBY** and **BACKUP**) wakeup from GPIO0, GPIO1 or GPIO2 using an external GPIO interrupt, complete the following steps:

1. Clear pending interrupt flags by writing a logic 1 to [*GPIOn_INT_CLR.pin*](#).
2. Activate the GPIO wakeup function by writing a logic 1 to [*GPIOn_WAKE_EN.en.pin*](#).
3. Configure the power manager to use the GPIO as a wakeup source by [*GCR_PM gpio_we*](#) field to 1.

5.6 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 5-11: GPIO Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 5-11: GPIO Register Summary

| Offset | Register | Description |
|----------|--------------------------------------|--|
| [0x0000] | GPIO_n_EN0 | GPIO Port n Configuration Enable Bit 0 Register |
| [0x0004] | GPIO_n_EN0_SET | GPIO Port n Configuration Enable Atomic Set Bit 0 Register |
| [0x0008] | GPIO_n_EN0_CLR | GPIO Port n Configuration Enable Atomic Clear Bit 0 Register |
| [0x000C] | GPIO_n_OUT_EN | GPIO Port n Output Enable Register |
| [0x0010] | GPIO_n_OUT_EN_SET | GPIO Port n Output Enable Atomic Set Register |
| [0x0014] | GPIO_n_OUT_EN_CLR | GPIO Port n Output Enable Atomic Clear Register |
| [0x0018] | GPIO_n_OUT | GPIO Port n Output Register |
| [0x001C] | GPIO_n_OUT_SET | GPIO Port n Output Atomic Set Register |
| [0x0020] | GPIO_n_OUT_CLR | GPIO Port n Output Atomic Clear Register |
| [0x0024] | GPIO_n_IN | GPIO Port n Input Register |
| [0x0028] | GPIO_n_INT_MOD | GPIO Port n Interrupt Mode Register |
| [0x002C] | GPIO_n_INT_POL | GPIO Port n Interrupt Polarity Register |
| [0x0030] | GPIO_n_IN_EN | GPIO Port n Input Enable Register |
| [0x0034] | GPIO_n_INT_EN | GPIO Port n Interrupt Enable Register |
| [0x0038] | GPIO_n_INT_EN_SET | GPIO Port n Interrupt Enable Atomic Set Register |
| [0x003C] | GPIO_n_INT_EN_CLR | GPIO Port n Interrupt Enable Atomic Clear Register |
| [0x0040] | GPIO_n_INT_STAT | GPIO Port n Interrupt Status Register |
| [0x0048] | GPIO_n_INT_CLR | GPIO Port n Interrupt Clear Register |
| [0x004C] | GPIO_n_WAKE_EN | GPIO Port n Wakeup Enable Register |
| [0x0050] | GPIO_n_WAKE_EN_SET | GPIO Port n Wakeup Enable Atomic Set Register |
| [0x0054] | GPIO_n_WAKE_EN_CLR | GPIO Port n Wakeup Enable Atomic Clear Register |
| [0x005C] | GPIO_n_INT_DUAL_EDGE | GPIO Port n Interrupt Dual Edge Mode Register |
| [0x0060] | GPIO_n_PAD_CFG1 | GPIO Port n Pad Configuration 1 Register |
| [0x0064] | GPIO_n_PAD_CFG2 | GPIO Port n Pad Configuration 2 Register |
| [0x0068] | GPIO_n_EN1 | GPIO Port n Configuration Enable Bit 1 Register |
| [0x006C] | GPIO_n_EN1_SET | GPIO Port n Configuration Enable Atomic Set Bit 1 Register |
| [0x0070] | GPIO_n_EN1_CLR | GPIO Port n Configuration Enable Atomic Clear Bit 1 Register |
| [0x0074] | GPIO_n_EN2 | GPIO Port n Configuration Enable Bit 2 Register |
| [0x0078] | GPIO_n_EN2_SET | GPIO Port n Configuration Enable Atomic Set Bit 2 Register |
| [0x007C] | GPIO_n_EN2_CLR | GPIO Port n Configuration Enable Atomic Clear Bit 2 Register |
| [0x00B0] | GPIO_n_DS | GPIO Port n Output Drive Strength Bit 0 Register |
| [0x00B4] | GPIO_n_DS1 | GPIO Port n Output Drive Strength Bit 1 Register |
| [0x00B8] | GPIO_n_PS | GPIO Port n Pulldown/Pullup Strength Select Register |
| [0x00C0] | GPIO_n_VSEL | Not Available on MAX78000 |

5.6.1 Register Details

Table 5-12: GPIO Port n Configuration Enable Bit 0 Register

| GPIO Port n Configuration Enable Bit 0 | | | | GPIOn_EN0 | [0x0000] |
|--|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | config | R/W | 1 | GPIO Configuration Enable Bit 0 These bits, in conjunction with bits in Table 5-2: MAX78000 GPIO Pin Function Configuration configure the corresponding device pin for digital I/O or an alternate function mode. This field can be modified directly by writing to this register or indirectly through GPIOn_EN0_SET or GPIOn_EN0_CLR . <i>Note: MAX78000 GPIO Alternate Function Configuration</i> depicts a detailed example of how each of these bits applies to each of the GPIO device pins <i>Note: Some GPIO are not implemented in all devices. The bits associated with unimplemented GPIO should not be changed from their default value.</i> <i>Note: This register setting does not affect input and interrupt functionality of the associated pin.</i> | |

Table 5-13: GPIO Port n Configuration Enable Atomic Set Bit 0 Register

| GPIO Port n Configuration Enable Atomic Set Bit 0 | | | | GPIOn_EN0_SET | [0x0004] |
|---|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Configuration Enable Atomic Set Bit 0 Writing 1 to one or more bits sets the corresponding bits in the GPIOn_EN0 register. 0: No effect. 1: Corresponding bits in GPIOn_EN0 register set to 1. | |

Table 5-14: GPIO Port n Configuration Enable Atomic Clear Bit 0 Register

| GPIO Port n Configuration Enable Atomic Clear Bit 0 | | | | GPIOn_EN0_CLR | [0x0008] |
|---|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Configuration Enable Atomic Clear Bit 0 Writing 1 to one or more bits clears the corresponding bits in the GPIOn_EN0 register. 0: No effect. 1: Corresponding bits in GPIOn_EN0 register cleared to 0. | |

Table 5-15: GPIO Port n Output Enable Register

| GPIO Port n Output Enable | | | | GPIOn_OUT_EN | [0x000C] |
|---------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | en | R/W | 0 | GPIO Output Enable Set bit to 1 to enable the output driver for the corresponding GPIO pin. A bit can be enabled directly by writing to this register or indirectly through GPIOn_OUT_EN_SET or GPIOn_OUT_EN_CLR . 0: Pin is set to input mode; output driver disabled. 1: Pin is set to output mode. | |

Table 5-16: GPIO Port n Output Enable Atomic Set Register

| GPIO Port n Output Enable Atomic Set | | | | GPIO _n _OUT_EN_SET | [0x0010] |
|--------------------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Output Enable Atomic Set Writing 1 to one or more bits sets the corresponding bits in the GPIO_n_OUT_EN register. 0: No effect. 1: Corresponding bits in GPIO_n_OUT_EN set to 1. | |

Table 5-17: GPIO Port n Output Enable Atomic Clear Register

| GPIO Port n Output Enable Atomic Clear | | | | GPIO _n _OUT_EN_CLR | [0x0014] |
|--|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Output Enable Atomic Clear Writing 1 to one or more bits sets the corresponding bits in the GPIO_n_OUT_EN register. 0: No effect. 1: Corresponding bits in GPIO_n_OUT_EN cleared to 0. | |

Table 5-18: GPIO Port n Output Register

| GPIO Port n Output | | | | GPIO _n _OUT | [0x0018] |
|--------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | lvl | R/W | 0 | GPIO Output Set the corresponding output pin high or low. 0: Drive the corresponding output pin low (logic 0). 1: Drive the corresponding output pin high (logic 1). | |

Table 5-19: GPIO Port n Output Atomic Set Register

| GPIO Port n Output Atomic Set | | | | GPIO _n _OUT_SET | [0x001C] |
|-------------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Output Atomic Set Writing 1 to one or more bits sets the corresponding bits in the GPIO_n_OUT register. 0: No effect. 1: Corresponding bits in GPIO_n_OUT_EN set to 1. | |

Table 5-20: GPIO Port n Output Atomic Clear Register

| GPIO Port n Output Atomic Clear | | | | GPIO _n _OUT_CLR | [0x0020] |
|---------------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | WO | 0 | GPIO Output Atomic Clear Writing 1 to one or more bits clears the corresponding bits in the GPIO_n_OUT register. 0: No effect. 1: Corresponding bits in GPIO_n_OUT_EN cleared to 0. | |

Table 5-21: GPIO Port n Input Register

| GPIO Port n Input | | GPIOn_IN | | | [0x0024] |
|-------------------|-------|----------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | input | RO | - | GPIO Input Returns the state of the input pin only if the corresponding bit in the GPIOn_IN_EN register is set. The state is not affected by the pin's configuration as an output or alternate function. 0: Input pin low 1: Input pin high. | |

Table 5-22: GPIO Port n Interrupt Mode Register

| GPIO Port n Interrupt Mode | | | | GPIOn_INT_MOD | [0x0028] |
|----------------------------|---------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | trigger | R/W | 0 | GPIO Interrupt Mode Selects interrupt mode for the corresponding GPIO pin. 0: Level triggered interrupt. 1: Edge triggered interrupt. <i>Note: This bit has no effect unless the corresponding bit in the GPIOn_INT_EN register is set.</i> | |

Table 5-23: GPIO Port n Interrupt Polarity Register

| GPIO Port n Interrupt Polarity | | | | GPIOn_INT_POL | [0x002C] |
|--------------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | pol | R/W | 0 | GPIO Interrupt Polarity Interrupt polarity selection bit for the corresponding GPIO pin. Level triggered mode (GPIOn_INT_MOD.trigger [pin] = 0): 0: Input low (logic 0) triggers interrupt. 1: Input high (logic 1) triggers interrupt. Edge triggered mode (GPIOn_INT_MOD.trigger [pin] = 1): 0: Falling edge triggers interrupt 1: Rising edge triggers interrupt. <i>Note: This bit has no effect unless the corresponding bit in the GPIOn_INT_EN register is set.</i> | |

Table 5-24: GPIO Port n Input Enable Register

| GPIO Port n Input Enable | | | | GPIOn_IN_EN | [0x0030] |
|--------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | en | R/W | 1 | GPIO Input Enable Connects the corresponding input pad to the specified input pin for reading the pin state using the GPIOn_IN register. 0: Input not connected. 1: Input pin connected to the pad for reading via GPIOn_IN register. | |

Table 5-25: GPIO Port n Interrupt Enable Register

| GPIO Port n Interrupt Enable | | | | GPIOOn_INT_EN | [0x0034] |
|------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | en | R/W | 0 | GPIO Interrupt Enable Enable or disable the interrupt for the corresponding GPIO pin. 0: GPIO interrupt disabled. 1: GPIO interrupt enabled. <i>Note: Disabling a GPIO interrupt does not clear pending interrupts for the associated pin. Use the GPIOOn_INT_CLR register to clear pending interrupts.</i> | |

Table 5-26: GPIO Port n Interrupt Enable Atomic Set Register

| GPIO Port Interrupt Enable Atomic Set | | | | GPIOOn_INT_EN_SET | [0x0038] |
|---------------------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Interrupt Enable Atomic Set Writing 1 to one or more bits sets the corresponding bits in the GPIOOn_INT_EN register. 0: No effect. 1: Corresponding bits in GPIOOn_INT_EN register set to 1. | |

Table 5-27: GPIO Port n Interrupt Enable Atomic Clear Register

| GPIO Port Interrupt Enable Atomic Clear | | | | GPIOOn_INT_EN_CLR | [0x003C] |
|---|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Interrupt Enable Atomic Clear Writing 1 to one or more bits clears the corresponding bits in the GPIOOn_INT_EN register. 0: No effect. 1: Corresponding bits in GPIOOn_INT_EN register cleared to 0. | |

Table 5-28: GPIO Port n Interrupt Status Register

| GPIO Port Interrupt Status | | | | GPIOOn_INT_STAT | [0x0040] |
|----------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | int | RO | 0 | GPIO Interrupt Status An interrupt is pending for the associated GPIO pin when this bit reads 1. 0: No interrupt pending for associated GPIO pin. 1: GPIO interrupt pending for associated GPIO pin. <i>Note: Write a 1 to the corresponding bit in the GPIOOn_INT_CLR register to clear the interrupt pending status flag.</i> | |

Table 5-29: GPIO Port n Interrupt Clear Register

| GPIO Port Interrupt Clear | | | | GPIOOn_INT_CLR | [0x0048] |
|---------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | clr | R/W1C | 0 | GPIO Interrupt Clear Write 1 to clear the associated interrupt status (GPIOOn_INT_STAT). 0: No effect on the associated GPIOOn_INT_STAT flag. 1: Clear the associated interrupt pending flag in the GPIOOn_INT_STAT register. | |

Table 5-30: GPIO Port n Wakeup Enable Register

| GPIO Port n Wakeup Enable | | | GPIOOn_WAKE_EN | | [0x004C] |
|---------------------------|-------|--------|----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | en | R/W | 0 | GPIO Wakeup Enable Enable the I/O as a wakeup from low power modes (SLEEP, DEEPSLEEP, BACKUP). 0: GPIO is not enabled as a wakeup source from low power modes. 1: GPIO is enabled as a wakeup source from low power modes. | |

Table 5-31: GPIO Port n Wakeup Enable Atomic Set Register

| GPIO Port Wakeup Enable Atomic Set | | | GPIOOn_WAKE_EN_SET | | [0x0050] |
|------------------------------------|-------|--------|--------------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Wakeup Enable Atomic Set Writing 1 to one or more bits sets the corresponding bits in the <i>GPIOOn_WAKE_ENr</i> register. 0: No effect. 1: Corresponding bits in <i>GPIOOn_WAKE_EN</i> register set to 1. | |

Table 5-32: GPIO Port n Wakeup Enable Atomic Clear Register

| GPIO Port Wakeup Enable Atomic Clear | | | GPIOOn_WAKE_EN_CLR | | [0x0054] |
|--------------------------------------|-------|--------|--------------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Wakeup Enable Atomic Clear Writing 1 to one or more bits clears the corresponding bits in the <i>GPIOOn_WAKE_ENr</i> register. 0: No effect. 1: Corresponding bits in <i>GPIOOn_WAKE_EN</i> register cleared to 0. | |

Table 5-33: GPIO Port n Interrupt Dual Edge Mode Register

| GPIO Port n Interrupt Dual Edge Mode | | | GPIOOn_INT_DUAL_EDGE | | [0x005C] |
|--------------------------------------|-------|--------|----------------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | en | R/W | 0 | GPIO Interrupt Dual-Edge Mode Select Setting this bit triggers interrupts on both the rising and falling edges of the corresponding GPIO if the associated <i>GPIOOn_INT_MOD</i> bit is set to edge triggered. The associated polarity (<i>GPIOOn_INT_POL</i>) setting has no effect when this bit is set. 0: Disable 1: Enable | |

Table 5-34: GPIO Port n Pad Configuration 1 Register

| GPIO Port n Pad Configuration 1 | | | GPIOOn_PAD_CFG1 | | [0x0060] |
|---------------------------------|--------|--------|-----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | config | R/W | 0 | GPIO Pad Configuration 1 Input mode configuration for the associated GPIO pin. Input mode selection and the selection of a weak or strong pullup or weak or strong pulldown resistor are described in <i>Table 5-3</i> . | |

Table 5-35: GPIO Port n Pad Configuration 2 Register

| GPIO Port n Pad Configuration 2 | | | GPIOOn_PAD_CFG2 | | [0x0064] |
|---------------------------------|--------|--------|-----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | config | R/W | 0 | GPIO Pad Configuration 2 Input mode configuration for the associated GPIO pin. Input mode selection and the selection of a weak or strong pullup or weak or strong pulldown resistor are described in Table 5-3: MAX78000 Input Mode Configuration . | |

Table 5-36: GPIO Port n Configuration Enable Bit 1 Register

| GPIO Port n Configuration Enable Bit 1 | | | GPIOOn_EN1 | | [0x0068] |
|--|--------|--------|------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | config | R/W | 0 | GPIO Configuration Enable Bit 1 These bits, in conjunction with bits in Table 5-2: MAX78000 GPIO Pin Function Configuration configure the corresponding device pin for digital I/O or an alternate function mode. This field can be modified directly by writing to this register or indirectly through GPIOOn_EN1_SET or GPIOOn_EN1_CLR . Table 5-6: MAX78000 GPIO Alternate Function Configuration depicts a detailed example of how each of these bits applies to each of the GPIO device pins <i>Note: Some GPIO are not implemented in all devices. The bits associated with unimplemented GPIO should not be changed from their default value.</i> <i>Note: This register setting does not affect input and interrupt functionality of the associated pin.</i> | |

Table 5-37: GPIO Port n Configuration Enable Atomic Set Bit 1 Register

| GPIO Port n Configuration Enable Atomic Set Bit 1 | | | GPIOOn_EN1_SET | | [0x006C] |
|---|-------|--------|----------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Configuration Enable Atomic Set Bit 1 Writing 1 to one or more bits sets the corresponding bits in the GPIOOn_EN1 register. 0: No effect. 1: Corresponding bits in GPIOOn_EN1 register set to 1. | |

Table 5-38: GPIO Port n Configuration Enable Atomic Clear Bit 1 Register

| GPIO Port n Configuration Enable Atomic Clear Bit 1 | | | GPIOOn_EN1_CLR | | [0x0070] |
|---|-------|--------|----------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Configuration Enable Atomic Clear Bit 1 Writing 1 to one or more bits clears the corresponding bits in the GPIOOn_EN1 register. 0: No effect. 1: Corresponding bits in GPIOOn_EN1 register cleared to 0. | |

Table 5-39: GPIO Port n Configuration Enable Bit 2 Register

| GPIO Port n Configuration Enable Bit 2 | | | GPIOn_EN2 | | [0x0074] |
|--|--------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | config | R/W | 0 | GPIO Configuration Enable Bit 2 These bits, in conjunction with bits in Table 5-2: MAX78000 GPIO Pin Function Configuration configure the corresponding device pin for digital I/O or an alternate function mode. This field can be modified directly by writing to this register or indirectly through GPIOn_EN2_SET or GPIOn_EN2_CLR . <i>Note: Some GPIO are not implemented in all devices. The bits associated with unimplemented GPIO should not be changed from their default value.</i> <i>Note: This register setting does not affect input and interrupt functionality of the associated pin.</i> | |

Table 5-40: GPIO Port n Configuration Enable Atomic Set Bit 2 Register

| GPIO Port n Configuration Enable Atomic Set Bit 2 | | | | GPIOn_EN2_SET | [0x0078] |
|---|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Alternate Function Select Atomic Set Bit 2 Writing 1 to one or more bits sets the corresponding bits in the GPIOn_EN2 register. 0: No effect. 1: Corresponding bits in GPIOn_EN2 register set to 1. | |

Table 5-41: GPIO Port n Configuration Enable Atomic Clear Bit 2 Register

| GPIO Port n Configuration Enable Atomic Clear Bit 2 | | | | GPIOn_EN2_CLR | [0x007C] |
|---|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | - | R/W1 | 0 | GPIO Alternate Function Select Atomic Clear Bit 2 Writing 1 to one or more bits clears the corresponding bits in the GPIOn_EN2 register. 0: No effect. 1: Corresponding bits in GPIOn_EN2 register cleared to 0. | |

Table 5-42: GPIO Port n Output Drive Strength Bit 0 Register

| GPIO Port n Output Drive Strength Bit 0 | | | | GPIOn_DS | [0x00B0] |
|---|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | str | R/W | 0 | GPIO Output Drive Strength Selection 0 See Table 5-4: MAX78000 Output Mode Configuration for details on how to set the GPIO output drive strength and other electrical characteristics. | |

Table 5-43: GPIO Port n Output Drive Strength Bit 1 Register

| GPIO Port n Output Drive Strength Bit 1 | | | | GPIOn_DS1 | [0x00B4] |
|---|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | str | R/W | 0 | GPIO Output Drive Strength Selection 1 See Table 5-4: MAX78000 Output Mode Configuration for details on how to set the GPIO output drive strength and other electrical characteristics. | |

Table 5-44: GPIO Port n Pulldown/Pullup Strength Select Register

| GPIO Port n Pulldown/Pullup Strength Select | | | | GPIOn_PS | [0x00B8] |
|---|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | str | R/W | 0 | GPIO Pulldown/Pullup Strength Select Selects the strength of the pullup or pulldown resistor for a pin configured for input mode. 0: Weak pulldown/pullup resistor for input pin. 1: Strong pulldown/pullup resistor for input pin. <i>Note: Refer to the datasheet for specific electrical characteristics of the Pulldown/Pullup resistances.</i> | |

Table 5-45: GPIO Port n Voltage Select Register

| GPIO Port n Voltage Select | | | | GPIOn_VSEL | [0x00C0] |
|----------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | vlvl | R/W | 0 | GPIO Supply Voltage Select Selects the voltage rail used for the pin. 0: VDDIO voltage selection 1: VDDIOH voltage selection | |

6. Flash Controller (FLC)

The MAX78000 Flash Controller manages read, write, and erase accesses to the internal flash. It provides the following features:

- Up to 512KB total internal flash memory
- 64 pages
- 8,192 bytes per page
- 2,048 words by 128 bits per page
- 128-bit data reads and writes
- Page erase and mass erase support
- Write protection

6.1 Instances

The device provides one instance of the FLC.

The 512KB of internal flash memory is programmable via serial wire debug interface (in-system) or directly with user application code (in-application).

The flash is organized as an array of 2,048 words by 128 bits, or 8,192 bytes per page. *Table 6-1, below*, shows the page start address and page end address of the internal flash memory.

Table 6-1. MAX78000 Internal Flash Memory Organization

| Instance Number | Page Number | Size (per page) | Start Address | End Address |
|-----------------|-------------|-----------------|---------------|-------------|
| FLC0 | 0 | 8,192 Bytes | 0x1000 0000 | 0x1000 1FFF |
| | 1 | 8,192 Bytes | 0x1000 2000 | 0x1000 3FFF |
| | 2 | 8,192 Bytes | 0x1000 4000 | 0x1000 5FFF |
| | 3 | 8,192 Bytes | 0x1000 6000 | 0x1000 7FFF |
| | ... | ... | ... | ... |
| | 62 | . | 0x1007 C000 | 0x1007 DFFF |
| | 63 | . | 0x1007 E000 | 0x1007 FFFF |

6.2 Usage

The Flash Controller manages write and erase operations for internal flash memory and provides a lock mechanism to prevent unintentional writes to the internal flash. In-application and in-system programming, page erase and mass erase operations are supported.

6.2.1 Clock Configuration

The FLC requires a 1MHz internal clock. See *Oscillator Sources* for details. Use the FLC clock divisor to generate $f_{FLCn_CLK} = 1MHz$, as shown in *Equation 6-1*. If using the IPO as the system clock, the *FLCn_CLKDIV.clkdiv* should be set to 100 (0x64).

Equation 6-1: FLC Clock Frequency

$$f_{FLCn_CLK} = \frac{f_{SYS_CLK}}{\text{FLCn_CLKDIV}.clkdiv} = 1MHz$$

6.2.2 Lock Protection

A locking mechanism prevents accidental memory writes and erases. All writes and erase operations require the *FLCn_CN.unlock* field be set to 2 prior to starting the operation. Writing any other value to this field, *FLCn_CN.unlock*, results in:

1. The flash instance remaining locked,
or,
2. The flash instance becoming locked from the unlocked state.

Note: If a write, page erase or mass erase operation is started and the unlock code was not set to 2, the flash controller hardware sets the access fail flag, [FLCn_INTR.af](#), to indicate an access violation occurred.

6.2.3 Flash Write Width

The FLC supports write widths of 128-bits only. The target address bits [FLCn_ADDR\[3:0\]](#) are ignored resulting in 128-bit alignment.

Table 6-2: Valid Addresses Flash Writes

| | FLCn_ADDR[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---------------|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 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 |
| 128-bit Write | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 | 0 | 0 | 0 | |

6.2.4 Flash Write

Writes to a flash location are only successful if the targeted location is already in its erased state. Perform the following steps to write to a flash memory instance:

1. If desired, enable flash controller interrupts by setting the [FLCn_INTR.afie](#) and [FLCn_INTR.doneie](#) bits.
2. Read the [FLCn_CN.pend](#) bit until it returns 0.
3. Configure [FLCn_CLKDIV.clkdiv](#) to match the selected SYS_CLK frequency.
4. Set the [FLCn_ADDR](#) register to a valid target address. Reference [Table 6-2](#).
5. Set [FLCn_DATA3](#), [FLCn_DATA2](#), [FLCn_DATA1](#), and [FLCn_DATA0](#) to the data to write. [FLCn_DATA3](#) is the most significant word and [FLCn_DATA0](#) is the least significant word. Each word of the data to write follows the little-endian format where the least significant byte of the word is stored at the lowest-numbered byte and the most significant byte is stored at the highest-numbered byte.
6. Set [FLCn_CN.unlock](#) to 2 to unlock the flash instance.
7. Set [FLCn_CN.wr](#) to 1. This field is automatically cleared by the FLC when the write operation is finished.
8. [FLCn_INTR.done](#) is set by hardware when the write completes and if an error occurred, the [FLCn_INTR.af](#) flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.
9. Set [FLCn_CN.unlock](#) to any value other than 2 to re-lock the flash instance.

Note: Code execution can occur within the same flash instance as targeted programming.

6.2.5 Flash Write

Writes to a flash location are only successful if the targeted location is already in its erased state. Perform the following steps to write to a flash memory instance:

1. If desired, enable flash controller interrupts by setting the `FLCn_INTR.afie` and `FLCn_INTR.doneie` bits.
2. Read the `FLCn_CN.pend` bit until it returns 0.
3. Configure `FLCn_CLKDIV.clkdiv` to match the SYS_CLK frequency.
4. Set the `FLCn_ADDR` register to a valid target address. Reference [Table 6-2](#).
5. Set `FLCn_DATA3`, `FLCn_DATA2`, `FLCn_DATA1`, and `FLCn_DATA0` to the data to write. `FLCn_DATA3` is the most significant word and `FLCn_DATA0` is the least significant word. Each word of the data to write follows the little-endian format where the least significant byte of the word is stored at the lowest-numbered byte and the most significant byte is stored at the highest-numbered byte.
6. Set `FLCn_CN.unlock` to 2 to unlock the flash instance.
7. Set `FLCn_CN.wr` to 1. This field is automatically cleared by the FLC when the write operation is finished.
8. `FLCn_INTR.done` is set by hardware when the write completes and if an error occurred, the `FLCn_INTR.af` flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.
9. Set `FLCn_CN.unlock` to any value other than 2 to re-lock the flash instance.

Note: Code execution can occur within the same flash instance as targeted programming.

6.2.6 Page Erase

CAUTION: Care must be taken to not erase the page from which application code is currently executing.

Perform the following to erase a page of a flash memory instance:

1. If desired, enable flash controller interrupts by setting the `FLCn_INTR.afie` and `FLCn_INTR.doneie` bits.
2. Read the `FLCn_CN.pend` bit until it returns 0.
3. Configure `FLCn_CLKDIV.clkdiv` to match the SYS_CLK frequency.
4. Set the `FLCn_ADDR` register to an address within the target page to be erased. `FLCn_ADDR[12:0]` are ignored by the FLC to ensure the address is page aligned.
5. Set `FLCn_CN.unlock` to 2 to unlock the flash instance.
6. Set `FLCn_CN.erase_code` to 0x55 for page erase.
7. Set `FLCn_CN.pge` to 1 to start the page erase operation.
8. The `FLCn_CN.pend` bit is set by the flash controller while the page erase is in progress and the `FLCn_CN.pge` and `FLCn_CN.pend` are cleared by the flash controller when the page erase is complete.
9. `FLCn_INTR.done` is set by hardware when the page erase completes and if an error occurred, the `FLCn_INTR.af` flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.
10. Set `FLCn_CN.unlock` to any value other than 2 to re-lock the flash instance.

6.2.7 Mass Erase

CAUTION: Care must be taken to not erase the flash from which application code is currently executing.

Mass erase clears the internal flash memory on an instance basis. Perform the following steps to mass erase a single flash memory instance:

1. Read the *FLCn_CN.pend* bit until it returns 0.
2. Configure *FLCn_CLKDIV.clkdiv* to match the SYS_CLK frequency.
3. Set *FLCn_CN.unlock* to 2 to unlock the internal flash.
4. Set *FLCn_CN.erase_code* to 0xAA for mass erase.
5. Set *FLCn_CN.me* to 1 to start the mass erase operation.
6. The *FLCn_CN.pend* bit is set by the flash controller while the mass erase is in progress and the *FLCn_CN.me* and *FLCn_CN.pend* are cleared by the flash controller when the mass erase is complete.
7. *FLCn_INTR.done* is set by the flash controller when the mass erase completes and if an error occurred, the *FLCn_INTR.af* flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.
8. Set *FLCn_CN.unlock* to any value other than 2 to re-lock the flash instance.

6.3 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 6-3: Flash Controller Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 6-3: Flash Controller Register Summary

| Offset | Register Name | Access | Description |
|----------|--------------------|--------|---|
| [0x0000] | <i>FLCn_ADDR</i> | R/W | Flash Controller Address Pointer Register |
| [0x0004] | <i>FLCn_CLKDIV</i> | R/W | Flash Controller Clock Divisor Register |
| [0x0008] | <i>FLCn_CN</i> | R/W | Flash Controller Control Register |
| [0x0024] | <i>FLCn_INTR</i> | R/W | Flash Controller Interrupt Register |
| [0x0030] | <i>FLCn_DATA0</i> | R/W | Flash Controller Data Register 0 |
| [0x0034] | <i>FLCn_DATA1</i> | R/W | Flash Controller Data Register 1 |
| [0x0038] | <i>FLCn_DATA2</i> | R/W | Flash Controller Data Register 2 |
| [0x0040] | <i>FLCn_ACNTL</i> | R/W | Flash Controller Access Control Register |
| [0x0080] | <i>FLCn_WELR0</i> | R/W | Flash Write/Erase Lock 0 Register |
| [0x0088] | <i>FLCn_WELR1</i> | R/W | Flash Write/Erase Lock 1 Register |
| [0x0090] | <i>FLCn_RLRO</i> | R/W | Flash Read Lock 0 Register |
| [0x0098] | <i>FLCn_RLR1</i> | R/W | Flash Read Lock 1 Register |

6.3.1 Register Details

Table 6-4: Flash Controller Address Pointer Register

| Flash Controller Address Pointer | | | FLCn_ADDR | | [0x0000] |
|----------------------------------|------|--------|-------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | addr | R/W | 0x1000 0000 | Flash Address This field contains the target address for a write operation. A valid internal flash memory address is required for all write operations. | |

Table 6-5: Flash Controller Clock Divisor Register

| Flash Controller Clock Divisor | | | FLCn_CLKDIV | | [0x0004] |
|--------------------------------|--------|--------|-------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:8 | - | RO | - | Reserved Do not modify | |
| 7:0 | clkdiv | R/W | 0x64 | Flash Controller Clock Divisor The APB clock is divided by the value in this field to generate the FLCn peripheral clock, f_{FLCn_CLK} . The FLCn peripheral clock must equal 1MHz. The default on POR, System Reset and Watchdog reset is 100, resulting in $f_{FLCn_CLK} = 1\text{MHz}$ when IPO is the system oscillator. The FLCn peripheral clock is only used during erase and program functions and not during read functions. Refer to Clock Configuration for additional details. | |

Table 6-6: Flash Controller Control Register

| Flash Controller Control | | | FLCn_CN | | [0x0008] |
|--------------------------|--------|--------|---------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:28 | unlock | R/W | 0 | Flash Unlock Write the unlock code, 2, prior to any flash write or erase operation to unlock the Flash. Writing any other value to this field locks the internal flash. 2: Flash unlock code | |
| 27:26 | - | RO | - | Reserved Do not modify | |
| 25 | lve | R/W | 0 | Low Voltage Enable Set this field to 1 to enable low voltage operation for the flash memory. 0: Low voltage operation disabled (Default). 1: Low voltage operation enabled. | |
| 24 | pend | RO | 0 | Flash Busy Flag When this field is set, writes to all flash registers except the FLCn_INTR register are ignored by the Flash Controller. This bit will be cleared by hardware once the flash becomes accessible. <i>Note: If the Flash Controller is busy (FLCn_CN.pend = 1), reads, writes and erase operations are not allowed and result in an access failure (FLCn_INTR.af = 1).</i> 0: Flash idle 1: Flash busy | |
| 23:16 | - | RO | 0 | Reserved Do not modify | |

| Flash Controller Control | | | | FLCn_CN | [0x0008] |
|--------------------------|------------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 15:8 | erase_code | R/W | 0 | Erase Code Prior to an erase operation this field must be set to 0x55 for a page erase or 0xAA for a mass erase. The flash must be unlocked prior to setting the erase code. This field is automatically cleared after the erase operation is complete. 0x00: Erase disabled. 0x55: Page erase code. 0xAA: Enable mass erase. | |
| 7:3 | - | RO | 0 | Reserved Do not modify | |
| 2 | pge | R/W1 | 0 | Page Erase Write a 1 to this field to initiate a page erase at the address in <i>FLCn_ADDR.addr</i> . The flash must be unlocked prior to attempting a page erase, see <i>FLCn_CN.unlock</i> for details. The Flash Controller hardware clears this bit when a page erase operation is complete. 0: No page erase operation in process or page erase is complete. 1: Write a 1 to initiate a page erase. If this field reads 1, a page erase operation is in progress. | |
| 1 | me | R/W1 | 0 | Mass Erase Write a 1 to this field to initiate a mass erase of the internal flash memory. The flash must be unlocked prior to attempting a mass erase, see <i>FLCn_CN.unlock</i> for details. The Flash Controller hardware clears this bit when the mass erase operation completes. 0: No operation 1: Initiate mass erase | |
| 0 | wr | R/W10 | 0 | Write If this field reads 0, no write operation is pending for the flash. To initiate a write operation, set this bit to 1 and the Flash Controller will write to the address set in the <i>FLCn_ADDR</i> register. 0: No write operation in process or write operation complete. 1: Write 1 to initiate a write operation. If this field reads 1, a write operation is in progress. <i>Note: This field is protected and cannot be set to 0 by application code.</i> | |

Table 6-7: Flash Controller Interrupt Register

| Flash Controller Interrupt | | | | FLCn_INTR | [0x0024] |
|----------------------------|------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:10 | - | RO | 0 | Reserved Do not modify | |
| 9 | afie | R/W | 0 | Flash Access Fail Interrupt Enable Set this bit to 1 to enable interrupts on flash access failures. 0: Disabled 1: Enabled | |

| Flash Controller Interrupt | | | | FLCn_INTR | [0x0024] |
|----------------------------|--------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 8 | doneie | R/W | 0 | Flash Operation Complete Interrupt Enable Set this bit to 1 to enable interrupts on flash operations complete. 0: Disabled 1: Enabled | |
| 7:2 | - | RO | 0 | Reserved Do not modify | |
| 1 | af | R/WOC | 0 | Flash Access Fail Interrupt Flag This bit is set when an attempt is made to write or erase the flash while the flash is busy or locked. Only hardware can set this bit to 1. Writing a 1 to this bit has no effect. This bit is cleared by writing a 0. 0: No access failure has occurred. 1: Access failure occurred. | |
| 0 | done | R/WOC | 0 | Flash Operation Complete Interrupt Flag This flag is automatically set by hardware after a flash write or erase operation completes. 0: Operation not complete or not in process. 1: Flash operation complete. | |

Table 6-8: Flash Controller Data 0 Register

| Flash Controller Data 0 | | | | FLCn_DATA0 | [0x0030] |
|-------------------------|------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | Flash Data 0 Flash data for bits 31:0. | |

Table 6-9: Flash Controller Data Register 1

| Flash Controller Data 1 | | | | FLCn_DATA1 | [0x0034] |
|-------------------------|------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | Flash Data 1 Flash data for bits 63:32 | |

Table 6-10: Flash Controller Data Register 2

| Flash Controller Data 2 | | | | FLCn_DATA2 | [0x0038] |
|-------------------------|------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | Flash Data 2 Flash data for bits 95:64 | |

Table 6-11: Flash Controller Data Register 3

| Flash Controller Data 3 | | | | FLCn_DATA3 | [0x003C] |
|-------------------------|------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | Flash Data 3 Flash data for bits 127:96. | |

Table 6-12: Flash Controller Access Control Register

| Flash Controller Access Control | | | FLCn_ACNTL | | [0x0040] |
|---------------------------------|-------|--------|------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | actrl | R/W | 0 | Access Control When this register is written with the access control sequence, the information block can be accessed. See Information Block Flash Memory for details. | |

Table 6-13: Flash Write/Lock 0 Register

| Flash Write/Lock 0 | | | FLCn_WELR0 | | [0x0080] |
|--------------------|------|--------|------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | - | R/W1C | 0xFFFF | Flash Write/Lock Bit Each bit in this register maps to a page of the internal flash. FLCn_WELR0 [0] maps to page 0 of the flash and FLCn_WELR0 [31] maps to page 31. Each flash page is 8,192 bytes. Write a 1 to a bit position in this register and the corresponding page of flash is immediately locked. The page protection can only be unlocked by an external reset or a POR. 0: The corresponding page of flash is <i>not</i> write protected 1: The corresponding page of flash is write protected | |

Table 6-14: Flash Write/Lock 1 Register

| Flash Write/Lock 1 | | | FLCn_WELR1 | | [0x0088] |
|--------------------|------|--------|------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | - | R/W1C | 0xFFFF | Flash Write/Lock Bit Each bit in this register maps to a page of the internal flash. FLCn_WELR1 [0] maps to page 32 of the flash and FLCn_WELR1 [31] maps to page 63 of flash. Each flash page is 8,192 bytes. Write a 1 to a bit position in this register and the corresponding page of flash is immediately locked. The page protection can only be unlocked by an external reset or a POR. 0: The corresponding flash page is <i>not</i> write protected 1: The corresponding flash page is write protected | |

Table 6-15: Flash Read Lock 0 Register

| Flash Read Lock 0 | | | FLCn_RLR0 | | [0x0090] |
|-------------------|------|--------|-----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | - | R/W1C | 0xFFFF | Read Lock Bit Each bit in this register maps to a page of the internal flash. FLCn_RLR0 [0] maps to page 32 of the flash and FLCn_RLR0 [31] maps to page 63 of flash. Each flash page is 8,192 bytes. Write a 1 to a bit position in this register and the corresponding page of flash is immediately read protected. The page's read protection can only be unlocked by an external reset or a POR. 0: The corresponding flash page is <i>not</i> read protected 1: The corresponding flash page is read protected | |

Table 6-16: Flash Read Lock 1 Register

| Flash Read Lock 1 | | | FLCn_RLR1 | | [0x0098] |
|-------------------|------|--------|-----------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | - | R/W1C | 0xFFFF | <p>Read Lock Bit</p> <p>Each bit in this register maps to a page of the internal flash. <i>FLCn_RLR1</i>[0] maps to page 32 of the flash and <i>FLCn_RLR1</i>[31] maps to page 63 of flash. Each flash page is 8,192 bytes. Write a 1 to a bit position in this register and the corresponding page of flash is immediately read protected. The page's read protection can only be unlocked by an external reset or a POR.</p> <p>0: The corresponding flash page is <i>not</i> read protected 1: The corresponding flash page is read protected</p> | |

7. Debug Access Port (DAP)

Some device versions might provide an Arm debug access port (DAP) which supports debugging during application development. Refer to the ordering information in the device data sheet to determine if a specific part number supports a customer-accessible DAP. Parts with a customer-accessible DAP should only be used for development and never used in a final customer product.

GCR_SYSST.icelock = 0 if the device provides a customer-accessible DAP.

7.1 Instances

The DAP interface communicates through the serial wire debug (SWD) and/or JTAG interface signals shown in *Table 7-1: MAX78000 DAP Instances*.

Table 7-1: MAX78000 DAP Instances

| Instance | Pin | Alternate Function | SWD Signal | JTAG Signal | Pullup |
|----------|-------|--------------------|------------|-------------|---------------------------|
| DAP0 | P0.28 | AF1 | SWDIO | N/A | 10kΩ to V _{DDIO} |
| | P0.29 | AF1 | SWCLK | N/A | |

7.2 Access Control

7.2.1 Factory Disabled DAP

Device versions that do not provide a DAP interface will have *GCR_SYSST.icelock* = 1 set at the factory, permanently disabling the DAP interface. No software action is needed to secure these devices.

7.2.2 Software Accessible DAP

Device versions that provide a DAP (*GCR_SYSST.icelock* = 0) always have their interface(s) enabled and running unless the software explicitly sets *GCR_SYSCTRL.sw_dis* field to 1. The read-only field *GCR_SYSST.icelock* is cleared to 0 and the software has read and write access to the *GCR_SYSCTRL.sw_dis* field. The *GCR_SYSCTRL.sw_dis* field resets to 0 after every POR to allow access to the DAP during development.

Software can disable the DAP by setting the *GCR_SYSCTRL.sw_dis* field to 1. The only practical application for disabling the DAP is to release the interface pins to operate as standard GPIO or in one of the supported alternate function modes, in a development environment. Customers can use device versions with the DAP enabled for development but should only use device versions with the factory disabled DAP in a final product.

7.3 Pin Configuration

Instances of SWD or JTAG signals in GPIO and Alternate Function matrices are for determining which GPIO pins are associated with a signal. It is not necessary to configure a pin for an alternate function to use the DAP following a POR.

By default, the pin associated with the bidirectional SWDIO/TMS signal is configured as a GPIO high-impedance input after any POR. While the DAP is in use, a pullup resistor should be connected to the SWDIO/TMS pin as shown in *Table 7-1: MAX78000 DAP Instances*. The pullup ensures the signal is in a known state when control of the SWDIO/TMS pin is transferred between the host and target. The pullup resistor should be removed if the associated pin is used as a GPIO to avoid unnecessary current consumption.

8. Semaphores

The Semaphore peripheral allows multiple cores in a system to cooperate when accessing shared resources. The peripheral contains eight semaphore registers that can be atomically set and cleared. Reading the status field of a Semaphore register returns the current state of the status field, and if the field is 0 automatically sets the status to 1. The Semaphore Status register reflects the state of each of the Semaphore Register's status. The Status register enables checking each of the Semaphore's states but is read only so it is not guaranteed that the Semaphore Status fields will not change after checking the Status register's value.

It is left to the discretion of the software architect to decide how and when the semaphores are used and how they are allocated. Existing hardware does not have to be modified for this type of cooperative sharing, and the use of semaphores is exclusively within the software domain.

The Semaphore peripheral includes two general purpose mailbox registers which enable communication between the RV32 and CM4 cores. Additionally, the either core can generate a Semaphore IRQ for either the CM4 or the RV32 providing immediate notification of communication via the mailbox registers.

8.1 Instances

There is one instance of the semaphore peripheral, shown in *Table 8-1: MAX78000 Semaphore Instances*.

Table 8-1: MAX78000 Semaphore Instances

| Instance | Number of Semaphore Registers |
|----------|-------------------------------|
| SEMA | 8 |

8.2 Multiprocessor Communications

The Semaphore includes support for multicore communications via two mailbox registers and provides the ability to generate an RV32 Semaphore Interrupt and a CM4 Semaphore Interrupt.

The mailbox registers, *SEMA_MAIL0* and *SEMA_MAIL1*, are general purpose 32-bit registers. The CM4 and RV32 have read and write access to both of these registers. Application firmware should manage how these registers are used to prevent collisions occurring if both cores attempt to modify the registers at the same time.

8.2.1 CM4 Semaphore Interrupt Generation

The *SEMA_IRQ0* register provides the ability to generate a CM4 Semaphore Interrupt. Setting the *SEMA_IRQ0.cm4_irq* bit to 1 and then setting the *SEMA_IRQ0.en* bit to 1 generates a CM4 Semaphore IRQ. The CM4 IRQ handler should write the *SEMA_IRQ0.en* and/or the *SEMA_IRQ0.cm4_irq* field(s) to 0 to clear the interrupt condition.

8.2.2 RV32 Semaphore Interrupt Generation

The *SEMA_IRQ1* register provides the ability to generate a RV32 Semaphore Interrupt. Setting the *SEMA_IRQ1.rv32_irq* bit to 1 and then setting the *SEMA_IRQ1.en* bit to 1 generates a RV32 Semaphore IRQ. The RV32 IRQ handler should write the *SEMA_IRQ1.en* and/or the *SEMA_IRQ1.rv32_irq* field(s) to 0 to clear the interrupt condition.

8.3 Registers

See *Table 2-4: APB Peripheral Base Address Map* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in *Table 8-2: Semaphore Register Summary*. Register names for a specific instance are defined by replacing "n" with the instance number. As an

example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 8-2: Semaphore Register Summary

| Offset | Register | Name |
|----------|----------------------------------|------------------------------|
| [0x0000] | SEMA_SEMIPHORE_0 | Semaphore 0 Register |
| [0x0004] | SEMA_SEMIPHORE1 | Semaphore 1 Register |
| [0x0008] | SEMA_SEMIPHORE2 | Semaphore 2 Register |
| [0x000C] | SEMA_SEMIPHORE3 | Semaphore 3 Register |
| [0x0010] | SEMA_SEMIPHORE4 | Semaphore 4 Register |
| [0x0014] | SEMA_SEMIPHORE5 | Semaphore 5 Register |
| [0x0018] | SEMA_SEMIPHORE6 | Semaphore 6 Register |
| [0x0020] | SEMA_SEMIPHORE7 | Semaphore 7 Register |
| [0x0040] | SEMA_IRQ0 | Semaphore IRQ 0 Register |
| [0x0044] | SEMA_MAIL0 | Semaphore Mailbox 0 Register |
| [0x0048] | SEMA_IRQ1 | Semaphore IRQ 1 Register |
| [0x004C] | SEMA_MAIL1 | Semaphore Mailbox 1 Register |
| [0x0100] | SEMA_STATUS | Semaphore Status Register |

8.3.1 Register Details

Table 8-3: Semaphore 0 Register

| Semaphore 0 | | | | SEMA_SEMIPHORE_0 | [0x0000] |
|-------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | | 0 | Reserved Do Not Modify. | |
| 0 | status | * | 0 | Semaphore Status Reading this field returns its current value and if 0, automatically sets the field to 1. Write 0 to clear this field. Modifications to this field are mirrored in the SEMA_STATUS.status0 field. 0: Semaphore is available 1: Semaphore is taken | |

Table 8-4: Semaphore 1 Register

| Semaphore 1 | | | | SEMA_SEMIPHORE1 | [0x0004] |
|-------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | | 0 | Reserved Do Not Modify. | |
| 0 | status | * | 0 | Semaphore Status Reading this field returns its current value and if 0, automatically sets the field to 1. Write 0 to clear this field. Modifications to this field are mirrored in the SEMA_STATUS.status1 field. 0: Semaphore is available 1: Semaphore is taken | |

Table 8-5: Semaphore 2 Register

| Semaphore 2 | | | | SEMA_SEMIPHORE2 | [0x0008] |
|-------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | | 0 | Reserved Do Not Modify. | |
| 0 | status | * | 0 | Semaphore Status Reading this field returns its current value and if 0, automatically sets the field to 1. Write 0 to clear this field. Modifications to this field are mirrored in the SEMA_STATUS.status2 field. 0: Semaphore is available 1: Semaphore is taken | |

Table 8-6: Semaphore 3 Register

| Semaphore 3 | | | | SEMA_SEMIPHORE3 | [0x000C] |
|-------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | | 0 | Reserved Do Not Modify. | |
| 0 | status | * | 0 | Semaphore Status Reading this field returns its current value and if 0, automatically sets the field to 1. Write 0 to clear this field. Modifications to this field are mirrored in the SEMA_STATUS.status3 field. 0: Semaphore is available 1: Semaphore is taken | |

Table 8-7: Semaphore 4 Register

| Semaphore 4 | | | | SEMA_SEMIPHORE4 | [0x0010] |
|-------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | | 0 | Reserved Do Not Modify. | |
| 0 | status | * | 0 | Semaphore Status Reading this field returns its current value and if 0, automatically sets the field to 1. Write 0 to clear this field. Modifications to this field are mirrored in the SEMA_STATUS.status4 field. 0: Semaphore is available 1: Semaphore is taken | |

Table 8-8: Semaphore 5 Register

| Semaphore 5 | | | | SEMA_SEMIPHORE5 | [0x0014] |
|-------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | | 0 | Reserved Do Not Modify. | |
| 0 | status | * | 0 | Semaphore Status Reading this field returns its current value and if 0, automatically sets the field to 1. Write 0 to clear this field. Modifications to this field are mirrored in the SEMA_STATUS.status5 field. 0: Semaphore is available 1: Semaphore is taken | |

Table 8-9: Semaphore 6 Register

| Semaphore 6 | | | | SEMA_SEMIPHORE6 | [0x0018] |
|-------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | | 0 | Reserved Do Not Modify. | |
| 0 | status | * | 0 | Semaphore Status Reading this field returns its current value and if 0, automatically sets the field to 1. Write 0 to clear this field. Modifications to this field are mirrored in the SEMA_STATUS.status6 field. 0: Semaphore is available 1: Semaphore is taken | |

Table 8-10: Semaphore 7 Register

| Semaphore 7 | | | | SEMA_SEMIPHORE7 | [0x001C] |
|-------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:1 | - | | 0 | Reserved Do Not Modify. | |
| 0 | status | * | 0 | Semaphore Status Reading this field returns its current value and if 0, automatically sets the field to 1. Write 0 to clear this field. Modifications to this field are mirrored in the SEMA_STATUS.status7 field. 0: Semaphore is available 1: Semaphore is taken | |

Table 8-11: Semaphore IRQ 0 Register

| Semaphore IRQ 0 | | | | SEMA_IRQ0 | [0x0040] |
|-----------------|---------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |
| 16 | cm4_irq | R/W | 0 | CM4 IRQ The RV32 can use this bit to communicate with the CM4 via the Semaphore IRQ. The RV32 generates a Semaphore IRQ for the CM4 by setting this field to 1 and also setting the SEMA_IRQ0.en bit to 1. | |
| 15:1 | - | R/W | 0 | Reserved Do not modify | |
| 0 | en | R/W | 0 | IRQ Enable Set this field to enable IRQ generation on Semaphore events. 0: IRQ disabled 1: IRQ enabled | |

Table 8-12: Semaphore Mailbox 0 Register

| Semaphore Mailbox 0 | | | | SEMA_MAIL0 | [0x0044] |
|---------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | Data This register is readable and writable by both the CM4 and RV32 cores allowing communication between the two cores. In conjunction with the SEMA_IRQ0 register, the RV32 can write data to this register and then notify the CM4 by generating a Semaphore IRQ. Alternately, the CM4 can write to this register and then notify the RV32 using the SEMA_IRQ1 register to generate an RV32 Semaphore IRQ event. <i>Note: The management of the SEMA_MAIL0 and SEMA_MAIL1 registers is left to application firmware. It is recommended that one mailbox is used for communication from the CM4 to the RV32 and the other mailbox register is used for communication from the RV32 to the CM4. However, there are no hardware read/write restrictions on the mailbox registers.</i> | |

Table 8-13: Semaphore IRQ 1 Register

| Semaphore IRQ 1 | | | | SEMA_IRQ1 | [0x0048] |
|-----------------|----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |
| 16 | rv32_irq | R/W | 0 | RV32 IRQ The CM4 can use this bit to communicate with the RV32 via the Semaphore IRQ. The CM4 generates a Semaphore IRQ for the RV32 by setting this field to 1 and also setting the SEMA_IRQ1.en bit to 1. 0: RV32 IRQ event not active or received by RV32. 1: RV32 IRQ event is generated when the SEMA_IRQ1.en bit is also set to 1. | |
| 15:1 | - | R/W | 0 | Reserved Do not modify | |
| 0 | en | R/W | 0 | IRQ Enable Set this field to generate a RV32 Semaphore IRQ when the SEMA_IRQ1.rv32_irq is also set to 1. The RV32 should write this bit to 0 when a Semaphore IRQ is generated to prevent repeat IRQ generation. 0: IRQ disabled 1: IRQ enabled | |

Table 8-14: Semaphore Mailbox 1 Register

| Semaphore Mailbox 1 | | | | SEMA_MAIL1 | [0x004C] |
|---------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | Data This register is readable and writable by both the CM4 and RV32 cores allowing communication between the two cores. In conjunction with the SEMA_IRQ0 register, the RV32 can write data to this register and then notify the CM4 by generating a Semaphore IRQ. Alternately, the CM4 can write to this register and then notify the RV32 using the SEMA_IRQ1 register to generate an RV32 Semaphore IRQ event. <i>Note: The management of the SEMA_MAIL0 and SEMA_MAIL1 registers is left to application firmware. It is recommended that one mailbox is used for communication from the CM4 to the RV32 and the other mailbox register is used for communication from the RV32 to the CM4. However, there are no hardware read/write restrictions on the mailbox registers.</i> | |

Table 8-15: Semaphore Status Register

| Semaphore Status | | | | SEMA_STATUS | [0x0100] |
|------------------|---------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved for Future Use Do Not Modify. | |
| 7 | status7 | RO | 0 | Semaphore 7 Status This field mirrors the Semaphore 7 status field. Reads from this field do not affect the corresponding Semaphore's status field. 0: <i>SEMA_SEMIPHORE7.status</i> is 0 1: <i>SEMA_SEMIPHORE7.status</i> is 1 | |
| 6 | status6 | RO | 0 | Semaphore 6 Status This field mirrors the Semaphore 6 status field. Reads from this field do not affect the corresponding Semaphore's status field. 0: <i>SEMA_SEMIPHORE6.status</i> is 0 1: <i>SEMA_SEMIPHORE6.status</i> is 1 | |
| 5 | status5 | RO | 0 | Semaphore 5 Status This field mirrors the Semaphore 5 status field. Reads from this field do not affect the corresponding Semaphore's status field. 0: <i>SEMA_SEMIPHORE5.status</i> is 0 1: <i>SEMA_SEMIPHORE5.status</i> is 1 | |
| 4 | status4 | RO | 0 | Semaphore 4 Status This field mirrors the Semaphore 4 status field. Reads from this field do not affect the corresponding Semaphore's status field. 0: <i>SEMA_SEMIPHORE4.status</i> is 0 1: <i>SEMA_SEMIPHORE4.status</i> is 1 | |
| 3 | status3 | RO | 0 | Semaphore 3 Status This field mirrors the Semaphore 3 status field. Reads from this field do not affect the corresponding Semaphore's status field. 0: <i>SEMA_SEMIPHORE3.status</i> is 0 1: <i>SEMA_SEMIPHORE3.status</i> is 1 | |
| 2 | status2 | RO | 0 | Semaphore 2 Status This field mirrors the Semaphore 2 status field. Reads from this field do not affect the corresponding Semaphore's status field. 0: <i>SEMA_SEMIPHORE2.status</i> is 0 1: <i>SEMA_SEMIPHORE2.status</i> is 1 | |
| 1 | status1 | RO | 0 | Semaphore 1 Status This field mirrors the Semaphore 1 status field. Reads from this field do not affect the corresponding Semaphore's status field. 0: <i>SEMA_SEMIPHORE1.status</i> is 0 1: <i>SEMA_SEMIPHORE1.status</i> is 1 | |
| 0 | status0 | RO | 0 | Semaphore 0 Status This field mirrors the Semaphore 0 status field. Reads from this field do not affect the corresponding Semaphore's status field. 0: <i>SEMA_SEMIPHORE_0.status</i> is 0 1: <i>SEMA_SEMIPHORE_0.status</i> is 1 | |

9. Standard DMA (DMA)

The Standard Direct Memory Access controller (DMA) is a hardware feature that provides the ability to perform high-speed, block memory transfers of data independent of an Arm core. All DMA transactions consist of burst read from the source into the internal DMA FIFO followed by a burst write from the internal DMA FIFO to the destination.

DMA transfers are one of three types:

- From a receive FIFO to a memory address
- To a transmit FIFO from a memory address, or
- From a source memory address to a destination memory address.

The DMA supports multiple channels. Each channel provides the following features:

- Full 32-bit source and destination addresses with 24-bit (16 Mbytes) address increment capability
- Ability to chain DMA buffers when a count-to-zero (CTZ) condition occurs
- Interrupt upon CTZ
- Up to 16 Mbytes for each DMA transfer
- 8×32 byte transmit and receive FIFO
- Programmable channel timeout period
- Programmable burst size
- Programmable priority
- Abort on error

9.1 Instances

There is one instance of the DMA, generically referred to as DMAm. Each instance provides 4 channels, generically referred to as DMA_CHn. Each instance of the DMA has a set of interrupt registers common to all its channels, and a set of registers unique to each channel instance

Table 9-1: MAX78000 DMA and Channel Instances

| DMAm Instance | DMA_CHn Channel Instance |
|---------------|--------------------------|
| DMA0 | DMA_CH0 |
| | DMA_CH1 |
| | DMA_CH2 |
| | DMA_CH3 |

9.2 DMA Channel Operation (DMA_CH)

9.2.1 DMA Channel Arbitration and DMA Bursts

The DMA peripheral contains an internal arbiter that allows enabled channels to access the AHB and move data. Once a channel is programmed and enabled, it generates a request to the arbiter immediately (for memory-to-memory DMA) or whenever its associated peripheral requests DMA (for memory-to-peripheral or peripheral-to-memory DMA).

Granting is done based on priority; a higher priority request is always granted. Within a given priority level, requests are granted on a round-robin basis. The [DMA_CHn_CTRL.pri](#) field determines the DMA channel priority.

When a channel's request is granted, the channel runs a DMA transfer. The arbiter grants requests to a single channel at a time. Once the DMA transfer completes, the channel relinquishes its grant.

A DMA channel is enabled using the [DMA_CHn_CTRL.en](#) bit.

When disabling a channel, poll the *DMA_CHn_STATUS*.status bit to determine if the channel is truly disabled. In general, *DMA_CHn_STATUS*.status follows the setting of the *DMA_CHn_CTRL*.en bit. However, the *DMA_CHn_STATUS*.status bit is automatically cleared under the following conditions:

- Bus error (cleared immediately)
- CTZ when the *DMA_CHn_CTRL*.rlden = 0 (cleared at the end of the AHB R/W burst)
- *DMA_CHn_CTRL*.en bit transitions to 0 (cleared at the end of the AHB R/W burst)

Whenever *DMA_CHn_STATUS*.status transitions from 1 to 0, the corresponding *DMA_CHn_CTRL*.en bit is also cleared. If an active channel is disabled during an AHB read/write burst, the current burst will continual until completed.

Only an error condition can interrupt an ongoing data transfer.

9.2.2 DMA Source and Destination Addressing

The source and destination for DMA transfers are dictated by the request select dedicated to the peripheral instance. The *DMA_CHn_CTRL*.request field dictates the source and destination for a channel's DMA transfer as shown in *Table 9-2*. The *DMA_CHn_SRC* and *DMA_CHn_DST* registers hold the source and/or destination memory addresses, depending on the specific operation.

The *DMA_CHn_CTRL*.srcinc field is ignored when the DMA source is a peripheral memory, and the *DMA_CHn_CTRL*.dstinc field is ignored when the DMA destination is a peripheral memory.

Table 9-2: DMA Source and Destination by Peripheral

| <i>DMA_CHn_CTRL</i> .request | Peripheral | DMA Source | DMA Destination |
|------------------------------|-------------------|--------------------------------|--------------------|
| 0 | Memory-to-Memory | <i>DMA_CHn_SRC</i> | <i>DMA_CHn_DST</i> |
| 1 | SPI1 | SPI1 Receive FIFO | <i>DMA_CHn_DST</i> |
| 2-3 | Reserved | - | - |
| 4 | UART0 | UART0 Receive FIFO | <i>DMA_CHn_DST</i> |
| 5 | UART1 | UART1 Receive FIFO | <i>DMA_CHn_DST</i> |
| 6 | Reserved | - | - |
| 7 | I ² C0 | I ² C0 Receive FIFO | <i>DMA_CHn_DST</i> |
| 8 | I ² C1 | I ² C1 Receive FIFO | <i>DMA_CHn_DST</i> |
| 9 | ADC | ADC Data Register | <i>DMA_CHn_DST</i> |
| 10 | I ² C2 | I ² C2 Receive FIFO | <i>DMA_CHn_DST</i> |
| 11-13 | Reserved | - | - |
| 14 | UART2 | UART2 Receive FIFO | <i>DMA_CHn_DST</i> |
| 15 | SPI0 | SPI0 Receive FIFO | <i>DMA_CHn_DST</i> |
| 16-27 | Reserved | - | - |
| 28 | LPUART0 (UART3) | LPUART0 (UART3) Receive FIFO | <i>DMA_CHn_DST</i> |
| 29 | Reserved | - | - |
| 30 | I ² S | I ² S Data Register | <i>DMA_CHn_DST</i> |
| 31-32 | Reserved | - | - |
| 33 | SPI1 | <i>DMA_CHn_SRC</i> | SPI1 Transmit FIFO |

| <i>DMA_CHn_CTRL.request</i> | Peripheral | DMA Source | DMA Destination |
|-----------------------------|-----------------|--------------------|-------------------------------|
| 34-35 | Reserved | - | - |
| 36 | UART0 | <i>DMA_CHn_SRC</i> | UART0 Transmit FIFO |
| 37 | UART1 | <i>DMA_CHn_SRC</i> | UART1 Transmit FIFO |
| 38 | Reserved | - | - |
| 39 | I2C0 | <i>DMA_CHn_SRC</i> | I2C0 Transmit FIFO |
| 40 | I2C1 | <i>DMA_CHn_SRC</i> | I2C1 Transmit FIFO |
| 41 | Reserved | - | - |
| 42 | I2C2 | <i>DMA_CHn_SRC</i> | I2C2 Transmit FIFO |
| 43 | Reserved | - | - |
| 44 | CRC | <i>DMA_CHn_SRC</i> | CRC Data Register |
| 45 | PCIF TX | <i>DMA_CHn_SRC</i> | PCIF Data Register |
| 46 | UART2 | <i>DMA_CHn_SRC</i> | UART2 Transmit FIFO |
| 47 | SPI0 | <i>DMA_CHn_SRC</i> | SPI0 Transmit FIFO |
| 48-59 | Reserved | - | - |
| 60 | LPUART0 (UART3) | <i>DMA_CHn_SRC</i> | LPUART0 (UART3) Transmit FIFO |
| 61 | Reserved | - | - |
| 62 | I2S | <i>DMA_CHn_SRC</i> | I2S Data Register |
| 63 | Reserved | - | - |

9.2.3 Data Movement from Source to DMA

Table 9-3 shows the fields that control the burst movement of data into the DMA FIFO. The source is a peripheral or memory.

Table 9-3: Data Movement from Source to DMA FIFO

| Register/Field | Description | Comments |
|--------------------------------|---|--|
| <i>DMA_CHn_SRC</i> | Source address | If the increment enable is set, this increments on every read cycle of the burst. This field is ignored when the DMA source is a peripheral. |
| <i>DMA_CHn_CNT</i> | Number of bytes to transfer before a CTZ condition occurs | This register is decremented on each read of the burst. |
| <i>DMA_CHn_CTRL.burst_size</i> | Burst size (1-32) | This maximum number of bytes moved during the burst read. |
| <i>DMA_CHn_CTRL.srcwd</i> | Source width | This determines the maximum data width used during each read of the AHB burst (byte, two bytes, or four bytes). The actual AHB width might be less if <i>DMA_CHn_CNT</i> is not great enough to supply all the needed bytes. |
| <i>DMA_CHn_CTRL.srinc</i> | Source increments enable | Increments <i>DMA_CHn_SRC</i> . This field is ignored when the DMA source is a peripheral. |

9.2.4 Data Movement from DMA to Destination

Table 9-4 shows the fields that control the burst movement of data out of the DMA FIFO. The destination is a peripheral or memory.

Table 9-4: Data Movement from the DMA FIFO to Destination

| Register/Field | Description | Comments |
|--------------------------------|--------------------------------------|--|
| <i>DMA_CHn_DST</i> | Destination address | If the increment enable is set, this increments on every write cycle of the burst. This field is ignored when the DMA destination is a peripheral. |
| <i>DMA_CHn_CTRL.burst_size</i> | Burst size (1-32) | The maximum number of bytes moved during a single AHB read/write burst. |
| <i>DMA_CHn_CTRL.dstwd</i> | Destination width | This determines the maximum data width used during each write of the AHB burst (one byte, two bytes, or four bytes). |
| <i>DMA_CHn_CTRL.dstinc</i> | Destination address increment enable | Increments <i>DMA_CHn_DST</i> . This field is ignored when the DMA destination is a peripheral. |

9.3 Usage

Use the following procedure to perform a DMA transfer from a peripheral's receive FIFO to memory, from memory to a peripheral's transmit FIFO, or from memory to memory.

1. Ensure *DMA_CHn_CTRL.en*, *DMA_CHn_CTRL.rlden* = 0, and *DMA_CHn_STATUS.ctz_if* = 0.
2. If using memory for the destination of the DMA transfer, configure *DMA_CHn_DST* to the starting address of the destination in memory.
3. If using memory for the source of the DMA transfer, configure *DMA_CHn_SRC* to the starting address of the source in memory.
4. Write the number of bytes to transfer to the *DMA_CHn_CNT* register.
5. Configure the following *DMA_CHn_CTRL* register fields in one or more instructions. Do not set *DMA_CHn_CTRL.en* to 1 or *DMA_CHn_CTRL.rlden* to 1 in this step:
 - a. Configure *DMA_CHn_CTRL.request* to select the transfer operation associated with the DMA channel.
 - b. Configure *DMA_CHn_CTRL.burst_size* for the desired burst size.
 - c. Configure *DMA_CHn_CTRL.pri* to set the channel priority relative to other DMA channels.
 - d. Configure *DMA_CHn_CTRL.dstwd* to dictate the number of bytes written in each transaction.
 - e. If desired, set *DMA_CHn_CTRL.dstinc* to 1 to enable automatic incrementing of the *DMA_CHn_DST* register upon every AHB transaction.
 - f. Configure *DMA_CHn_CTRL.srcwd* to dictate the number of bytes read in each transaction.
 - g. If desired, set *DMA_CHn_CTRL.srccinc* to 1 to enable automatic incrementing of the *DMA_CHn_DST* register upon every AHB transaction.
 - h. If desired, set *DMA_CHn_CTRL.dis_ie* = 1 to generate an interrupt when the channel becomes disabled. The channel becomes disabled when the DMA transfer completes, or a bus error occurs.
 - i. If desired, set *DMA_CHn_CTRL.ctz_ie* 1 to generate an interrupt when the *DMA_CHn_CNT* register is decremented to zero.
 - j. If using the reload feature, configure the reload registers to set the destination, source, and count for the following DMA transaction.
 - 1) Load the *DMA_CHn_SRCRLD* register with the source address reload value.
 - 2) Load the *DMA_CHn_DSTRLD* register with the destination address reload value.
 - 3) Load the *DMA_CHn_CTRLD* register with the count reload value.
 - k. If desired, enable the channel timeout feature described in *Channel Timeout Detect*. Clear *DMA_CHn_CTRL.to_clkdiv* to 0 to disable the channel timeout feature.
6. Set *DMA_CHn_CTRL.rlden* to 1 to enable the reload feature if using.
7. Set *DMA_CHn_CTRL.en* = 1 to immediately start the DMA transfer.
8. Wait for the interrupt flag to become 1 to indicate the completion of the DMA transfer.

9.4 Count-To-Zero (CTZ) Condition

When an AHB channel burst completes, a CTZ condition exists if *DMA_CHn_CNT* is decremented to 0.

At this point, there are two possible responses depending on the value of the *DMA_CHn_CTRL.rlden*:

1. If *DMA_CHn_CTRL.rlden* = 1, then the *DMA_CHn_SRC*, *DMA_CHn_DST*, and *DMA_CHn_CNT* registers are loaded from the reload registers, and the channel remains active and continues operating using the newly-loaded address/count values and the previously programmed configuration values.
2. If *DMA_CHn_CTRL.rlden* = 0, then the channel is disabled, and *DMA_CHn_STATUS.status* is cleared.

9.5 Chaining Buffers

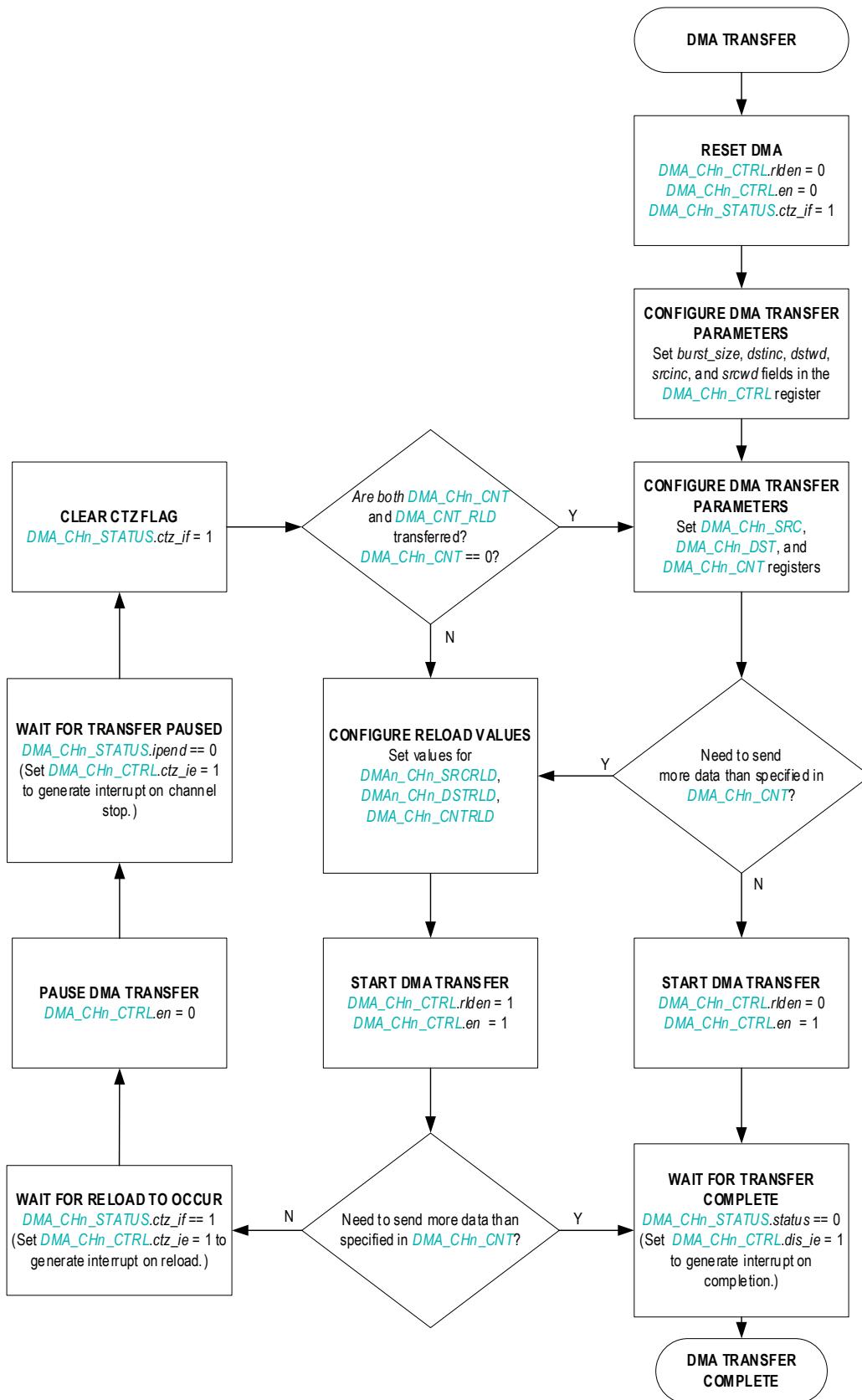
Chaining buffers reduces the DMA ISR response time and allows DMA to service requests without intermediate processing from the CPU. [Figure 9-1: DMA Block-Chaining Flowchart](#) shows the procedure for generating a DMA transfer using one or more chain buffers.

Configure the following reload registers to configure a channel for chaining:

- *DMA_CHn_SRC*
- *DMA_CHn_DST*
- *DMA_CHn_CNT*
- *DMA_CHn_SRCRLD*
- *DMA_CHn_DST*
- *DMA_CHn_CNTRLD*

Writing to any register while a channel is disabled is supported, but there are certain restrictions when a channel is enabled. The *DMA_CHn_STATUS.status* bit indicates whether the channel is enabled or not. Because an active channel might be in the middle of an AHB read/write burst, do not write to the *DMA_CHn_SRC*, *DMA_CHn_DST*, or *DMA_CHn_CNT* registers while a channel is active (*DMA_CHn_STATUS.status* = 1). To disable any DMA channel, clear the *DMAM_INTEN.ch<n>_ien* bit. Then, poll the *DMA_CHn_STATUS.status* bit to verify that the channel is disabled.

Figure 9-1: DMA Block-Chaining Flowchart



9.6 DMA Interrupts

Enable interrupts for each channel by setting `DMAm_INTEN.ch<n>.ien`. When an interrupt for a channel is pending, the corresponding `DMAm_INTFL.ch<n>.ipend` = 1. Set the corresponding enable bit to cause an interrupt when the flag is set.

A channel interrupt (`DMA_CHn_STATUS.ipend` = 1) is caused by:

- `DMA_CHn_CTRL.ctz_ie` = 1
 - ◆ If enabled all CTZ occurrences set the `DMA_CHn_STATUS.ipend` bit.
- `DMA_CHn_CTRL.dis_ie` = 1
 - ◆ If enabled, any clearing of the `DMA_CHn_STATUS.status` bit sets the `DMA_CHn_STATUS.ipend` bit. Examine the `DMA_CHn_STATUS` register to determine which reasons caused the disable. The `DMA_CHn_CTRL.dis_ie` bit also enables the `DMA_CHn_STATUS.to_if` bit. The `DMA_CHn_STATUS.to_if` bit does not clear the `DMA_CHn_STATUS.status` bit.

To clear the channel interrupt, write 1 to the cause of the interrupt (the `DMA_CHn_STATUS.ctz_if`, `DMA_CHn_STATUS.rld_if`, `DMA_CHn_STATUS.bus_err`, or `DMA_CHn_STATUS.to_if` bits).

When running in normal mode without buffer chaining (`DMA_CHn_CTRL.rlden` = 0), set the `DMA_CHn_CTRL.dis_ie` bit only. An interrupt is generated upon DMA completion or an error condition (bus error or timeout error).

When running in buffer chaining mode (`DMA_CHn_CTRL.rlden` = 1), set both the `DMA_CHn_CTRL.dis_ie` and `DMA_CHn_CTRL.ctz_ie` bits. The CTZ interrupts occur on completion of each DMA (count reaches zero and reload occurs). The setting of `DMA_CHn_CTRL.dis_ie` ensures that an error condition generates an interrupt. If `DMA_CHn_CTRL.ctz_ie` = 0, then the only interrupt occurs when the DMA completes and `DMA_CHn_CTRL.rlden` = 0 (final DMA).

9.7 Channel Timeout Detect

Each channel can optionally generate an interrupt when its associated peripheral does not request a transfer in a user-configurable period. When the timeout start conditions are met, an internal 10-bit counter begins incrementing at a frequency determined by the AHB clock, `DMA_CHn_CTRL.to_clkdiv`, and `DMA_CHn_CTRL.to_per` shown in [Table 9-5: DMA Channel Timeout Configuration](#). A channel timeout event is generated if the timer is not reset by one of the events listed below before the timeout period expires.

Table 9-5: DMA Channel Timeout Configuration

| <code>DMA_CHn_CTRL.to_clkdiv</code> | Timeout Period (μs) |
|-------------------------------------|---|
| 0 | <i>Channel timeout disabled.</i> |
| 1 | $\frac{2^8 \times [\text{Value from } \text{DMA_CHn_CTRL}.to_per]}{f_{HCLK}}$ |
| 2 | $\frac{2^{16} \times [\text{Value from } \text{DMA_CHn_CTRL}.to_per]}{f_{HCLK}}$ |
| 3 | $\frac{2^{24} \times [\text{Value from } \text{DMA_CHn_CTRL}.to_per]}{f_{HCLK}}$ |

The start of the timeout period is controlled by `DMA_CHn_CTRL.to_wait`:

- If `DMA_CHn_CTRL.to_wait` = 0, the timer begins counting immediately after `DMA_CHn_CTRL.to_per` is configured to a value other than 0 and the channel is enabled.
- If `DMA_CHn_CTRL.to_wait` = 1, the timer begins counting when the first DMA request is received from the peripheral.

The timer is reset whenever:

- The DMA request line programmed for the channel is activated.
- The channel is disabled for any reason ([DMA_CHn_STATUS.status = 0](#)).

If the timeout timer period expires, hardware sets [DMA_CHn_STATUS.to_if = 1](#) to indicate a channel timeout event has occurred. A channel timeout does not disable the DMA channel.

9.8 Memory-to-Memory DMA

Memory-to-memory transfers are processed as if the request is always active. This means that the DMA channel generates an almost constant request for the bus until its transfer is complete. For this reason, assign a lower priority to channels executing memory-to-memory transfers to prevent starvation of other DMA channels.

9.9 DMA Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 9-6: DMA Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 9-6: DMA Register Summary

| Offset | Register | Description |
|----------|----------------------------|-------------------------------|
| [0x0000] | DMAm_INTEN | DMA Control register |
| [0x0004] | DMAm_INTFL | DMA Interrupt Status register |

9.9.1 DMA Register Details

Table 9-7: DMAm Interrupt Flag Register

| DMAm Interrupt Flag | | | DMAm_INTEN | | [0x0000] |
|---------------------|-----------|--------|------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | ch<n>_ien | R/W | 0 | DMAn Channel Interrupt Enable Each bit in this field enables the corresponding channel interrupt <i>m</i> in DMAm_INTFL . Register bits associated with unimplemented channels should not be changed from their default reset value. 0: Disabled 1: Enabled | |

Table 9-8: DMAm Interrupt Enable Register

| DMAm Interrupt Enable | | | DMAm_INTFL | | [0x0004] |
|-----------------------|-------------|--------|------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | ch<n>_ipend | RO | 0 | DMAm Channel Interrupt Flag Each bit in this field represents an interrupt for the corresponding channel interrupt <n>. To clear an interrupt, clear the corresponding active interrupt bit in the DMA_CHn_STATUS register. An interrupt bit in this field is set only if the corresponding interrupt enable field is set in the DMAm_INTEN register. Register bits associated with unimplemented channels should be ignored. 0: No interrupt 1: Interrupt pending | |

9.10 DMA Channel Registers

Table 9-9: Standard DMA Channel 0 to Channel 3 Register Summary

| Offset | DMA Channel | Description |
|----------|-------------|---------------|
| [0x0100] | DMA_CH0 | DMA Channel 0 |
| [0x0120] | DMA_CH1 | DMA Channel 1 |
| [0x0140] | DMA_CH2 | DMA Channel 2 |
| [0x0160] | DMA_CH3 | DMA Channel 3 |

9.10.1 DMA Channel Register Details

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 9-10: DMA Channel Registers Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 9-10: DMA Channel Registers Summary

| Offset | Register | Description |
|----------|---------------------------------|---|
| [0x0000] | DMA_CHn_CTRL | DMA Channel n Configuration Register |
| [0x0004] | DMA_CHn_STATUS | DMA Channel n Status Register |
| [0x0008] | DMA_CHn_SRC | DMA Channel n Source Register |
| [0x000C] | DMA_CHn_DST | DMA Channel n Destination Register |
| [0x0010] | DMA_CHn_CNT | DMA Channel n Count Register |
| [0x0014] | DMA_CHn_SRCRLD | DMA Channel n Source Reload Register |
| [0x0018] | DMA_CHn_DSTRLD | DMA Channel n Destination Reload Register |
| [0x001C] | DMA_CHn_CNTRLRD | DMA Channel n Count Reload Register |

9.10.2 DMA Channel Register Details

Table 9-11: DMA_CH n Control Register

| DMA Channel n Control | | | DMA_CHn_CTRL | [0x0100] |
|-----------------------|--------|--------|--------------|--|
| Bits | Field | Access | Reset | Description |
| 31 | ctz_ie | R/W | 0 | CTZ Interrupt Enable 0: Disabled 1: Enabled. DMAm_INTFL.ch<n>.ipend is set to 1 whenever a CTZ event occurs. |
| 30 | dis_ie | R/W | 0 | Channel Disable Interrupt Enable 0: Disabled 1: Enabled. DMAm_INTFL.ch<n>.ipend bit is set to 1 whenever DMA_CHn_STATUS.status changes from 1 to 0. |
| 29 | - | RO | 0 | Reserved Do not modify |

| DMA Channel n Control | | | | DMA_CHn_CTRL | [0x0100] |
|-----------------------|------------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 28:24 | burst_size | R/W | 0 | Burst Size The number of bytes transferred into and out of the DMA FIFO in a single burst. 0b00000: 1 byte 0b00001: 2 bytes 0b00010: 3 bytes ... 0b11111: 32 bytes | |
| 23 | - | RO | 0 | Reserved Do not modify | |
| 22 | dstinc | R/W | 0 | Destination Increment Enable This bit enables the automatic increment of the DMA_CHn_DST register upon every AHB transaction. This bit is ignored for a DMA transmit to peripherals. 0: Disabled 1: Enabled | |
| 21:20 | dstwd | R/W | 0 | Destination Width Indicates the width of each AHB transaction to the destination peripheral or memory (the actual width might be less than this if there are insufficient bytes in the DMA FIFO for the full width). 0: One byte 1: Two bytes 2: Four bytes 3: Reserved | |
| 19 | - | RO | 0 | Reserved Do not modify | |
| 18 | srcinc | R/W | 0 | Source Increment on AHB Transaction Enable This bit enables the automatic increment of the DMA_CHn_SRC register upon every AHB transaction. This bit is ignored for a DMA receive from peripherals. 0: Disabled 1: Enabled | |
| 17:16 | srcwd | R/W | 0 | Source Width Indicates the width of each AHB transaction from the source peripheral or memory. The actual width might be less than this if the DMA_CHn_CNT register indicates a smaller value. 0: One byte 1: Two bytes 2: Four bytes 3: Reserved | |
| 15:14 | to_clkdiv | R/W | 0 | Timeout Timer Clock Pre-Scale Select Selects the Pre-Scale divider for the timer clock input. 0: Timer disabled. 1: $\frac{f_{HCLK}}{28}$ 2: $\frac{f_{HCLK}}{216}$ 3: $\frac{f_{HCLK}}{224}$ | |

| DMA Channel n Control | | | | DMA_CHn_CTRL | [0x0100] |
|-----------------------|---------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 13:11 | to_per | R/W | 0 | Timeout Period Select Selects the number of pre-scaled clocks seen by the channel timer before a timeout condition is generated. The value is approximate because of synchronization delays between timers <ul style="list-style-type: none"> 0: 3 to 4 1: 7 to 8 2: 15 to 16 3: 31 to 32 4: 63 to 64 5: 127 to 128 6: 255 to 256 7: 511 to 512 | |
| 10 | to_wait | R/W | 0 | Request DMA Timeout Timer Wait Enable This field controls when the timeout timer starts, either immediately when the timeout timer is enabled or after the first DMA transaction occurs. <ul style="list-style-type: none"> 0: Start timer immediately when enabled. 1: Delay timer start until after the first DMA transaction occurs. | |
| 9:4 | request | R/W | 0 | Request Select Selects the source and destination for the transfer as shown in Table 9-2: DMA Source and Destination by Peripheral . | |
| 3:2 | pri | R/W | 0 | Channel Priority Sets the priority of the channel relative to other channels of the DMA peripheral. Channels of the same priority are serviced in a round-robin fashion. <ul style="list-style-type: none"> 0: Highest priority 1: ... 2: ... 3: Lowest priority | |
| 1 | rlden | R/W | 0 | Reload Enable Setting this bit to 1 allows reloading the DMA_CHn_SRC , DMA_CHn_DST and DMA_CHn_CNT registers upon a CTZ. When this bit is set to 0 and a CTZ occurs, the channel is disabled and the DMA_CHn_STATUS.status bit is set to 0. <ul style="list-style-type: none"> 0: The channel is disabled when a CTZ occurs and the DMA_CHn_STATUS.status is set to 0. 1: Automatically reload the DMA_CHn_SRC, DMA_CHn_DST and DMA_CHn_CNT registers on a CTZ. | |
| 0 | en | R/W | 0 | Channel Enable This bit is automatically cleared when DMA_CHn_STATUS.status changes from 1 to 0. <ul style="list-style-type: none"> 0: Disabled 1: Enabled | |

Table 9-12: DMA Status Register

| DMA Channel n Status | | | DMA_CHn_STATUS | | [0x0104] |
|----------------------|---------|--------|----------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:7 | - | RO | 0 | Reserved for Future Use Do not modify this field from its reset default value. | |
| 6 | to_if | R/W1C | 0 | Timeout Interrupt Flag Timeout. Write 1 to clear. 0: No time out. 1: A channel time out has occurred | |
| 5 | - | RO | 0 | Reserved Do not modify | |
| 4 | bus_err | R/W1C | 0 | Bus Error If this bit reads 1, an AHB abort occurred and the channel was disabled by hardware. Write 1 to clear. 0: No error found 1: An AHB bus error occurred | |
| 3 | rld_if | R/W1C | 0 | Reload Interrupt Flag Reload. Write 1 to clear. 0: Reload has not occurred. 1: Reload occurred. | |
| 2 | ctz_if | R/W1C | 0 | CTZ Interrupt Flag Write 1 to clear. 0: CTZ has not occurred. 1: CTZ has occurred. | |
| 1 | ipend | RO | 0 | Channel Interrupt Pending 0: No interrupt 1: Interrupt pending | |
| 0 | status | RO | 0 | Channel Status This bit indicates when it is safe to change the configuration, address, and count registers for the channel. Whenever this bit is cleared by hardware, the <i>DMA_CHn_CTRL.en</i> bit is also cleared. 0: Disabled 1: Enabled. | |

Table 9-13: DMA Channel n Source Register

| DMA Channel n Source | | | DMA_CHn_SRC | | [0x0108] |
|----------------------|-------|--------|-------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | addr | R/W | 0 | Source Device Address For peripheral transfers, the actual address field is either ignored or forced to zero because peripherals only have one location to read/write data based on the request select chosen. If DMA_CHn_CTRL.srcinc = 1, then this register is incremented on each AHB transfer cycle by one, two, or four bytes depending on the data width. If DMA_CHn_CTRL.srcinc = 0, this register remains constant. If a CTZ condition occurs while DMA_CHn_CTRL.rlden = 1, then this register is reloaded with the contents of the DMA_CHn_SRCRLD register. | |

Table 9-14: DMA Channel n Destination Register

| DMA Channel n Destination | | | DMA_CHn_DST | | [0x010C] |
|---------------------------|-------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | addr | R/W | 0 | Destination Device Address For peripheral transfers, the actual address field is either ignored or forced to zero because peripherals only have one location to read/write data based on the request select chosen. If DMA_CHn_CTRL.dstinc = 1, then this register is incremented on every AHB transfer cycle by one, two, or four bytes depending on the data width. If a CTZ condition occurs while DMA_CHn_CTRL.rlden = 1, then this register is reloaded with the contents of the DMA_CHn_DSTRLD register. | |

Table 9-15: DMA Channel n Count Register

| DMA Channel n Count | | | DMA_CHn_CNT | | [0x0110] |
|---------------------|-------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:24 | - | RO | 0 | Reserved Do not modify | |
| 23:0 | cnt | R/W | 0 | DMA Counter Load this register with the number of bytes to transfer. This field decreases on every AHB access to the DMA FIFO. The decrement is one, two, or four bytes depending on the data width. When the counter reaches 0, a CTZ condition is triggered. If a CTZ condition occurs while DMA_CHn_CTRL.rlden = 1, then this register is reloaded with the contents of the DMA_CHn_CNTRLRD.cnt_rld field. | |

Table 9-16: DMA Channel n Source Reload Register

| DMA Channel n Source Reload | | | DMA_CHn_SRCRLD | | [0x0114] |
|-----------------------------|-------|--------|----------------|----------------------------------|----------|
| Bits | Field | Access | Reset | Description | |
| 31 | - | RO | 0 | Reserved Do not modify | |

| DMA Channel n Source Reload | | | DMA_CHn_SRCRLD | | [0x0114] |
|-----------------------------|---------|--------|----------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 30:0 | src_rld | R/W | 0 | Source Address Reload Value If DMA_CHn_CTRL.rlden = 1, then the value of this register is loaded into DMA_CHn_SRC upon a CTZ condition. | |

Table 9-17: DMA Channel n Destination Reload Register

| DMA Channel n Destination Reload | | | DMA_CHn_DSTRLD | | [0x0118] |
|----------------------------------|---------|--------|----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31 | - | RO | 0 | Reserved Do not modify | |
| 30:0 | dst_rld | R/W | 0 | Destination Address Reload Value If DMA_CHn_CTRL.rlden = 1, then the value of this register is loaded into DMA_CHn_DST upon a CTZ condition. | |

Table 9-18: DMA Channel n Count Reload Register

| DMA Channel n Count Reload | | | DMA_CHn_CNTRL | | [0x011C] |
|----------------------------|---------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:24 | - | RO | 0 | Reserved Do not modify | |
| 23:0 | cnt_rld | R/W | 0 | Count Reload Value If DMA_CHn_CTRL.rlden = 1, then the value of this register is loaded into DMA_CHn_CNT upon a CTZ condition. | |

10. Analog to Digital Converter (ADC) and Comparators (COMP)

The Analog to Digital Converter (ADC) is a 10-bit sigma-delta ADC with a single-ended input multiplexer and an integrated reference generator. The multiplexer selects an input channel from either of the 8 external analog input signals or the internal power supply inputs. The external analog input signals are defined as alternate functions on GPIO as shown in *Table 10-1, below*.

The 10-bit ADC conversions are stored as a 16-bit value selectable as most-significant bit (MSB) or least-significant bit (LSB) aligned. The 8 external analog inputs can be configured as 4 two-input comparators with interrupt capabilities. Comparator 0, COMP0, is configurable to wake the device from **SLEEP**, **LPM**, **UPM**, **STANDBY** and **BACKUP**. The remaining three comparators, COMP1, COMP2 and COMP3, are configurable as wakeup sources from **SLEEP**, **LPM**, and **UPM**.

10.1 Features

- 8MHz maximum ADC clock rate
- Two reference sources, an internal 1.22V bandgap or the V_{DDA} analog supply
- 8 External analog inputs that can be configured as 4 two-input comparators
- 10 Internal power supply monitor inputs
- Fixed 10-bit word conversion time of 1024 ADC clock cycles
- Programmable out-of-range (limit) detection
- Interrupt generation for limit detection, conversion start, conversion complete, and internal reference powered on
- Serial ADC data measurements
- ADC conversion 10 output either MSB or LSB aligned

10.2 Instances

Table 10-1: MAX78000 ADC Peripheral Pins

| Function | 81 CTBGA | |
|------------|----------|--------------------|
| | Pin | Alternate Function |
| AIN0/AIN0N | P2.0 | AF1 |
| AIN1/AIN0P | P2.1 | AF1 |
| AIN2/AIN1N | P2.2 | AF1 |
| AIN3/AIN1P | P2.3 | AF1 |
| AIN4/AIN2N | P2.4 | AF1 |
| AIN5/AIN2P | P2.5 | AF1 |
| AIN6/AIN3N | P2.6 | AF1 |
| AIN7/AIN3P | P2.7 | AF1 |

10.3 Architecture

The ADC is a first-order sigma-delta converter with a 10-bit output. The ADC operates at a maximum frequency of 8MHz with a fixed-sample rate as shown in [Equation 10-1](#). Details of selecting the ADC clock frequency, f_{adclk} , are covered in the [Clock Configuration](#) section.

Equation 10-1: ADC 10-bit Word Sample Rate

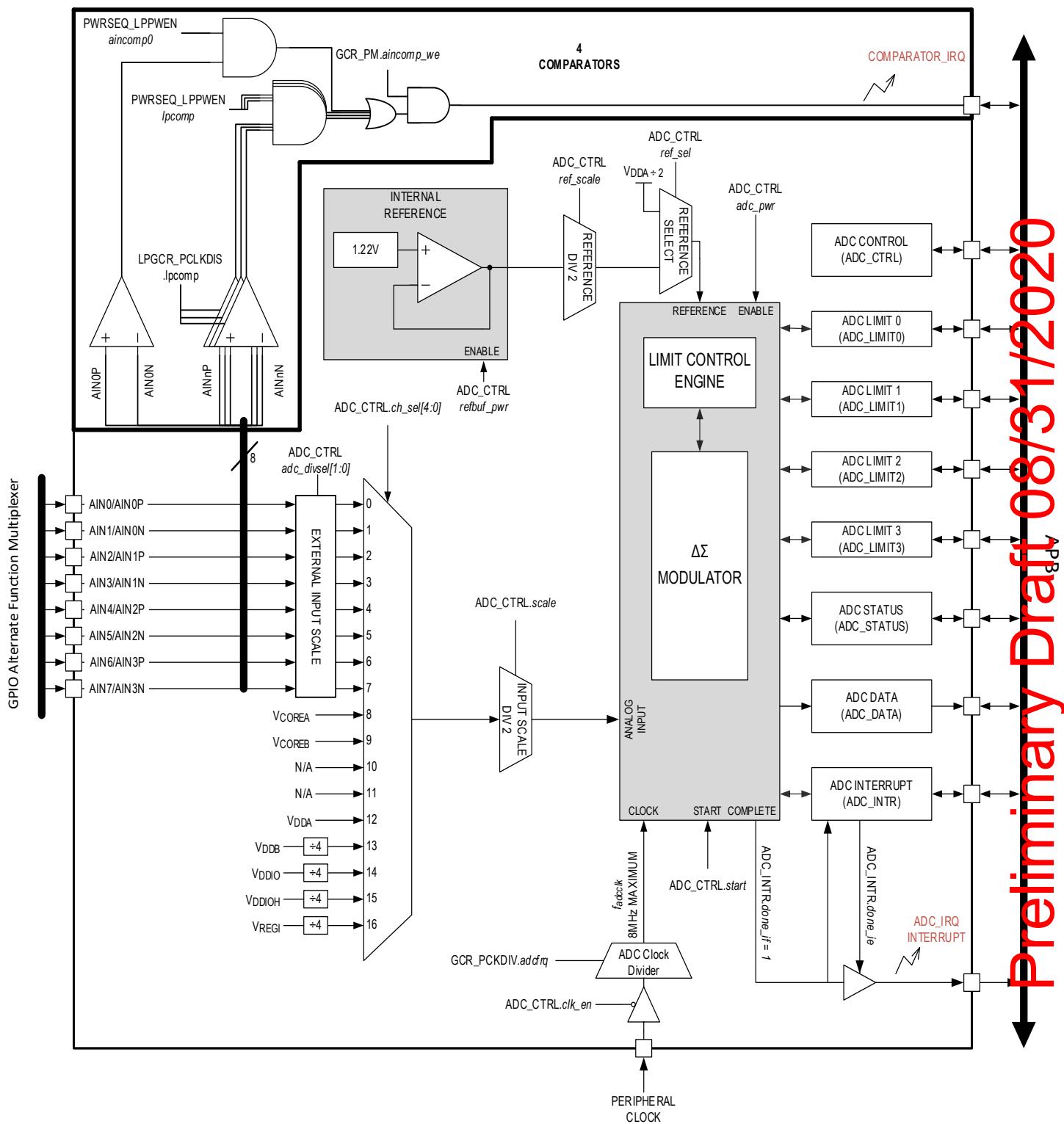
$$t_{adc_sample} = 1024 \times \left(\frac{1}{f_{adclk}} \right)$$

ADC offset is factory trimmed and automatically loaded into the ADC controller during system power-up.

The ADC uses a switched capacitor network to perform the conversion; this results in dynamic switching current and requires settling time for the external analog input signals (AIN0 – AIN7). This dynamic switching current sets the upper limit of the source impedance of the external analog input signals to approximately 10kΩ.

The ADC supports a gain of 2× to provide additional conversion resolution if the input signals are less than half the reference voltage.

Figure 10-1: Analog to Digital Converter Block Diagram



10.4 Clock Configuration

The ADC clock, $adcclk$, is controlled by the **GCR_PCLKDIV.adcfrq** register field. Configure this field for the target ADC sample frequency. The maximum clock supported by the ADC is 8MHz. The divisor selection, **GCR_PCLKDIV.adcfrq**, for the ADC depends on the peripheral clock. *Equation 10-2* shows the calculation for the ADC clock frequency, where:

$$f_{PCLK} = f_{SYSCLK}/2$$

Equation 10-2: ADC Clock Frequency

$$f_{adcclk} = \frac{f_{PCLK}}{\text{GCR_PCKDIV}.adcfrq}$$

The **GCR_PCLKDIV.adcfrq** register field setting must result in a value for $f_{adcclk} \leq 8\text{MHz}$ as shown in *Table 10-2* with the System Clock set as the IPO.

Table 10-2: ADC Clock Frequency and ADC Conversion Time ($f_{SYSCLK} = 100\text{MHz}$, $f_{PCLK} = 50\text{MHz}$)

| GCR_PCLKDIV.adcfrq | ADC Clock Frequency (Hz) f_{adcclk} | 10-Bit Word Conversion Time (μs) t_{adc_sample} |
|---------------------------|---|--|
| 0 – 7 | Invalid | Invalid |
| 8 | 6,250,000 | 164 |
| 9 | 5,555,555 | 184 |
| 10 | 5,000,000 | 205 |
| 11 | 4,545,454 | 225 |
| 12 | 4,166,666 | 246 |
| 13 | 3,846,153 | 266 |
| 14 | 3,571,428 | 287 |
| 15 | 3,333,333 | 307 |

10.5 Power-Up Sequence

Complete the following steps to configure the ADC:

1. Disable the ADC clock by setting `ADC_CTRL.clk_en` to 0.
2. Set the ADC clock (f_{adclk}) using `GCR_PCLKDIV.adcfrq`. See [Clock Configuration](#)
3. Enable the ADC clock by setting `ADC_CTRL.clk_en` to 1
4. Clear the ADC reference ready interrupt flag by writing a 1 to `ADC_INTR.ref_ready_if`.
5. Optionally enable the ADC reference ready interrupt by writing 1 to the `ADC_INTR.ref_ready_ie` field. Set up an interrupt vector for the ADC IRQ.
6. Select one of the following ADC reference sources:
 - a. Internal 1.22V bandgap reference (`ADC_CTRL.ref_sel` = 0).
 - b. $\frac{V_{DDA}}{2}$ reference (`ADC_CTRL.ref_sel` = 1).
7. Complete the following steps to enable power:
 - a. Set `ADC_CTRL.pwr` to 1 to turn on the ADC.
 - b. Set `ADC_CTRL.refbuf_pwr` to 1 to turn on the internal reference buffer If using the internal bandgap reference.
 - c. Wait until hardware sets the `ADC_INTR.ref_ready_if` bit to 1 indicating the charge pump is fully stabilized.
 - d. Clear the ADC reference ready interrupt flag by writing 1 to `ADC_INTR.ref_ready_if`.
 - e. Optionally disable the ADC reference ready interrupt (`ADC_INTR.ref_ready_ie` = 0).

10.6 Conversion

After the power-up sequence is complete, the ADC is ready for data conversion. Complete the following steps to perform a data conversion.

1. Select the ADC input channel for the conversion by setting `ADC_CTRL.ch_sel` field. See [ADC Channel Select](#) for details.
2. Optionally set input and reference scaling. See [Reference Scaling and Input Scaling](#) for details on each input channel's scale requirements.
3. Set the data alignment for the conversion output data using the `ADC_CTRL.data_align` field, 0 for LSB alignment or 1 for MSB alignment. See [Table 10-4](#) for alignment details of the DATA register.
4. Clear the ADC done interrupt flag by writing 1 to the `ADC_INTR.done_if`.
5. Optionally enable the ADC done interrupt (`ADC_INTR.done_ie` = 1), and enable the ADC interrupt vector (ADC IRQ). See the Interrupt chapter for details.
6. Start the ADC conversion by setting `ADC_CTRL.start` to 1.
7. Poll the `ADC_INTR.done_if` flag until you read 1, or wait for the ADC interrupt to occur if enabled in step 5.
8. Read the data from the `ADC_DATA.data`, and clear the ADC done interrupt flag by writing 1 to `ADC_INTR.done_if`.

10.7 Reference Scaling and Input Scaling

For small signals, the ADC input, ADC reference or both can be scaled by 50%. This enables flexibility to achieve better resolution on the ADC conversion. Each input channel, supports the default of no scaling of the input (`ADC_CTRL.scale` = 0) and no scaling of the reference (`ADC_CTRL.ref_scale` = 0). The following sections describe the scale options for each of the ADC input channels.

10.7.1 AIN0 – AIN7 Scale Limitations

The external inputs, AIN0 through AIN7, support scaling of the input by 50%, the reference by 50%, or both by 50%. Also, the scaling can further be modified by additional factors of 2, 3, or 4 as defined by [ADC_CTRL.adc_divsel](#). The scale settings for the given input signal and reference must satisfy the following equation to be valid:

Equation 10-3: Input and Reference Scale Requirements Equation

$$\frac{AINn}{2^{scale}} < \frac{V_{REF}}{2^{ref_scale}}$$

10.7.2 Scale Limitations for All Other Input Channels

For the remaining internal input channels, the scale settings must either both be disabled, or both be enabled as shown in [Table 10-3, below](#).

Table 10-3: Input and Reference Scale Support by ADC Input Channel

| ADC Channel | ADC Input Signal | ADC_CTRL.scale | ADC_CTRL.ref_scale |
|-------------|-----------------------|--------------------------------|------------------------------------|
| 8 | V_{COREA} | 0 | 0 |
| | | 1 | 1 |
| 9 | V_{COREB} | 0 | 0 |
| | | 1 | 1 |
| 10 | N/A | - | - |
| | | - | - |
| 11 | N/A | - | - |
| | | - | - |
| 12 | V_{DDA} | 0 | 0 |
| | | 1 | 1 |
| 13 | $\frac{V_{DDB}}{4}$ | 0 | 0 |
| | | 1 | 1 |
| 14 | $\frac{V_{DDIO}}{4}$ | 0 | 0 |
| | | 1 | 1 |
| 15 | $\frac{V_{DDIOH}}{4}$ | 0 | 0 |
| | | 1 | 1 |
| 16 | $\frac{V_{REGI}}{4}$ | 0 | 0 |
| | | 1 | 1 |

10.7.3 Data Conversion Output Alignment

The ADC outputs a total of 10-bits per conversion and stores the data in the DATA register LSB justified by default. [Table 10-4](#) shows the ADC data alignment based on the value of the [ADC_CTRL.data_align](#) bit.

Table 10-4: ADC Data Register Alignment Options

| ADC_CTRL.data_align = 0 | | | | | | | | | | | | | | | | | | | | | |
|---|-----|----|----|----|----|----|------|---|---|---|---|---|---|---|-----|---|--|--|--|--|--|
| | MSB | | | | | | | | | | | | | | LSB | | | | | | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | |
| ADC_DATA | 0 | 0 | 0 | 0 | 0 | 0 | data | | | | | | | | | | | | | | |

| <i>ADC_CTRL.data_align = 1</i> | | | | | | | | | | | | | | | | |
|--------------------------------|------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|-----|
| | MSB | | | | | | | | | | | | | | | LSB |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| <i>ADC_DATA</i> | data | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 |

10.7.4 Data Conversion Value Equations

Use the following equations to calculate the ADC data value for a conversion for the selected channel. If using the internal reference, VREF = 1.22V; otherwise VREF = VDDA.

Equation 10-4: ADC Data Calculation for Input Signal $ADC_CTRL.ch_sel = 0x00$ thru $0x07$ (A_{IN0} – A_{IN7})

$$ADC_DATA = \text{round} \left\{ \left(\frac{\left(\frac{Input\ Signal}{2^{scale}} * (adc_divsel + 1) \right)}{\left(\frac{V_{REF}}{2^{ref_scale}} \right)} \right) \times (2^{10} - 1) \right\}$$

Note: Must satisfy Equation 10-3.

Equation 10-5: ADC Data Equation for Input Signal $ADC_CTRL.ch_sel = 0x08$ thru $0x0C$ (V_{COREA} , V_{COREB} , V_{DDA})

$$ADC_DATA = \text{round} \left\{ \left(\frac{\left(\frac{Input\ Signal}{2^{scale}} \right)}{\left(\frac{V_{REF}}{2^{ref_scale}} \right)} \right) \times (2^{10} - 1) \right\}$$

Note: See Table 10-3 for limitations.

Equation 10-6: ADC Data Calculation Input Signal $ADC_CTRL.ch_sel = 0x0D$ thru $0x10$ (V_{DDB} , V_{DDIO} , V_{DDIOH} , V_{REGI})

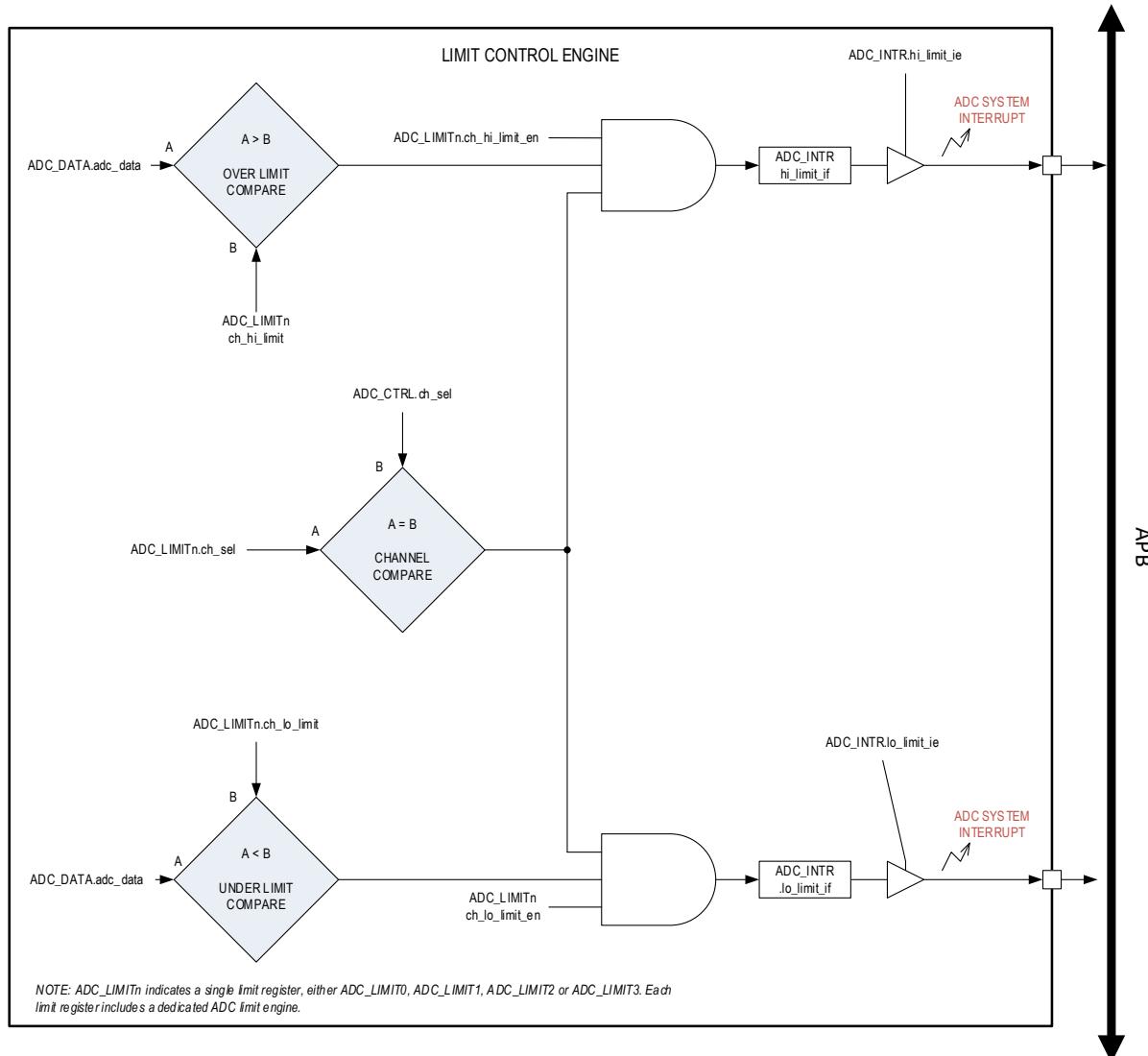
$$ADC_DATA = \text{round} \left\{ \left(\frac{\left(\frac{Input\ Signal}{4^{scale}} \right)}{\left(\frac{V_{REF}}{2^{ref_scale}} \right)} \right) \times (2^{10} - 1) \right\}$$

Note: See Table 10-3 for limitations.

10.7.5 Data Limits and Out of Range Interrupts

Channel limits are implemented to minimize power consumption for power supply monitoring. The ADC includes four limit registers, *ADC_LIMIT0* to *ADC_LIMIT3*, that you can use to set a high limit, low limit, and the ADC channel number to apply the limits against. A block diagram of the limit engine for each of the four limit registers is shown in [Figure 10-2](#).

Figure 10-2: ADC Limit Engine



When a measurement is taken on the ADC, the limit engine determines if the channel measured matches one of the channels selected by the limit registers. If it does and the data converted is above or below the high or low limit, an interrupt flag is set resulting in an ADC interrupt if the interrupt is enabled.

Complete the following steps to enable a high and low limit for an ADC input channel using the [ADC_LIMIT0](#) register. Perform these steps after the ADC is configured for measurement, and the configuration is identical for all four limit registers except for the limit register name:

1. Verify the ADC is not actively taking a measurement by checking [ADC_STATUS.active](#) until it reads 0.
2. Set [ADC_LIMIT0.ch_sel](#) field to the selected channel for the high and low limit.
3. Set the high limit, [ADC_LIMIT0.ch_hi_limit](#), to the selected 10-bit trip point. When enabled, an ADC measurement greater than this field on the channel selected ([ADC_LIMIT0.ch_sel](#)) generates an ADC interrupt.
4. Set the low limit, [ADC_LIMIT0.ch_lo_limit](#), to the selected 10-bit low trip point. When enabled, an ADC measurement lower than this field on the channel selected ([ADC_LIMIT0.ch_sel](#)) generates an ADC interrupt.
5. Enable the high limit, the low limit, or both interrupt signals by writing a 1 to [ADC_LIMIT0.ch_high_limit_en](#), [ADC_LIMIT0.ch_low_limit_en](#), or both. Note: Each limit register is independently enabled for high- and low-limit interrupts.
6. Clear the ADC interrupt high and low interrupt flags by writing 1 to [ADC_INTR.hi_limit_if](#) and [ADC_INTR.lo_limit_if](#).
7. Enable the high, low, or both interrupts for the ADC by setting [ADC_INTR.hi_limit_if](#) to 1, [ADC_INTR.lo_limit_ie](#) to 1, or both.
8. If an ADC conversion occurs that is above or below the enabled limits, an ADC_IRQ is generated with the [ADC_LIMIT0.adc_high_limit_if](#), [ADC_LIMIT0.adc_low_limit_if](#), or both set to 1. The [ADC_CTRL.ch_sel](#) value indicates the channel that caused the interrupt, and the value of the ADC conversion that is out of bounds is in the [ADC_DATA.data](#) field.

10.7.5.1 Power-Down Sequence

Complete the following steps to power-down the ADC:

1. Set [ADC_CTRL.pwr](#) to 0, disabling the ADC converter power.
2. [ADC_CTRL.refbuf_pwr](#) to 0, disabling the internal reference buffer power.
3. Set [ADC_CTRL.clk_en](#) to 0, disabling the ADC internal clock.

10.8 Comparator Operation

10.8.1 Comparator 0 Usage

Comparator 0 is controlled individually using the [MCR_CMPO_CTRL](#) register. Enable comparator 0 by setting the [MCR_CMPO_CTRL.en](#) field to 1. Comparator 0's output is readable using the [MCR_CMPO_CTRL.out](#). Enable interrupt events for comparator 0 by setting the [MCR_CMPO_CTRL.int_en](#) field to 1. Interrupts for comparator 0 occur when the output changes to its active state. The active state is controlled using the [MCR_CMPO_CTRL.pol](#) field. When the output state is active, hardware automatically sets the [MCR_CMPO_CTRL.if](#) flag to 1. To clear the interrupt flag, write 1 to [MCR_CMPO_CTRL.if](#).

10.8.2 Low-Power Comparators 1, 2 and 3 Usage

Comparator 1, 2 and 3 are controlled using the Low Power Comparator, [LPCOMPn](#), registers.

10.8.3 Using Comparator 0 as a Wakeup Source

After configuring Comparator 0, configure it as a wakeup source from **SLEEP**, **LPM**, **UPM**, **STANDBY** and **BACKUP** by performing the following steps:

1. Enable comparator wakeup events by setting [*GCR_PM.comp*](#) to 1.
2. Enable comparator 0 as a wakeup source by setting [*PWRSEQ_LPPWEN.comp0*](#) to 1.
3. If desired, provide an IRQ handler for the comparators ([*LPCOMP_IRQHandler*](#)).

After the device exists a low power mode, determine if the wakeup event was the result of Comparator 0 by checking the [*PWRSEQ_LPPWST.comp0*](#) and the [*MCR_CMPO_CTRL.if*](#). Wakeup events generated by Comparator 0 from **STANDBY** and **BACKUP** mode result in the [*PWRSEQ_LPPWST.comp0*](#) bit being set. Write 1 to clear the [*PWRSEQ_LPPWST.comp0*](#) bit and the [*MCR_CMPO_CTRL.if*](#) bit.

10.8.4 Low-Power Comparators 1, 2 and 3 Wakeup

Wake up from the low power comparators, [*LPCOMPn*](#), by setting [*PWRSEQ_LPPWEN.ipcomp*](#) bit to 1. If any of the three low-power comparators ([*LPCOMPn*](#)) cause the device to wake up, the specific comparator's interrupt flag is set to 1. Inspection of each comparator's interrupt flag identifies which comparator resulted in the wakeup event, refer to the [*LPCOMPn.if*](#) for details.

Enable wakeup events from the low-power comparators by setting the [*PWRSEQ_LPPWEN.comp*](#) field to 1. If a comparator event occurs and wakes the device from a low power operating mode, the [*PWRSEQ_LPPWST.comp*](#) field is set to 1. Clear the comparator wakeup status flag by writing 1 to [*PWRSEQ_LPPWST.comp*](#).

*Note: Comparator 0, if enabled, wakes the device from **SLEEP**, **LPM**, **UPM**, **STANDBY** and **BACKUP**. Comparators 1, 2 and 3, if enabled, wake the device from **SLEEP**, **LPM** and **UPM**.*

Comparator 0 is individually enabled using [*MCR_CMPO_CTRL.en*](#). When the inputs to any comparator cross their bias potential, the flag bit in the [*PWRSEQ_LPPWST*](#) register will be set. Should the user desire, an interrupt can be generated by setting the [*aincomp0*](#) bit in the [*PWRSEQ_LPPWST*](#) register. To enable Wakeup Events from comparator 0, set [*PWRSEQ_LPPWEN.comp0*](#) field to 1. The interrupts must be globally enabled by setting the [*GCR_PM.comp_we*](#) bit.

10.9 ADC Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 10-5. ADC Registers Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 10-5. ADC Registers Summary

| Offset | Name | Description |
|----------|----------------------------|--------------------------------|
| [0x0000] | ADC_CTRL | ADC Control Register |
| [0x0004] | ADC_STATUS | ADC Status Register |
| [0x0008] | ADC_DATA | ADC Output Data Register |
| [0x000C] | ADC_INTR | ADC Interrupt Control Register |
| [0x0010] | ADC_LIMIT0 | ADC Limit 0 Register |
| [0x0014] | ADC_LIMIT1 | ADC Limit 1 Register |
| [0x0018] | ADC_LIMIT2 | ADC Limit 2 Register |
| [0x001C] | ADC_LIMIT3 | ADC Limit 3 Register |

10.9.1 ADC Register Details

Table 10-6: ADC Control Register

| ADC Control | | ADC_CTRL | | | [0x0000] |
|-------------|------------|----------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:21 | - | RO | 0x050 | Reserved for Future Use Do not modify this field. | |
| 20 | data_align | R/W | 0 | ADC Data Alignment Selects the alignment of the 16-bit data conversion stored in the DATA register. 0: Data is LSB justified in 16-bit DATA register. DATA[15:10] = 0. 1: Data is MSB justified in 16-bit DATA register. DATA[5:0] = 0. | |
| 19 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 18:17 | adc_divsel | R/W | 0 | External Input Scale Scales the external inputs AIN0-AIN7. All eight of external inputs are scaled by the same value 0: No scaling. 1: Divide by 2 2: Divide by 3 3: Divide by 4 | |

| ADC Control | | | ADC_CTRL | | [0x0000] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------|-------------------------|-------------------------|----------|--|----------|--------|-------------------|-------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|---|--------|------|---|--------|------|----|--------|------|----|--------|------|----|------|------|----|----------|------|----|-----------|------|----|------------|------|----|-----------|-------------|-------------------------|-------------------------|
| Bits | Field | Access | Reset | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16:12 | ch_sel | R/W | 0 | ADC Channel Select Selects the active channel for the next ADC conversion. <table border="1" data-bbox="660 333 1330 946"> <thead> <tr> <th>ch_sel</th><th>ADC Input Channel</th><th>Input</th></tr> </thead> <tbody> <tr><td>0x00</td><td>0</td><td>AIN0</td></tr> <tr><td>0x01</td><td>1</td><td>AIN1</td></tr> <tr><td>0x02</td><td>2</td><td>AIN2</td></tr> <tr><td>0x03</td><td>3</td><td>AIN3</td></tr> <tr><td>0x04</td><td>4</td><td>AIN4</td></tr> <tr><td>0x05</td><td>5</td><td>AIN5</td></tr> <tr><td>0x06</td><td>6</td><td>AIN6</td></tr> <tr><td>0x07</td><td>7</td><td>AIN7</td></tr> <tr><td>0x08</td><td>8</td><td>VCOREA</td></tr> <tr><td>0x09</td><td>9</td><td>VCOREB</td></tr> <tr><td>0x0A</td><td>10</td><td>VRXOUT</td></tr> <tr><td>0x0B</td><td>11</td><td>VTXOUT</td></tr> <tr><td>0x0C</td><td>12</td><td>VDDA</td></tr> <tr><td>0x0D</td><td>13</td><td>VDDB / 4</td></tr> <tr><td>0x0E</td><td>14</td><td>VDDIO / 4</td></tr> <tr><td>0x0F</td><td>15</td><td>VDDIOH / 4</td></tr> <tr><td>0x10</td><td>16</td><td>VREGI / 4</td></tr> <tr><td>0x11 – 0x1F</td><td>Reserved for future use</td><td>Reserved for future use</td></tr> </tbody> </table> | | ch_sel | ADC Input Channel | Input | 0x00 | 0 | AIN0 | 0x01 | 1 | AIN1 | 0x02 | 2 | AIN2 | 0x03 | 3 | AIN3 | 0x04 | 4 | AIN4 | 0x05 | 5 | AIN5 | 0x06 | 6 | AIN6 | 0x07 | 7 | AIN7 | 0x08 | 8 | VCOREA | 0x09 | 9 | VCOREB | 0x0A | 10 | VRXOUT | 0x0B | 11 | VTXOUT | 0x0C | 12 | VDDA | 0x0D | 13 | VDDB / 4 | 0x0E | 14 | VDDIO / 4 | 0x0F | 15 | VDDIOH / 4 | 0x10 | 16 | VREGI / 4 | 0x11 – 0x1F | Reserved for future use | Reserved for future use |
| ch_sel | ADC Input Channel | Input | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x00 | 0 | AIN0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x01 | 1 | AIN1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x02 | 2 | AIN2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x03 | 3 | AIN3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x04 | 4 | AIN4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x05 | 5 | AIN5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x06 | 6 | AIN6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x07 | 7 | AIN7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x08 | 8 | VCOREA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x09 | 9 | VCOREB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0A | 10 | VRXOUT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0B | 11 | VTXOUT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0C | 12 | VDDA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0D | 13 | VDDB / 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0E | 14 | VDDIO / 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0F | 15 | VDDIOH / 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x10 | 16 | VREGI / 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x11 – 0x1F | Reserved for future use | Reserved for future use | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | clk_en | R/W | 0 | ADC Clock Enable 0: Disabled 1: Enabled | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | - | RO | 0 | Reserved for Future Use Do not modify this field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | scale | R/W | 0 | ADC Input Scale Scales ADC input by 50 percent. 0: ADC input is not scaled 1: ADC input is scaled by $\frac{1}{2}$. <i>Note: See Reference Scaling and Input Scaling for valid settings for each ADC input.</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | ref_scale | R/W | 0 | Reference Scale Scales the internal bandgap reference by 50 percent. 0: Internal bandgap reference is not scaled. 1: Internal bandgap reference is scaled by $\frac{1}{2}$. <i>Note: See Reference Scaling and Input Scaling for valid settings for each ADC input</i> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7:5 | - | RO | 0 | Reserved for Future Use Do not modify this field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | ref_sel | R/W | 0 | ADC Reference Select 0: Internal bandgap reference is used for the ADC reference 1: $V_{DDA} \div 2$ is used for the ADC reference | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | refbuf_pwr | R/W | 0 | Reference Buffer Power Enable 0: Disabled 1: Enabled | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | - | RO | 0 | Reserved for Future Use Do not modify this field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| ADC Control | | | ADC_CTRL | | [0x0000] |
|-------------|-------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 1 | pwr | R/W | 0 | ADC Power Enable Set this field to 1 to enable power to the ADC peripheral. 0: Disabled 1: Enabled | |
| 0 | start | R/W | 0 | Start ADC Conversion Write this bit to 1 to start an ADC conversion. When the conversion is complete, the hardware automatically sets this bit to 0 indicating the conversion is complete. 0: ADC inactive or data conversion complete. 1: Start ADC conversion and remains set until complete. | |

Table 10-7: ADC Status Register

| ADC Status | | | ADC_STATUS | | [0x0004] |
|------------|-------------------|--------|------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 3 | overflow | RO | 0 | ADC Overflow Flag 0: No overflow on last conversion 1: Overflow on last conversion | |
| 2 | afe_pwr_up_active | RO | 0 | ADC Power-Up State This field is set to 1 when the ADC charge pump is powering up. 0: AFE is not in power-up delay. 1: AFE is currently in the power-up delay state. | |
| 1 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 0 | active | RO | 0 | ADC Conversion in Progress 0: ADC is idle 1: ADC conversion is in progress | |

Table 10-8: ADC Data Register

| ADC Data | | | ADC_DATA | | [0x0008] |
|----------|-------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 15:0 | data | RO | 0 | ADC Data This field holds the ADC conversion output data. See the Data Conversion Output Alignment for details. | |

Table 10-9: ADC Interrupt Control Register

| ADC Interrupt Control | | | ADC_INTR | | [0x000C] |
|-----------------------|-------------|--------|----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:23 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 22 | pending | RO | 0 | ADC Interrupt Pending 0: No ADC interrupt pending. 1: At least one ADC interrupt is pending, and the corresponding interrupt enable bit is set. | |
| 21 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 20 | overflow_if | R/W1C | 0 | ADC Overflow Interrupt Flag 1: The last conversion resulted in an overflow | |

| ADC Interrupt Control | | | ADC_INTR | | [0x000C] |
|-----------------------|--------------|--------|----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 19 | lo_limit_if | R/W1C | 0 | ADC Low Limit Interrupt Flag 1: The last conversion resulted in a low-limit condition for one of the limit registers. | |
| 18 | hi_limit_if | R/W1C | 0 | ADC High Limit Interrupt Flag 1: The last conversion resulted in a high-limit condition for one of the limit registers. | |
| 17 | ref_ready_if | R/W1C | 0 | ADC Reference Ready Interrupt Flag 0: Not Ready 1: Ready. | |
| 16 | done_if | R/W1C | 0 | ADC Conversion Complete Interrupt Flag Set by the ADC hardware when an ADC conversion is complete. 1: ADC conversion complete | |
| 15:5 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 4 | overflow_ie | R/W | 0 | ADC Overflow Interrupt Enable 0: Disabled. 1: Enables interrupt assertion when hardware sets ADC_INTR.overflow_if . | |
| 3 | lo_limit_ie | R/W | 0 | ADC Low Limit Interrupt Enable 0: Disabled. 1: Enables interrupt assertion when hardware sets the ADC_INTR.lo_limit_if . | |
| 2 | hi_limit_ie | R/W | 0 | ADC High Limit Interrupt Enable 0: Disabled. 1: Enables interrupt assertion when hardware sets ADC_INTR.hi_limit_if . | |
| 1 | ref_ready_ie | R/W | 0 | ADC Reference Ready Interrupt Enable 0: Disabled. 1: Enables interrupt assertion when hardware sets ADC_INTR.ref_ready_if . | |
| 0 | done_ie | R/W | 0 | ADC Conversion Complete 0: Disabled. 1: Enables interrupt assertion when hardware sets ADC_INTR.done_if . | |

Table 10-10: ADC Limit 0 to 3 Registers

| ADC Limit 0 | | | ADC_LIMIT0 | | [0x0010] |
|-------------|----------------|--------|------------|---|----------|
| ADC Limit 1 | | | ADC_LIMIT1 | | [0x0014] |
| ADC Limit 2 | | | ADC_LIMIT2 | | [0x0018] |
| ADC Limit 3 | | | ADC_LIMIT3 | | [0x001C] |
| Bits | Field | Access | Reset | Description | |
| 31:30 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 29 | ch_hi_limit_en | R/W | 0 | High Limit Monitoring Enable If set, then an ADC conversion that results in a value greater than the ch_high_limit field generates an ADC interrupt if the ADC high-limit interrupt is enabled (ADC_INTR.hi_limit_ie = 1). 1: The high-limit comparison for the ch_sel channel is active. 0: The high-limit comparison is not enabled. | |
| 28 | ch_lo_limit_en | R/W | 0 | Low Limit Monitoring Enable If set, then an ADC conversion that results in a value less than the ch_low_limit field generates an ADC interrupt if the ADC low-limit interrupt is enabled (ADC_INTR.lo_limit_ie = 1). 1: The low-limit comparison for the ch_sel channel is active. 0: The low-limit comparison is not enabled. | |

| ADC Limit 0 | | | ADC_LIMIT0 | [0x0010] |
|--------------------|-------------|--------|-------------------|--|
| ADC Limit 1 | | | ADC_LIMIT1 | [0x0014] |
| ADC Limit 2 | | | ADC_LIMIT2 | [0x0018] |
| ADC Limit 3 | | | ADC_LIMIT3 | [0x001C] |
| Bits | Field | Access | Reset | Description |
| 27:24 | ch_sel | R/W | 0 | ADC Channel for Limit Monitoring Sets the ADC input channel for high- and low-limit thresholds. See ADC_CTRL.ch_sel for valid values for this field. |
| 23:22 | - | RO | 0 | Reserved for Future Use Do not modify this field. |
| 21:12 | ch_hi_limit | R/W | 0x3FF | High Limit Threshold Sets the threshold for high-limit comparisons. This field is a 10-bit value compared against any ADC conversion on the channel set in the ch_sel field. ADC conversions greater than this field are over threshold and can result in interrupt assertion if the ch_hi_limit_en field is set. Valid values for this field are 0x000 to 0x3FF. |
| 11:10 | - | RO | 0 | Reserved for Future Use Do not modify this field. |
| 9:0 | ch_lo_limit | R/W | 0 | Low Limit Threshold Sets the threshold for low-limit comparisons. This field is a 10-bit value compared against any ADC conversion on the channel set in the ch_sel field. ADC conversions less than this field are under threshold and can result in interrupt assertion if the ch_lo_limit_en field is set. Valid values for this field are 0x000 to 0x3FF. |

10.10 Low-Power Comparator Registers

See [Table 2-4: APP Peripheral Base Address Map](#) for the base address of this peripheral/module.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 10-11. Low-Power Comparator Registers Summary

| Offset | Name | Description |
|----------|---------|---------------------------------|
| [0x0000] | LPCOMP1 | Low Power Comparator 1 Register |
| [0x0004] | LPCOMP2 | Low Power Comparator 2 Register |
| [0x0008] | LPCOMP3 | Low Power Comparator 3 Register |

10.10.1 Low-Power Comparator Register Details

Table 10-12: Low-Power Comparator n Registers

| Low Power Comparator 1 | | | LPCOMP1 | [0x0000] |
|-------------------------------|-------|--------|----------------|----------------------------------|
| Low Power Comparator 2 | | | LPCOMP2 | [0x0004] |
| Low Power Comparator 3 | | | LPCOMP3 | [0x0008] |
| Bits | Field | Access | Reset | Description |
| 31:16 | - | RO | 0 | Reserved Do not modify |

| Low Power Comparator 1 | | | LPCOMP1 | | [0x0000] |
|------------------------|--------|--------|---------|--|----------|
| Low Power Comparator 2 | | | LPCOMP2 | | [0x0004] |
| Low Power Comparator 3 | | | LPCOMP3 | | [0x0008] |
| Bits | Field | Access | Reset | Description | |
| 15 | if | R/W1C | 0 | Low-Power Comparator n Interrupt Flag This field is set to 1 by hardware when the comparator output changes to the active state as set using the <i>pol</i> field. Write 1 to clear this flag. 0: No interrupt 1: Interrupt occurred | |
| 14 | out | RO | * | Low-Power Comparator n Output This field is the comparator's output state. 0: Output low 1: Output high | |
| 13:7 | - | RO | 0 | Reserved Do not modify | |
| 6 | int_en | R/W | 0 | Low-Power Comparator n Interrupt Enable Set this field to 1 to enable the IRQ for specific low-power comparator. 0: Interrupt disabled 1: Interrupt enabled | |
| 5 | pol | R/W | 0 | Comparator n Interrupt Polarity Select Set this field to select the polarity of the output change that generates a low-power comparator interrupt. 0: Interrupt occurs from a transition from low to high 1: Interrupt occurs from a transition from high to low | |
| 4:1 | - | RO | 0 | Reserved Do not modify | |
| 0 | en | R/W | 0 | Low Power Comparator n Enable Set this field to 1 to enable the comparator 0: Comparator disabled 1: Comparator enable | |

11. UART (UART/LPUART)

The universal asynchronous receiver/transmitter (UART) and the low-power universal asynchronous receiver/transmitter (LPUART) interfaces communicate with external devices using industry standard serial communications protocols. The UARTs are full-duplex serial ports. Each UART instance can be configured independently except for shared external clock sources.

The LPUART is a special version of the peripheral that can receive characters at 9600 baud while in the low power modes shown in [Table 11-1: MAX78000 UART/LPUART Instances](#). Valid characters are loaded into the RX FIFO and can generate a wakeup to ACTIVE mode if enabled.

DMA transfers are supported, and each instance has separate channels to its RX and TX FIFOs. DMA capability is not available while in LPM, and μ PM modes.

The peripheral provides the following features:

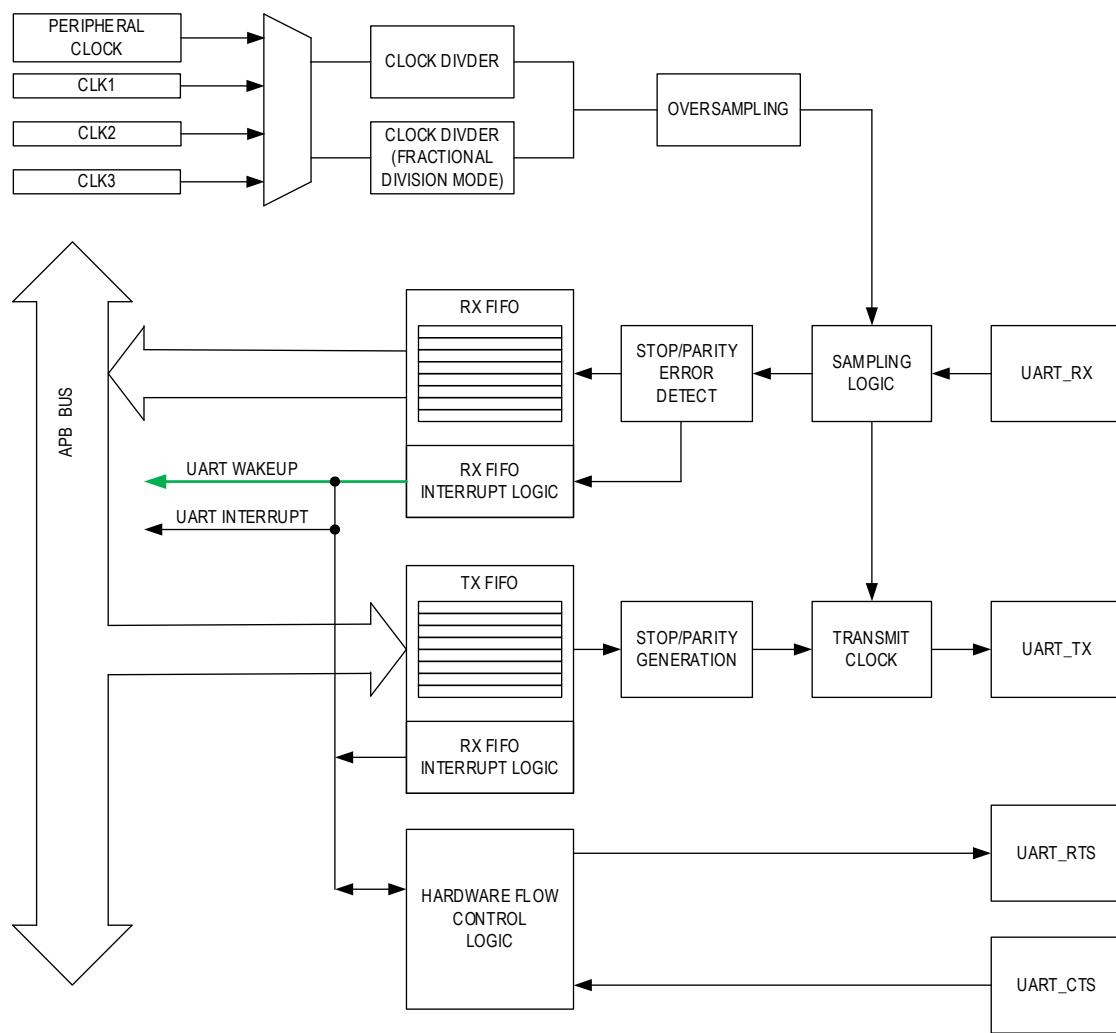
- Flexible baud rate generation up to 12.5Mbps for UART
- Flexible baud rate generation up to 1.85Mbps for UART
- Programmable character size of 5-bits to 8-bits
- Stop bit settings of 1, 1.5, or 2-bits
- Parity settings of even, odd, mark (always 1), space (always 0), and no parity
- Automatic parity error detection with selectable parity bias
- Automatic framing error detection
- Separate 8-byte transmit and receive FIFOs
- Flexible interrupt conditions
- Hardware flow control for RTS and CTS
- DMA capable

The LPUART instance provides these additional features:

- Ability to receive characters in ACTIVE, SLEEP, LPM, and μ PM modes
- Fractional baud rate divisor settings to allow greater accuracy at slow baud rates
- Wakeup to ACTIVE on multiple RX FIFO conditions

[Figure 11-1: UART Block Diagram](#) is a high-level description of the UART peripheral.

Figure 11-1: UART Block Diagram



11.1 Instances

Instances of the peripheral are shown in *Table 11-1: MAX78000 UART/LPUART Instances*

The standard UARTs and the Low Power UARTs are functionally similar, so for common functionality they are referred to as UART. The Low Power UART instance supports fractional division mode (FDM) and is referenced by the phrase "instances supporting the FDM function."

Table 11-1: MAX78000 UART/LPUART Instances

| Instance | Register Access Name | FDM Support | Power Modes | Peripheral Clock | Clock Option 2 | Clock Option 3 | Hardware Flow Control | TX/RX FIFO Depth |
|----------|----------------------|-------------|-----------------------------|------------------|-------------------|----------------|-----------------------|------------------|
| UART0 | UART0 | No | ACTIVE SLEEP | PCLK | INRO [†] | ERFO | Yes | 8/8 |
| UART1 | UART1 | | | | | | No | |
| UART2 | UART2 | | | | | | No | |
| LPUART0 | UART3 | Yes | ACTIVE SLEEP LPM UPM | PCLK | ERTCO | INRO* | No | 8/8 |

[†]Refer to the data sheet for the accuracy of this oscillator and how it may affect baud rate operation.

11.2 DMA

Each peripheral instance supports DMA transfers in both the transmit and receive directions with a 32-byte TX FIFO with a dedicated DMA channel and a 32-byte RX FIFO with a dedicated DMA channel

The DMA channels are configured using the DMA Configuration Register, [*UARTn_DMA*](#). Enable the RX FIFO DMA channel by setting [*UARTn_DMA.rxdma_en*](#) to 1 and enable the TX FIFO DMA channel by setting [*UARTn_DMA.txdma_en*](#) to 1. DMA transfers are automatically triggered based on the number of bytes in the RX or TX FIFO

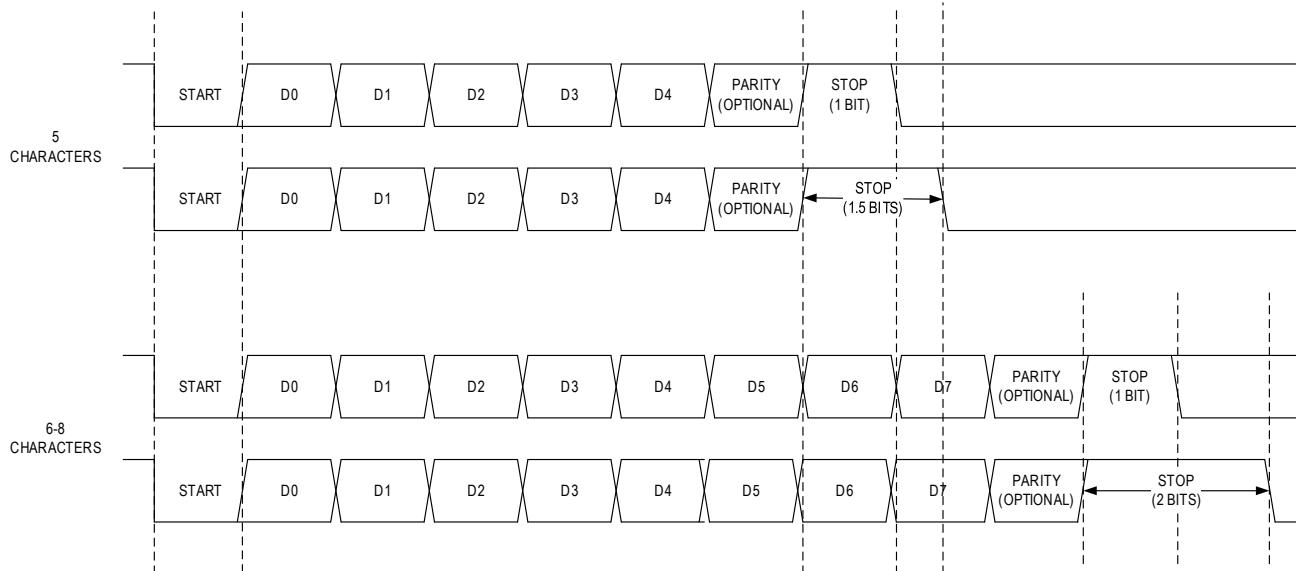
The followings describe the behavior of the RX and TX DMA requests when the DMA function is enabled.

- RX DMA request is asserted when the number of valid bytes in the RX FIFO is greater than or equal to the RX FIFO threshold.
- TX DMA request is asserted when the number of valid bytes in the TX FIFO is less than the TX FIFO threshold.

11.3 UART Frame

Figure 11-2: UART Frame Structure shows a UART frame. Character sizes of 5 to 8 bits are configurable through the [*UARTn_CTRL.size*](#) field. Stop bits are configurable for as 1 or 1.5 bits for 5-character frames and 1 or 2 stop bits for 6, 7 or 8-character frames. Parity support includes even, odd, mark, space or none.

Figure 11-2: UART Frame Structure



11.4 FIFOs

Separate read (RXFIFO) and write (TX FIFO) FIFOs are provided. They are both accessed through the same `UARTn_DATA.data` field. The current level of the TX FIFO is read from `UARTn_STATUS.tx_num`, and the current level of the RX FIFO is read from `UARTn_STATUS.rx_num`.

11.4.1 TX FIFO Operation

Writing data to `UARTn_DATA.data` increments the TX FIFO pointer, `UARTn_STATUS.tx_num`, and loads the data into the TX FIFO. The `UARTn_TXFIFO.data` register provides a feature that allows software to "peek" at the current value of the write-only TX FIFO without changing `UARTn_STATUS.tx_num`.

Writes to the TX FIFO will be ignored while `UARTn_STATUS.tx_num = C_TX_FIFO_DEPTH` shown in [Table 11-1: MAX78000 UART/LPUART Instances](#)

11.4.2 RX FIFO Operation

Reads of `UARTn_DATA.data` return the character values in the RX FIFO and decrement `UARTn_STATUS.rx_num`. Data for character sizes less than 7 bits are right justified. An overrun event will occur if a valid frame, including parity, is detected while `UARTn_STATUS.rx_num = C_RX_FIFO_DEPTH` shown in [Table 11-1: MAX78000 UART/LPUART Instances](#). In this case the frame will be discarded.

A parity error event indicates that the value read from `UARTn_DATA.data` contains a parity error.

11.4.3 Flushing

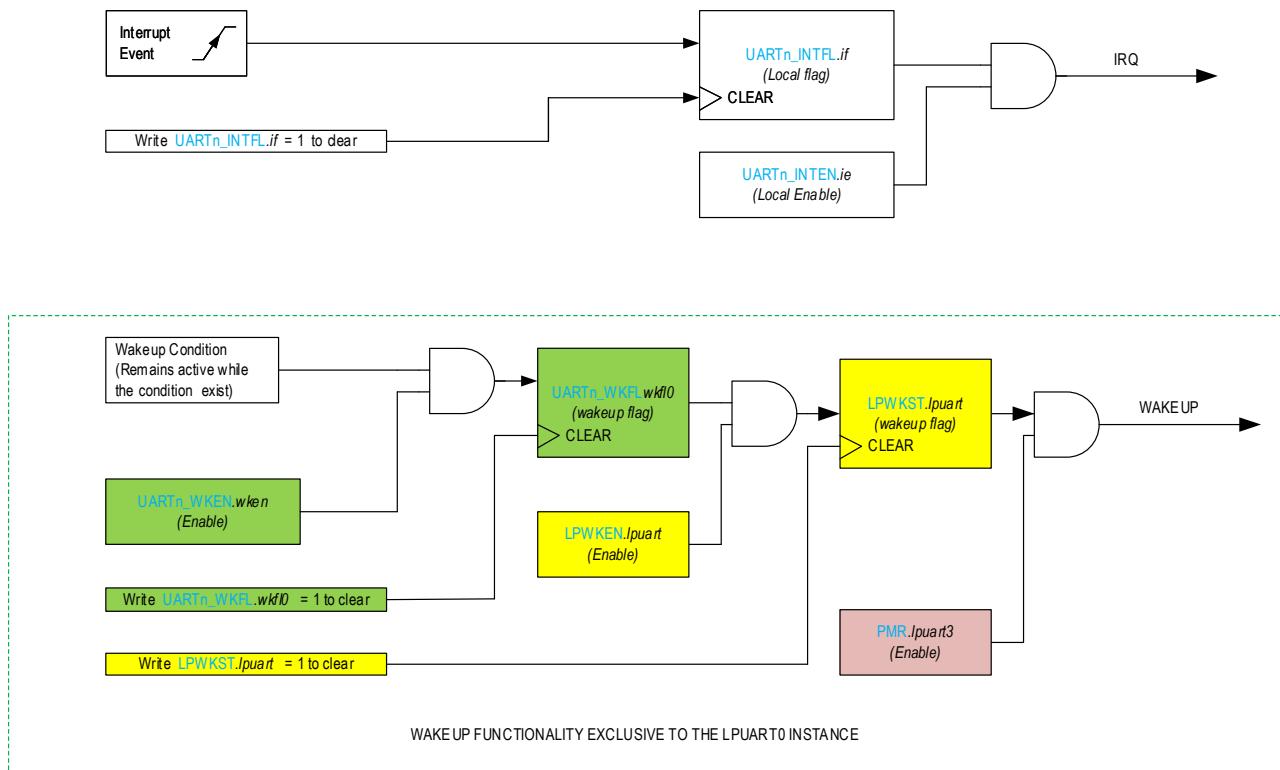
The FIFOs are flushed on the following conditions:

- Setting `UARTn_CTRL.rx_flush` flushes the RX FIFO by setting its pointer to 0.
- Setting `UARTn_CTRL.tx_flush` flushes the TX FIFO by setting its pointer to 0.
- Setting the corresponding field in the GCR_RST register causes a peripheral reset which flushes both the TX FIFO and RX FIFO by erasing their contents and setting their pointers to 0.

11.5 Interrupt Events

The peripheral generates interrupts for the events shown in [Figure 11-3: MAX78000 UART Interrupt and Wakeup Functional Diagram](#). Unless noted otherwise, each instance has its own independent set of interrupts, and higher-level flag and enable fields as shown in [Table 11-2: MAX78000 Interrupt Events](#).

Figure 11-3: MAX78000 UART Interrupt and Wakeup Functional Diagram



Some activity may cause more than one event, setting one or more event flags. An event interrupt will occur if the corresponding interrupt enable is set. The flags must be cleared by software in the ISR.

Table 11-2: MAX78000 Interrupt Events

| Event | Local Interrupt Flag | Local Interrupt Enable |
|--------------------|-------------------------------|-------------------------------|
| Frame Error | UARTn_INTFL.rx_ferr_if | UARTn_INTEN.rx_ferr_ie |
| Parity Error | UARTn_INTFL.rx_par_if | UARTn_INTEN.rx_par_ie |
| CTS Signal Change | UARTn_INTFL.cts_if | UARTn_INTEN.cts_ie |
| RX FIFO Overrun | UARTn_INTFL.rx_ov_if | UARTn_INTEN.rx_ov_ie |
| RX FIFO Threshold | UARTn_INTFL.rx_thd_if | UARTn_INTEN.rx_thd_ie |
| TX FIFO Half-Empty | UARTn_INTFL.tx_he_if | UARTn_INTEN.tx_he_ie |

11.5.1 Frame Error

A frame error is generated when the UART sampling circuitry detects an invalid bit. Each bit is sampled three times at the times described earlier and can generate frame errors on the state of the start, stop and optionally parity and data bits.

The frame error criteria are different depending on whether or not FDM is present in that instance and enabled or not as shown in [Table 11-3: Frame Error Detection \(FDM not present, or FDM = 0 and DPFE = 0\)](#) and [Table 11-4: Frame Error Detection \(FDM = 1 and DPFE = 1\)](#).

Table 11-3: Frame Error Detection (FDM not present, or FDM = 0 and DPFE = 0)

| UARTn_CTRL.par_en | UARTn_CTRL.par_md | UARTn_CTRL.par_eo | Start | Data | Parity | Stop |
|--------------------------|--------------------------|--------------------------|---------|------|-------------|-------|
| 0 | N/A | N/A | 3/3 = 0 | 2/3 | Not Present | 3/3=1 |

| <i>UARTn_CTRL</i> . <i>par_en</i> | <i>UARTn_CTRL</i> . <i>par_md</i> | <i>UARTn_CTRL</i> . <i>par_eo</i> | Start | Data | Parity | Stop |
|--------------------------------------|--------------------------------------|--------------------------------------|-------|------|---|------|
| 1 | 0 | 0 | | | 3/3 = 1 if even number "1" 3/3 = 0 if odd number "0" | |
| | 0 | 1 | | | 3/3 = 1 if odd number "1" 3/3 = 0 if even number "0" | |
| | 1 | 0 | | | 3/3 = 1 if even number "0" 3/3 = 0 if odd number "1" | |
| | 1 | 1 | | | 3/3 = 1 if odd number "0" 3/3 = 0 if even number "1" | |

Table 11-4: Frame Error Detection (FDM = 1 and DPFE = 1)

| <i>UARTn_CTRL</i> . <i>par_en</i> | <i>UARTn_CTRL</i> . <i>par_md</i> | <i>UARTn_CTRL</i> . <i>par_eo</i> | Start | Data | <i>PARITY</i> | <i>STOP</i> |
|--------------------------------------|--------------------------------------|--------------------------------------|-------|------|---|-------------|
| 0 | N/A | N/A | | | Not Present | 3/3=1 |
| 1 | 0 | 0 | | | 3/3 = 1 if even number "1" 3/3 = 0 if odd number "0" | |
| | 0 | 1 | | | 3/3 = 1 if odd number "1" 3/3 = 0 if even number "0" | |
| | 1 | 0 | | | 3/3 = 1 if even number "0" 3/3 = 0 if odd number "1" | |
| | 1 | 1 | | | 3/3 = 1 if odd number "0" 3/3 = 0 if even number "1" | |

11.5.2 Parity Error

Set *UARTn_CTRL.par_en* = 0 to enable parity checking of the received frame. If the calculated parity does not match the parity bit, then the corresponding interrupt flag is set.

11.5.3 CTS Signal Change

If hardware flow control is disabled, the CTS pin is a general-purpose input and the RTS pin is a general-purpose output. In that case, a CTS signal change event occurs on each rising or falling edge of the sampling on CTS input pin.

CTS sample value is reset to 1 after power-on reset. While the UART baud clock is running, CTS sampling process continues if *UARTn_CTRL.ctssd* is not 1. The sampled value will be set to 1 when the UART baud clock stops running because *UARTn_CTRL.bckrdy* has been cleared to 0 again.

11.5.4 Overrun

An overrun condition occurs if a valid frame is received when the RX FIFO is full. The interrupt flag is set at the end of the stop bit and the frame is discarded.

11.5.5 RX FIFO Threshold

An RX threshold event occurs when the valid frame is received that causes the number of bytes to exceed the configured threshold *UARTn_CTRL.rx_thd*.

11.5.6 TX FIFO One Byte Remaining

This event occurs when the TX FIFO transitions from *UARTn_STATUS.tx_num* = 2 to *UARTn_STATUS.tx_num* = 1.

11.5.7 TX FIFO Half-Empty

The TX FIFO Half-Empty event occurs when `UARTn_STATUS.tx_num` transitions from more than half-full to half empty as shown in [Equation 11-1](#).

Equation 11-1: UART TX FIFO Half-Empty Condition

$$\left(\frac{C_TX_FIFO_DEPTH}{2} + 1 \right) \xrightarrow{\text{Transitions from}} \left(\frac{C_TX_FIFO_DEPTH}{2} \right)$$

11.6 Wakeup Events

Instances which support fractional division mode can receive characters while in the low-power modes listed in [Table 11-1: MAX78000 UART/LPUART Instances](#). If enabled, any of the wakeup sources in [Table 11-5: MAX78000 Wakeup Events](#) will cause the device to exit the current low-power mode and return to ACTIVE mode.

Unlike interrupts, wakeup activity is based on a condition, not an event. As long as the condition is true and the local wakeup enable field is set to 1, the local wakeup flag will remain set.

Table 11-5: MAX78000 Wakeup Events

| Condition | Local Wakeup Flag <code>UARTn_WKFL</code> | Local Wakeup Enable <code>UARTn_WKEN</code> | Low Power Peripheral Wakeup Flag | Low Power Peripheral Wakeup Enable | Power Management Wakeup Enable |
|-------------------|--|--|--|------------------------------------|--|
| RX FIFO Threshold | <code>rx_thd_wf</code> | <code>rx_thd_we</code> | <code>PWRSEQ_LPPWST</code> <code>Ipuart0</code> | <code>PWRSEQ_Ipuart0</code> | <code>LPGCR_PCLKDIS</code> <code>Ipuarto</code> |
| RX FIFO Full | <code>rx_fu_wf</code> | <code>rx_fl_we</code> | | | |
| RX FIFO Not Empty | <code>rx_ne_wf</code> | <code>rx_ne_we</code> | | | |

11.6.1 RX FIFO Threshold

This condition persists while `UARTn_STATUS.rx_num` \geq `UARTn_CTRL.rx_thd`.

11.6.2 RX FIFO Full

This condition persists while `UARTn_STATUS.rx_num` ≥ 8 .

11.6.3 RX Not Empty

This condition persists while `UARTn_STATUS.rx_num` > 0 .

11.7 Resets

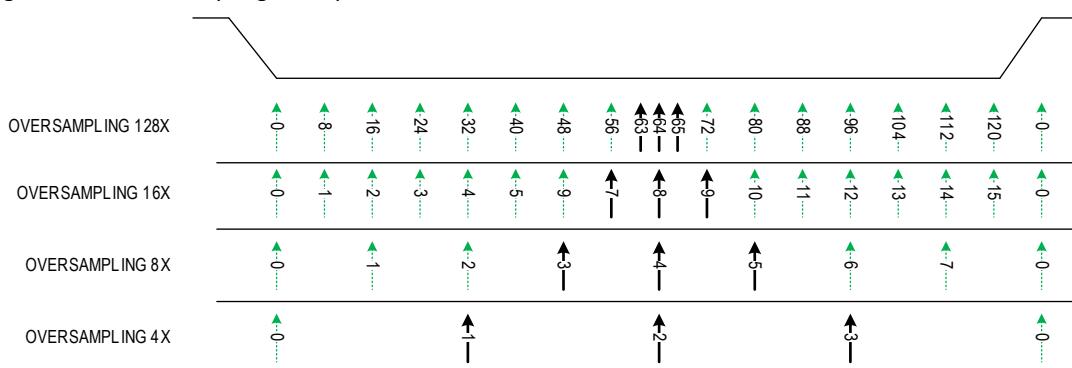
The following stimuli can reset the peripheral.

- The device will remain in reset while `UARTn_CTRL.bcken` = 0.
- The device will remain in reset while `UARTn_CTRL.bckrdy` = 0 after setting `UARTn_CTRL.bcken` to 1.
- Any write to `UARTn_CKDIV.clkdiv` while `UARTn_CTRL.bcken` = 1.
- Any write to `UARTn_OSR.osr` while `UARTn_CTRL.bcken` = 1.

11.8 RX Sampling

Each bit of a frame is oversampled to improve noise immunity. The oversampling rate (OSR) is configurable with the `UARTn_OSR.osr` field. In most cases the bit will be evaluated based on three samples at the midpoint of each bit time as shown in [Figure 11-4: Oversampling Example](#).

Figure 11-4: Oversampling Example



Whenever `UARTn_CLKDIV.clkdiv < 0x10` (i.e., division rate less than 8.0), OSR is not used and over sampling rate is adjusted to full sampling by hardware. That means RX is sampled in every clock cycle, despite of the value of OSR.

For 9600 baud low power operation, the dual-edge sampling mode must be enabled (`UARTn_CTRL.desm = 1`).

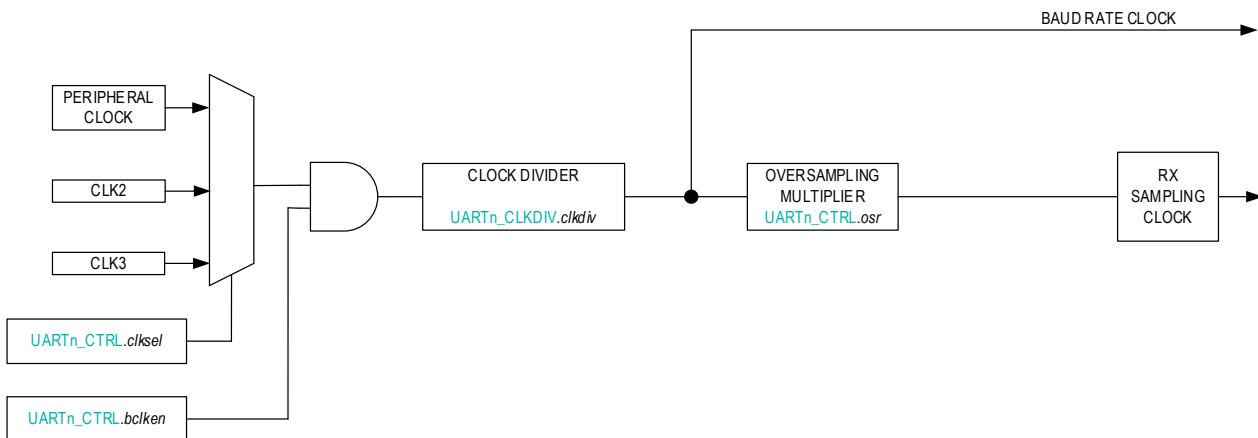
11.9 Baud Rate Generation

The baud rate is determined by the selected UART clock source and value of the clock divider circuit. Multiple clock sources are available for each instance and can be configured independently for each UART instance as shown in *Table 11-1: MAX78000 UART/LPUART Instances*.

11.9.1 Clock Sources (FDM not supported)

In UART instances the clock sources are disabled in all low power modes except SLEEP modes. In this case the UART is not running and cannot be used as a wakeup source.

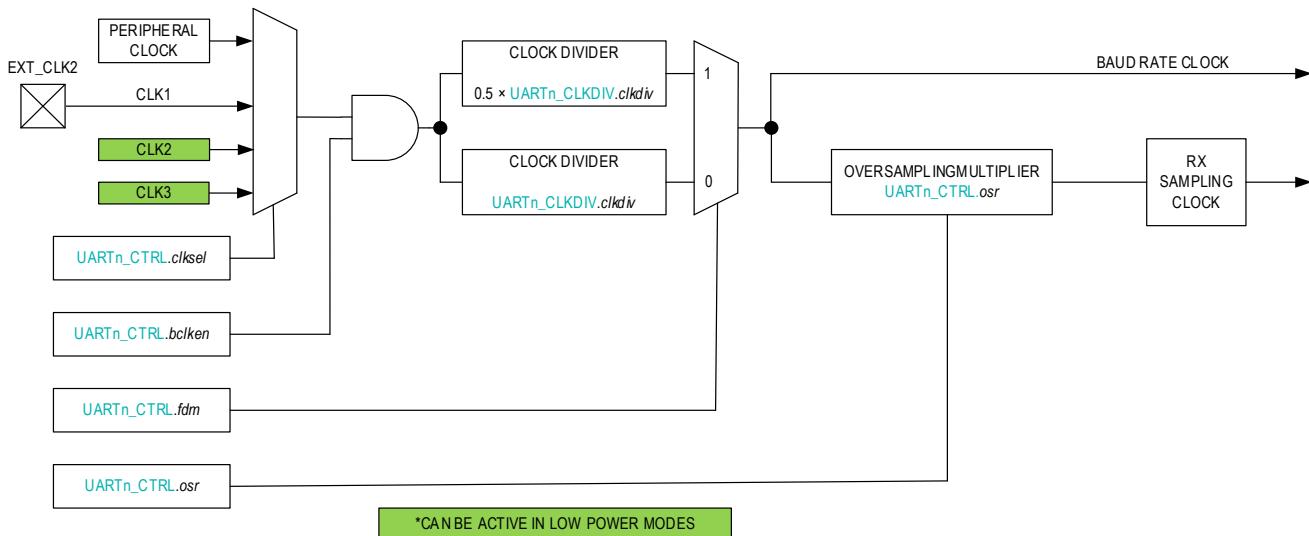
Figure 11-5: UART Timing Generation (FDM not supported)



11.9.2 Clock Sources (FDM supported)

Instances that support FDM can be configured to operate the LPUART receiver at 9600 baud using the 32kHz RTC as shown in *Table 11-1: MAX78000 UART/LPUART Instances*. This clock source can be configured to remain active in LPM, and μ PM modes, allowing the device to receive data and serve as a wakeup source, while power consumption is at a minimum.

Figure 11-6: LPUART Timing Generation (FDM supported)



Changing the clock source should only be done between data transfers to avoid corrupting an ongoing data transfer.

11.10 Baud Rate Calculation

The TX and RX circuits share a common baud rate clock, which is the selected UART clock source divided by the clock divisor. Instances that support FDM offer a 0.5 fractional clock division when enabled by setting **UARTn_CTRL.fdm** = 1. This allows for greater accuracy when operating at low baud rates, as well as finer granularity for the oversampling rate.

Use the following formula to calculate the **UARTn_CKDIV.clkdiv** value based on the clock source, and desired baud rate, and integer or fractional divisor.

Equation 11-2: UART Clock Divisor Formula for FDM = 0

UARTn_CTRL.fdm = 0:

$$\text{UARTn_CKDIV.clkdiv} = \text{INT}\left[\frac{\text{UART Clock}}{\text{Baud Rate}}\right]$$

Equation 11-3: Clock Divisor Formula for FDM =1

UARTn_CTRL.fdm = 1:

$$\text{UARTn_CKDIV.clkdiv} = \text{INT}\left[\frac{\text{UART Clock}}{\text{Baud Rate}} \times 2\right]$$

For example, in a case where the UART clock is 50MHz and target baud rate is 115,200 bps:

- When FDM=0, **UARTn_CKDIV.clkdiv** = $50,000,000 / 115,200 = 434$
- When FDM=1, **UARTn_CKDIV.clkdiv** = $(50,000,000 / 115,200) \times 2 = 434.03 \times 2 = 868$

11.11 Low-Power 9600 Baud Receiver Operation

Peripheral instances which support FDM have the option to configure the receiver for 9600 baud and keep it enabled in the low-power modes listed in *Table 11-1: MAX78000 UART/LPUART Instances*. Receipt of a valid frame will load the RX FIFO and increment **UARTn_STATUS.rx_num**. If enabled, any of the conditions in *Table 11-5: MAX78000 Wakeup Events* will cause the device to exit the current low-power mode and return to ACTIVE mode.

Instances that support FDM offer a 0.5 fractional clock division when enabled by setting **UARTn_CTRL.fdm** = 1. This allows for greater accuracy when operating at low baud rates, as well as finer granularity for the oversampling rate. But whenever

UARTn_CKDIV.clkdiv < 16 (division rate less than 8.0) the oversampling feature is not used and the RX signal is sampled in every clock cycle.

Use the following formula to calculate the *UARTn_CKDIV.clkdiv* value for low-baud rate operation when *UARTn_CTRL.fdm* = 1.

Equation 11-4: Low Baud Rate Operation Clock Divisor Formula for FDM =1

UARTn_CTRL.fdm = 1:

$$\text{UARTn_CKDIV.clkdiv} = \text{INT}\left[\frac{\text{UART clock}}{\text{Baud Rate}} \times 2\right]$$

Table 11-6: Slow Baud Rate Generation Example (FDM=1)

| Clock Source | BAUD | Ratio (Clock/BAUD) | Calculated <i>UARTn_CKDIV.clkdiv</i> | Error | <i>UARTn_OSR.osr</i> |
|--------------|-------------|--------------------|--------------------------------------|-------|---|
| 32,768Hz | 9,600 bit/s | 3.413 | 7 | -2.5% | N/A (1x) |
| | 7,200 bit/s | 4.551 | 9 | +1.1% | N/A (1x) |
| | 4,800 bit/s | 6.827 | 14 | -2.5% | N/A (1x) |
| | 2,400 bit/s | 13.653 | 27 | +1.1% | 0x0: 8x 1: 12x |
| | 1,800 bit/s | 18.204 | 36 | +1.1% | 0: 8x 1: 12x 2: 16x |
| | 1,200 bit/s | 27.307 | 54 | +1.1% | 0: 8x 1: 12x 2: 16x 3: 20x 4: 24x |

Changing the clock source should only be done between data transfers to avoid corrupting an ongoing data transfer.

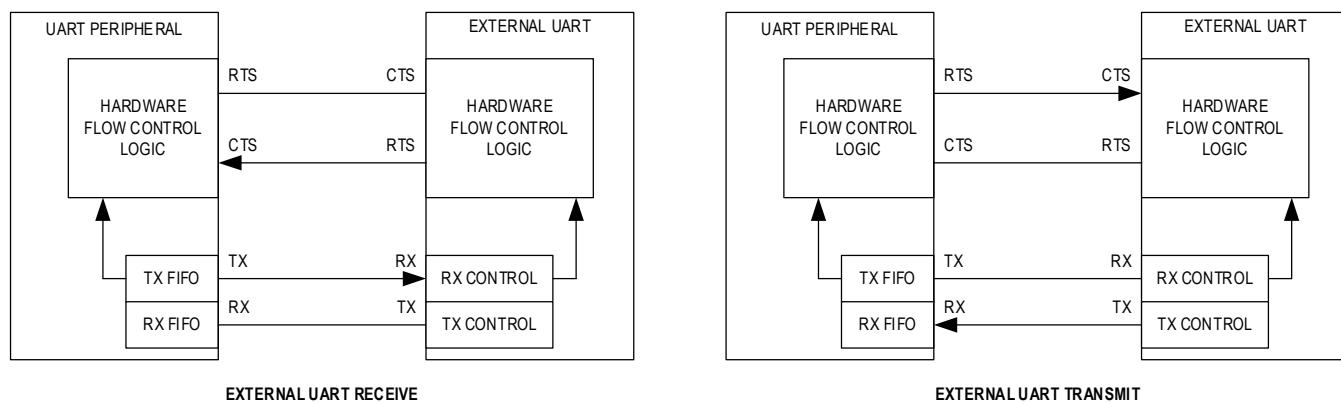
Use the following procedure to receive characters at 9600 baud while in low-power modes:

1. Clear *UARTn_CTRL.bcken* = 0 to disable the baud clock. Hardware will immediately clear *UARTn_CTRL.bckrdy* to 0.
2. Set *PWRSEQ_LPCN.x32ken*= 1 ensure the 32kHz clock source remains active in **LPM** and **UPM** modes.
3. Ensure *UARTn_CTRL.ucagm* = 1.
4. Configure *UARTn_CTRL.clksel* to the 32kHz clock source.
5. Set *UARTn_CTRL.fdm* = 1.
6. Set *UARTn_CKDIV.clkdiv* = 7.
7. Set *UARTn_CTRL.desm* = 1.
8. Choose the desired wakeup conditions from *Table 11-5: MAX78000 Wakeup Events* and set the corresponding wakeup enable fields to 1. Ensure the conditions are not active before setting the wakeup enable fields.
9. Set *UARTn_CTRL.bcken* = 1 to re-enable the baud clock.
10. Poll until hardware sets *UARTn_CTRL.bckrdy* = 1.
11. Enter the desired low power mode.

11.12 Hardware Flow Control

The optional hardware flow control (HFC) uses two pins, CTS (Clear-to-send) and RTS (Request-to-Send), as a handshaking protocol to manage UART communications. For full duplex operation, the RTS output pin on the peripheral is connected to the CTS input pin on the external UART, and the CTS input pin on the peripheral is connected to the RTS output pin on the external UART as shown in *Figure 11-7. Hardware Flow Control, Physical Connection*.

Figure 11-7. Hardware Flow Control, Physical Connection



In HFC operation, a UART transmitter that wants to transmit data waits for the other device to assert its CTS input pin. If CTS is asserted, then the UART begins transmitting data from its TX FIFO to the slave RX FIFO. The RX FIFO will keep CTS asserted until the RX FIFO fills to the specified value. It will then deassert CTS until the receiver has cleared the buffer below the specified value. CTS will then be asserted again, and more data sent.

Hardware flow control can be fully automated by the peripheral, or by software through monitoring of the CTS input signal.

11.12.1 Automated HFC

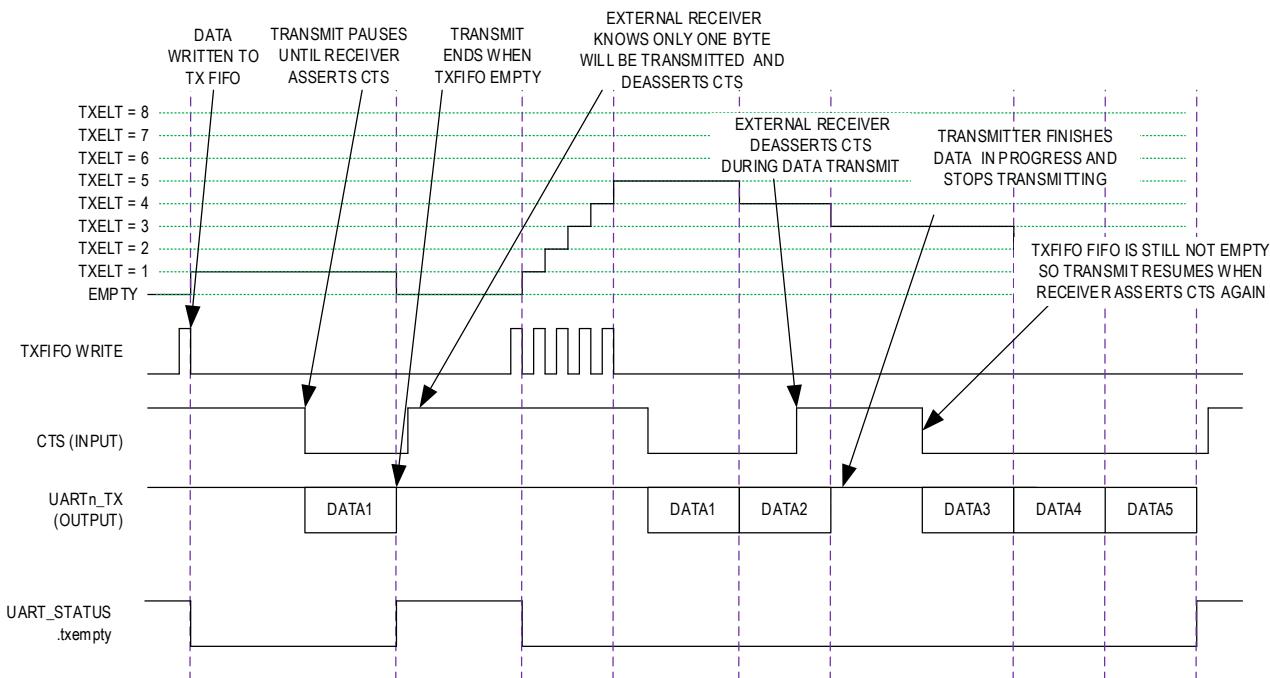
Setting `UARTn_CTRL.flow_ctrl = 1` enables hardware flow control. The CTS and RTS external signals are directly managed by hardware without CPU intervention. The assertion of the RTS to CTS signal is dependent on the `UARTn_CTRL.rtslegacy` field:

- `UARTn_CTRL.rtslegacy = 0`: Deassert RTS while `UARTn_STATUS.rx_num = C_RX_FIFO_DEPTH`
- `UARTn_CTRL.rtslegacy = 1`: Deassert RTS while `UARTn_STATUS.rx_num ≥= UARTn_CTRL.rx_thd`

The transmitter will continue to send data as long as the CTS signal is asserted and there is data in the TX FIFO. If the receiver deasserts the transmitter's CTS, the transmitter will finish transmission of the current character and then wait until CTS is asserted again. *Figure 11-8. Hardware Flow Control Signaling, Transmit to External Receiver* shows the state of the CTS pin during a transmission under hardware flow control.

HFC by itself does not generate interrupt events. FIFO management must be handled by the software, using the user-configurable interrupt event enables.

Figure 11-8. Hardware Flow Control Signaling, Transmit to External Receiver



11.12.2 Software-Controlled HFC

If HFC is disabled (`UARTn_CTRL.flow_ctrl = 1`), the CTS pin is a general-purpose input and the RTS pin is a general-purpose output. In that case, a signal change interrupt is raised (if enabled) at each rising or falling edge of the sampling on CTS input pin. The CTS signal change interrupt should be disabled if flow control is not used.

`UARTn_CTRL.ctssd` is reset to 1 after power-on reset. When UART baud clock starts and runs, CTS sampling process continues if `UARTn_CTRL.ctssd` is not 1. The sampled value will be set to 1 again when UART baud clock stops running because `UARTn_CTRL.bcken` has been cleared to 0 again.

11.13 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 11-7: UART/LPUART Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register `PERIPHERALn_CTRL` resolves to `PERIPHERAL0_CTRL` and `PERIPHERAL1_CTRL` for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

All registers and fields apply to both UART and LPUART instances unless specified otherwise.

Table 11-7: UART/LPUART Register Summary

| Offset | Register | Name |
|----------|---------------------------|--------------------------------|
| [0x0000] | <code>UARTn_CTRL</code> | UART Control Register |
| [0x0004] | <code>UARTn_STATUS</code> | UART Status Register |
| [0x0008] | <code>UARTn_INTEN</code> | UART Interrupt Enable Register |

| Offset | Register | Name |
|----------|---------------------|---------------------------------------|
| [0x000C] | <i>UARTn_INTFL</i> | UART Interrupt Flag Register |
| [0x0010] | <i>UARTn_CKDIV</i> | UART Clock Divisor Register |
| [0x0014] | <i>UARTn_OSR</i> | UART Oversampling Control Register |
| [0x0018] | <i>UARTn_TXFIFO</i> | UART Transmit FIFO |
| [0x0020] | <i>UARTn_PNR</i> | UART Pin Control Register |
| [0x0030] | <i>UARTn_DATA</i> | UART FIFO Data Register |
| [0x0034] | <i>UARTn_DMA</i> | UART DMA Control Register |
| [0x002C] | <i>UARTn_WKEN</i> | UART Wakeup Interrupt Enable Register |
| [0x0030] | <i>UARTn_WKFL</i> | UART Wakeup Interrupt Flag Register |

11.13.1 Register Details

Table 11-8: UART Control Register

| UART Control | | | UARTn_CTRL | | [0x0000] |
|--------------|--------|--------|------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:23 | - | DNM | 0 | Reserved. Do Not Modify. | |
| 22 | desm | R/W | 0 | RX Dual Edge Sampling Mode 0: Sample RX on clock rising only 1: Sample RX on both rising and falling edges This field is undefined if the instance does not support FDM. | |
| 21 | fdm | R/W | 0 | Fractional Division Mode 0: Baud rate divisor is an integer 1: Baud rate divisor supports 0.5 division resolution. This field is undefined if the instance does not support FDM. | |
| 20 | ucagm | R/W | 0 | UART Clock Auto Gating Mode Software must always set this field to 1 for proper operation. 0: No gating 1: UART clock is paused during TX/RX idle states | |
| 19 | bckrdy | R | 0 | Baud Clock Ready 0: Baud clock not ready 1: Baud clock ready | |
| 18 | dpfee | R/W | 0 | Data/Parity bit frame error detection enable 0: Disable. Do not detect frame errors on RX between start bit and stop bit. 1: Enable. Detect frame errors when RX changes at the center of bit time. This field is undefined if the instance does not support FDM. | |
| 17:16 | clksel | R/W | 0 | Baud Clock Source Selects the baud clock source. See Table 11-1: MAX78000 UART/LPUART Instances for specific values for each instance. 0: Peripheral Clock 1: External clock 2: CLK2 3: CLK3 | |
| 15 | bcken | R/W | 0 | Baud Clock Enable 0: Disabled 1: Enabled | |

| UART Control | | | UARTn_CTRL | | [0x0000] |
|--------------|-----------------|--------|------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 14 | rtslegacy | R | 0 | Hardware Flow Control RTS Deassert Condition 0: Deassert RTS when RX FIFO Level = C_RX_FIFO_DEPTH (FIFO full) 1: Deassert RTS while RX FIFO Level >= <i>UARTn_CTRL.rx_thd</i> | |
| 13 | flow_ctrl | R/W | 0 | Hardware Flow Control Enable 0: Disabled 1: Enabled | |
| 12 | stopbits | R/W | 0 | Number of Stop Bits 0: 1 stop bit 1: 1.5 stop bits (for 5 bit mode) or 2 stop bits (for 6/7/8 bit mode) | |
| 11:10 | char_size | R/W | 0 | Character Length 0: 5 bits 1: 6 bits 2: 7 bits 3: 8 bits | |
| 9 | rx_flush | W1 | 0 | RX FIFO Flush Write 1 to flush the FIFO. This bit will always read 0. 0: N/A 1: Flush FIFO | |
| 8 | tx_flush | W1 | 0 | TX FIFO Flush Write 1 to flush the FIFO. This bit will always read 0. 0: N/A 1: Flush FIFO | |
| 7 | cts_samples_dis | R/W | 0 | CTS Sampling Disable 0: Enabled 1: Disabled | |
| 6 | par_md | R/W | 0 | Parity Value Select 0: Parity calculation is based on "1" bits. (Mark) 1: Parity calculation is based on "0" bits. (Space) | |
| 5 | par_eo | R/W | 0 | Parity Odd/Even Select 0: Even parity 1: Odd parity | |
| 4 | par_en | R/W | 0 | TX Parity Generation Enable 0: Parity transmission disabled. 1: Parity bit is calculated and transmitted after the last character bit. | |
| 3:0 | rx_thd | R/W | 0 | Receive FIFO Threshold Valid settings are from 0x01 to (C_RX_FIFO_DEPTH – 1). 0: Reserved 1: 1 2: 2 3: 3 4: 4 5: 5 6: 6 7: 7 8: 8 9 - 15: Reserved for Future Use | |

Table 11-9: UART Status Register

| UART Status | | | UARTn_STATUS | | [0x0004] |
|-------------|----------|--------|--------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved Do not modify. | |
| 15:12 | tx_num | RO | 0 | TX FIFO Level Number of characters in the TX FIFO. 0: 0 1: 1 2: 2 3: 3 4: 4 5: 5 6: 6 7: 7 8: 8 9 - 15: Reserved for Future Use | |
| 11:8 | rx_num | RO | 0 | RX FIFO Level Number of characters in the RX FIFO. 0: 0 1: 1 2: 2 3: 3 4: 4 5: 5 6: 6 7: 7 8: 8 9 - 15: Reserved for Future Use | |
| 7 | tx_fu_st | RO | 0 | Transmit FIFO Full 0: Not full 1: Full | |
| 6 | tx_em_st | RO | 1 | Transmit FIFO Empty 0: Not empty 1: Empty | |
| 5 | rx_fu_st | RO | 0 | Receive FIFO Full 0: Not full 1: Full | |
| 4 | rx_em_st | RO | 1 | Receive FIFO Empty 0: Not empty 1: Empty | |
| 3:2 | - | RO | 0 | Reserved Do not modify. | |
| 1 | rx_busy | RO | 0 | Receive Busy 0: UART is not receiving a character 1: UART is receiving a character | |
| 0 | tx_busy | RO | 0 | Transmit Busy 0: UART is not transmitting data 1: UART is transmitting data | |

Table 11-10: UART Interrupt Enable Register

| UART Interrupt Enable | | | UARTn_INTEN | | [0x0008] |
|-----------------------|------------|--------|-------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:7 | - | RO | 0 | Reserved Do not modify. | |
| 6 | tx_he_ie | R/W | 0 | TX FIFO Half-Empty Event Interrupt Enable 0: Disabled 1: Enabled | |
| 5 | - | RO | 0 | Reserved Do not modify. | |
| 4 | rx_thd_ie | R/W | 0 | RX FIFO Threshold Event Interrupt Enable 0: Disabled 1: Enabled | |
| 3 | rx_ov_ie | R/W | 0 | RX FIFO Overrun Event Interrupt Enable 0: Disabled 1: Enabled | |
| 2 | cts_ie | R/W | 0 | CTS Signal Change Event Interrupt Enable 0: Disabled 1: Enabled | |
| 1 | rx_par_ie | R/W | 0 | RX Parity Event Interrupt Enable 0: Disabled 1: Enabled | |
| 0 | rx_ferr_ie | R/W | 0 | RX Frame Error Event Interrupt Enable 0: Disabled 1: Enabled | |

Table 11-11: UART Interrupt Flag Register

| UART Interrupt Flag | | | UARTn_INTFL | | [0x000C] |
|---------------------|-----------|--------|-------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:7 | - | RO | 0 | Reserved Do not modify. | |
| 6 | tx_he_if | R/W1C | 0 | TX FIFO Half-Empty Event Interrupt Flag 0: Disabled 1: Enabled | |
| 5 | - | RO | 0 | Reserved Do not modify. | |
| 4 | rx_thd_if | R/W1C | 0 | RX FIFO Threshold Event Interrupt Flag 0: Disabled 1: Enabled | |
| 3 | rx_ov_if | R/W1C | 0 | RX FIFO Overrun Event Interrupt Flag 0: Disabled 1: Enabled | |
| 2 | cts_if | R/W1C | 0 | CTS Signal Change Event Interrupt Flag 0: Disabled 1: Enabled | |
| 1 | rx_par_if | R/W1C | 0 | RX Parity Error Event Interrupt Flag 0: Disabled 1: Enabled | |

| UART Interrupt Flag | | | | UARTn_INTFL | [0x000C] |
|---------------------|------------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 0 | rx_ferr_if | R/W1C | 0 | RX Frame Error Event Interrupt Flag 0: Disabled 1: Enabled | |

Table 11-12: UART Clock Divisor Register

| UART Clock Divisor | | | | UARTn_CKDIV | [0x0010] |
|--------------------|--------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:20 | - | RO | 0 | Reserved Do not modify. | |
| 19:0 | clkdiv | R/W | 0 | Baud Rate Divisor This field sets the divisor used to generate the baud tick from the baud clock. If UARTn_CTRL.fdm = 1 the fractional divisors are in increments of 0.5. The over sampling rate must be no greater than this divisor. Refer to section Baud Rate Generation for information on how to use this field. | |

Table 11-13: UART Oversampling Control Register

| UART Oversampling Control | | | | UARTn_OSRR | [0x0014] |
|---------------------------|------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:3 | - | RO | 0 | Reserved Do not modify. | |
| 2:0 | osr | R/W | 0 | Over Sampling Rate In Fractional Division Mode (FDM=1), 0: 8 × 1: 12 × 2: 16 × 3: 20 × 4: 24 × 5: 28 × 6: 32 × 7: 36 × Not in Fractional Division Mode (FDM=0) 0: 128 × 1: 64 × 2: 32 × 3: 16 × 4: 8 × 5: 4 × 6 - 7: Reserved for Future Use | |

Table 11-14: UART Transmit FIFO Register

| UART Transmit FIFO | | | | UARTn_TXFIFO | [0x0018] |
|--------------------|------|--------|-------|-----------------------------------|----------|
| Bits | Name | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved Do not modify. | |

| UART Transmit FIFO | | | | UARTn_TXFIFO | [0x0018] |
|--------------------|------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 7:0 | data | RO | 0 | Transmit FIFO Data Read the TX FIFO next data. Reading from this field does not affect the contents of the TX FIFO. <i>Note: The parity bit is available from this field.</i> Reading from this field returns the next character available at the output of the TX FIFO (if any data is available, otherwise this field reads 0). | |

Table 11-15: UART Pin Control Register

| UART Pin Control | | | | UARTn_PNR | [0x001C] |
|------------------|------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:2 | - | RO | 0 | Reserved Do not modify | |
| 1 | rts | R/W | 1 | RTS Pin Output State 0: RTS signal is driven to 0 1: RTS signal is driven to 1 | |
| 0 | cts | RO | 1 | CTS Pin State Returns the current sampled state of the GPIO associated with the CTS signal. 0: CTS state is 0 1: CTS state is 1 | |

Table 11-16: UART Data Register

| UART Data | | | | UARTn_DATA | [0x0020] |
|-----------|-----------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:9 | - | RO | 0 | Reserved Do not modify. | |
| 8 | rx_parity | R | 0 | Receive FIFO Byte Parity If parity feature is disabled, this bit always read 0. If a parity error occurred during the reception of the character at the output end of the RX FIFO (that would be returned by reading the <i>UARTn_DATA.data</i> field), this bit will read 1, otherwise it will read 0. | |
| 7:0 | data | R/W | 0 | Transmit/Receive FIFO Value Writing to this field loads the next character into the TX FIFO (if space is available). Reading from this field returns the next character available at the output of the RX FIFO (if one is available, otherwise 0 is returned). For 5/6/7-bit width characters, the unused bit(s) are ignored when the TX FIFO is loaded, and the unused high bit(s) read 0 on characters read from the RX FIFO | |

Table 11-17: UART DMA Register

| UART DMA | | | | UARTn_DMA | [0x0030] |
|----------|-----------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:10 | - | RO | 0 | Reserved Do not modify. | |
| 9 | dma_rx_en | 0 | 0 | RX DMA Channel Enable 0: Disabled 1: Enabled | |

| UART DMA | | | UARTn_DMA | | [0x0030] |
|----------|------------|--------|-----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 8:5 | dma_rx_thd | 0 | 0 | RX FIFO Level DMA Threshold If <i>UARTn_STATUS.rx_num < UARTn_DMA.dma_rx_thd</i> , then the RX FIFO DMA interface will send a signal to the system DMA to notify that RX FIFO has characters to transfer to memory | |
| 4 | dma_tx_en | R/W | 0 | TX DMA Channel Enable 0: Disabled 1: Enabled | |
| 3:0 | dma_tx_thd | R/W | 0 | TX FIFO Level DMA Threshold If <i>UARTn_STATUS.tx_num < UARTn_DMA.dma_tx_thd</i> , then the TX FIFO DMA interface will send a signal to system DMA to notify that TX FIFO is ready to receive data from memory. | |

Table 11-18: UART Wakeup Enable

| UART Wakeup Enable | | | UARTn_WKEN | | [0x0034] |
|--------------------|-----------|--------|------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:3 | - | RO | 0 | Reserved Do not modify. | |
| 2 | rx_thd_we | R/W | 0 | RX FIFO Threshold Wake-up Event Enable 0: Disabled 1: Enabled | |
| 1 | rx_fu_we | R/W | 0 | RX FIFO Full Wake-Up Event Enable 0: Disabled 1: Enabled | |
| 0 | rx_ne_we | R/W | 0 | RX FIFO Not Empty Wake-Up Event Enable 0: Disabled 1: Enabled | |

Table 11-19. UART Wakeup Flag Register

| UART Wakeup Flag | | | UARTn_WKFL | | [0x0038] |
|------------------|-----------|--------|------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:3 | - | RO | 0 | Reserved Do not modify. | |
| 2 | rx_thd_wf | R/W | 0 | RX FIFO Threshold Wake-up Event 0: Disabled 1: Enabled | |
| 1 | rx_fu_wf | R/W | 0 | RX FIFO Full Wake-Up Event 0: Disabled 1: Enabled | |
| 0 | rx_ne_wf | R/W | 0 | RX FIFO Not Empty Wake-Up Event 0: Disabled 1: Enabled | |

12. Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) is a highly configurable, flexible, and efficient synchronous interface between multiple SPI devices on a single bus. The SPI bus uses a single clock signal, and a single, dual or quad data lines, and one or more slave select lines for communication with external SPI devices.

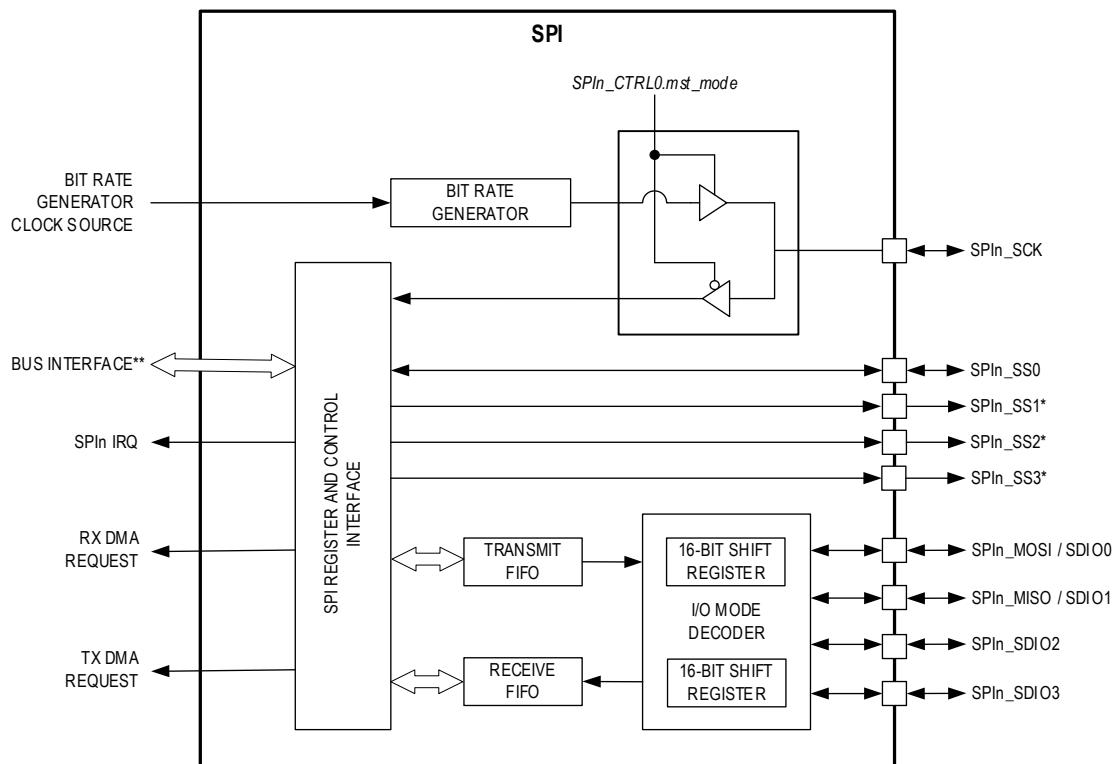
The provided SPI ports support full-duplex, bi-direction I/O and each SPI includes a Bit Rate Generator (BRG) for generating the clock signal when operating in master mode. Each SPI port operates independently and requires minimal processor overhead. All instances of the SPI peripheral support both master and slave modes and support single master and multi-master networks.

Features include:

- Dedicated Bit Rate Generator for precision serial clock generation in Master Mode
 - ◆ Up to $\frac{f_{PCLK}}{2}$ for instances on the APB bus
 - ◆ Up to $\frac{f_{HCLK}}{2}$ for instances on the AHB bus
 - ◆ Programmable SCK duty cycle timing
- Full-duplex, synchronous communication of 2 to 16-bit characters
 - ◆ 1-bit and 9-bit characters are not supported
 - ◆ 2-bit and 10-bit characters do not support maximum clock speed. *SPIn_CLKCTRL.clkdiv* must be > 0
- 3-wire and 4-wire SPI operation for single-bit communication
- Single, Dual or Quad I/O operation
- Byte-wide Transmit and Receive FIFOs with 32-byte depth
 - ◆ For character sizes greater than 8, each character uses 2 entries per character resulting in 16 entries for the Transmit and Receive FIFO
- Transmit and Receive DMA support
- SPI modes 0, 1, 2, 3
- Configurable slave select lines
 - ◆ Programmable slave select level
- Programmable slave select timing with respect to SCK starting edge and ending edge
- Multi-master mode fault detection

Figure 12-1: SPI Block Diagram shows the structure of the peripheral. See *Table 12-1: MAX78000 SPI Instances* for the peripheral-specific peripheral bus assignment and bit rate generator clock source.

Figure 12-1: SPI Block Diagram



* The number of slave select signals can vary for each instance of the peripheral.

** The bus interface (APB or AHB) can vary for each instance of the peripheral.

12.1 Instances

There are two instances of the SPI peripheral as shown in *Table 12-1: MAX78000 SPI Instances*. *Table 12-2: MAX78000 SPI Peripheral Pins* lists the locations of the SPI signals for each of the SPI instances.

Table 12-1: MAX78000 SPI Instances

| Instance | Formats | | | | Hardware Bus | Bit Rate Generator Clock Source Frequency | Slave Select Signals | |
|----------|---------|--------|------|------|--------------|---|----------------------|--|
| | 3-Wire | 4-Wire | Dual | Quad | | | 81-CTBGA | |
| SPI0 | Yes | Yes | Yes | Yes | AHB | f_{SYS_CLK} | 3 | |
| SPI1 | Yes | Yes | Yes | Yes | APB | f_{PCLK} | 1 | |

Note: Refer to the MAX78000 Datasheet for the definitive list of alternate function assignments for each peripheral.

Table 12-2: MAX78000 SPI Peripheral Pins

| Instance | Signal Description | Alternate Function | Alternate Function Number | 81 CTBGA |
|----------|--------------------|--------------------|---------------------------|----------|
| SPI0 | SPI Clock | SPI0_SCK | AF1 | P0.7 |
| | Slave Select 0 | SPI0_SS0 | AF1 | P0.4 |
| | Slave Select 1 | SPI0_SS1 | AF2 | P0.11 |
| | Slave Select 2 | SPI0_SS2 | AF2 | P0.10 |
| | MOSI (SDIO0) | SPI0_MOSI | AF1 | P0.5 |
| | MISO (SDIO1) | SPI0_MISO | AF1 | P0.6 |
| | SDIO2 | SPI0_SDIO2 | AF1 | P0.8 |
| | SDIO3 | SPI0_SDIO3 | AF1 | P0.9 |
| SPI1 | SPI Clock | SPI1_SCK | AF1 | P0.23 |
| | Slave Select 0 | SPI1_SS0 | AF1 | P0.20 |
| | MOSI (SDIO0) | SPI1_MOSI | AF1 | P0.21 |
| | MISO (SDIO1) | SPI1_MISO | AF1 | P0.22 |
| | SDIO2 | SPI1_SDIO2 | AF1 | P0.24 |
| | SDIO3 | SPI1_SDIO3 | AF1 | P0.25 |

12.2 SPI Formats

12.2.1 Four-Wire SPI

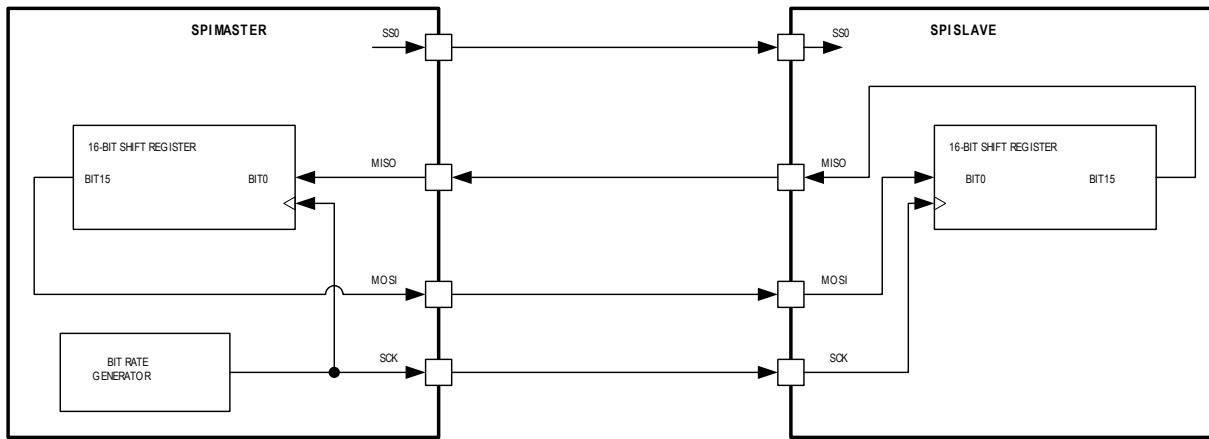
SPI devices operate as either a master or slave device. In four-wire SPI, four signals are required for communication as shown in *Table 12-3*.

Table 12-3: Four-Wire Format Signals

| Signal | Description | Direction |
|--------|---------------------------|---|
| SCK | Serial Clock | The master generates the Serial Clock signal, which is an output from the master and an input to the slave. |
| MOSI | Master Output Slave Input | In master mode, this signal is used as an output for sending data to the slave. In slave mode this is the input data from the master. |
| MISO | Master Input Slave Output | In master mode, this signal is used as an input for receiving data from the slave. In slave mode, this signal is an output for transmitting data to the master. |
| SS | Slave Select | In master mode, this signal is an output used to select a slave device prior to communication. Peripherals may have multiple slave select outputs to communicate with one or more external slave devices. In slave mode SPI _n _SS0 is a dedicated input which indicates an external master is going to start communication. Other slave select signals into the peripheral are ignored in slave mode. |

In a typical SPI network, the master device selects the slave device using the slave select output. The master starts the communication by selecting the slave device by asserting the slave select output. The master then starts the SPI clock via the SCK output pin. When a slave device's slave select pin is deasserted, the device is required to put the SPI pins in tri-state mode.

Figure 12-2. 4-Wire SPI Connection Diagram



12.2.2 Three-Wire SPI

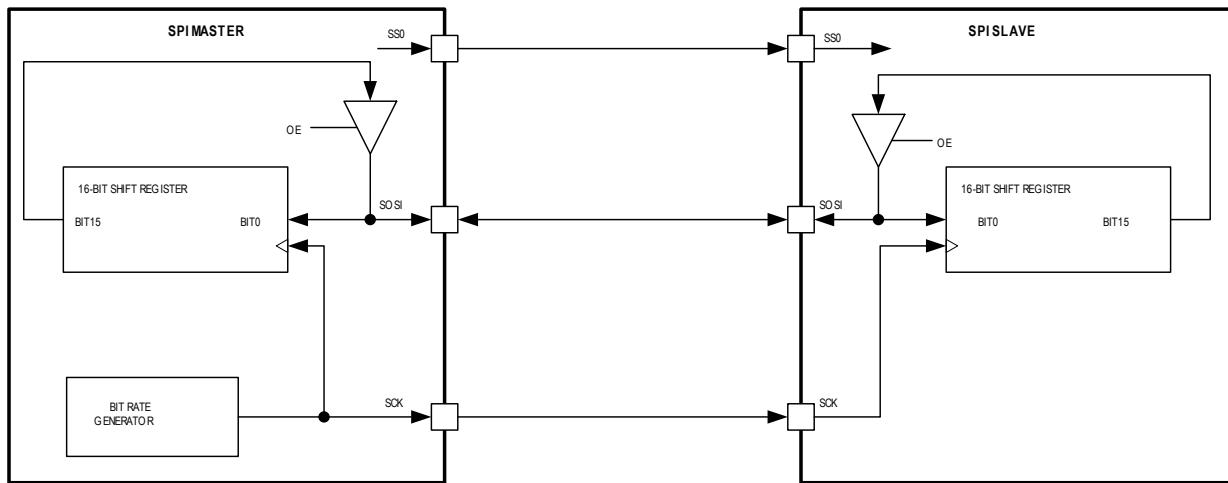
The signals in three-wire SPI operation are shown in *Table 12-4: Three-Wire Format Signals*. The MOSI signal is used as a bi-directional, half-duplex I/O referred to as Slave Input Slave Output (SISO). Three-wire SPI also uses a serial clock signal generated by the master and a slave select pin controlled by the master.

Table 12-4: Three-Wire Format Signals

| Signal | Description | Direction |
|--------|---------------------------|---|
| SCK | Serial Clock | The master generates the serial clock signal, which is an output from the master and an input to the slave. |
| MOSI | Master Output Slave Input | This is a half-duplex, bidirectional I/O pin used for communication between the SPI master and slave. This signal is used to transmit data from the master to the slave and to receive data from the slave by the master. |
| SS | Slave Select | In master mode, this signal is an output used to select a slave device prior to communication. In slave mode SPI _n _SS _O is a dedicated input which indicates an external master is going to start communication. Other slave select signals into the peripheral are ignored in slave mode |

A three-wire SPI network is shown in *Figure 12-3*. The master device selects the slave device using the slave select output. The communication starts with the master asserting the slave select line and then starting the clock (SCK). In three-wire SPI communication, the master and slave must both know the intended direction of the data to prevent bus contention. For a write, the master drives the data out the SISO pin. For a read, the master must release the SISO line and let the slave drive the SISO line. The direction of transmission is controlled using the FIFO. Writing to the FIFO starts the three-wire SPI write and reading from the FIFO starts a three-wire SPI read transaction.

Figure 12-3: Generic 3-Wire SPI Master to Slave Connection



12.3 Pin Configuration

Before configuring the SPI peripheral, first disable any SPI activity for the port by clearing the `SPIn_CTRL0.en` field to 0.

12.3.1 SPIn Alternate Function Mapping

Pin selection and configuration is required to use the SPI port. The following information applies to SPI master and slave operation as well as three-wire, four-wire, dual and quad mode communications. Determine the pins required for the SPI type and mode in the application and configure the required GPIO as described in the following sections. Refer to the MAX78000 data sheet for pin availability for a specific package.

When the SPI port is disabled, `SPIn_CTRL0.en` = 0, the GPIO pins enabled for SPI alternate function are placed in high-impedance input mode.

12.3.2 Four-wire Format Configuration

Four wire SPI uses SCK, MISO, MOSI, and one or more SS pins. Four wire SPI may use more than one slave select pin for a transaction, resulting in more than four wires total, however the communication is referred to as four wire for legacy reasons.

Note: Select the pins mapped to the SPI external device in the design and modify the setup accordingly. There is no restriction on which alternate function is used for a specific SPI pin and each SPI pin can be used independently from the other pins chosen. However, it is recommended that only one set of GPIO port pins be used for any network.

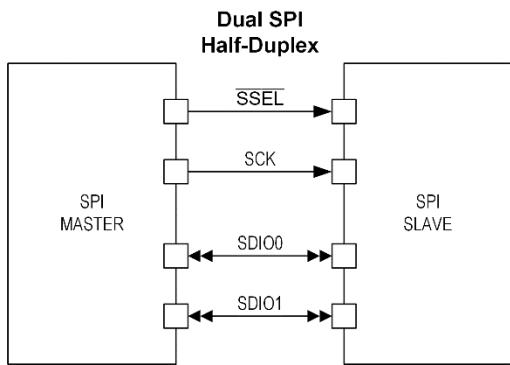
12.3.3 Three-Wire Format Configuration

Three wire SPI uses SCK and MOSI, and one or more slave select pins for an SPI transaction. Three-wire SPI configuration is identical to the four wire configuration except SPIn_MISO does not need to be set up for the SPI alternate function. The direction of communication in three-wire SPI mode is controlled by the SPIn Transmit and Receive FIFO enables. Enabling the Receive FIFO and disabling the Transmit FIFO indicates a read transaction. Enabling the Transmit FIFO and disabling the Receive FIFO indicates a write transaction. It is an illegal condition to enable both the Transmit and Receive FIFOs in three wire SPI operation.

12.3.4 Dual Mode Format Configuration

In Dual mode SPI two I/O pins are used to transmit 2-bits of data per SCK clock cycle. The communication is half-duplex and the direction of the data transmission must be known by both the master and slave for a given transaction. Dual mode SPI uses SCK, SDIO0, SDIO1 and one or more slave select lines as shown in [Figure 12-4](#). The configuration of the GPIO pins for Dual mode SPI is identical to four wire SPI and the mode is controlled by setting [*SPIIn_CTRL2.data_width*](#) to 1 indicating to the SPIn hardware to use SDIO0 and SDIO1 for half-duplex communication rather than full-duplex communication.

Figure 12-4: Dual Mode SPI Connection Diagram



12.3.5 Quad Mode Format Pin Configuration

Quad mode SPI uses four I/O pins to transmit four-bits of data per transaction. In Quad mode SPI, the communication is half-duplex and the master and slave must know the direction of transmission for each transaction. Quad mode SPI uses SCK, SDIO0, SDIO1, SDIO2, SDIO3 and one or more slave select pins.

Quad mode SPI transmits four bits per SCK cycle. Selection of Quad mode SPI is selected by setting [*SPIIn_CTRL2.data_width*](#) to 2.

12.4 SPI Clock Configuration

12.4.1 Serial Clock

The SCK signal synchronizes data movement in and out of the device. The master drives SCK as an output to the slave's SCK pin. When SPIn is set to master mode, the SPIn bit rate generator creates the serial clock and outputs it on the configured SPIn_SCK pin. When SPIn is configured for slave operation the SPIn_SCK pin is an input from the external master and the SPIn hardware synchronizes communications using the SCK input. Operating as a slave, if a SPIn slave select input is not asserted, the SPIn ignores any signals on the serial clock and serial data lines.

In both master and slave devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. Data availability and sampling time is controlled using the SPI phase control field, [*SPIIn_CTRL2.clkpha*](#). The SCK clock polarity field, [*SPIIn_CTRL2.clkpol*](#), controls if the SCK signal is active high or active low.

The SPIn peripheral supports four combinations of SCK phase and polarity referred to as SPI Modes 0, 1, 2, and 3. Clock Polarity ([*SPIIn_CTRL2.clkpol*](#)) selects an active low/high clock and has no effect on the transfer format. Clock Phase ([*SPIIn_CTRL2.clkpha*](#)) selects one of two different transfer formats.

For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the SCK edge for the slave to latch the data. See section [Clock Phase and Polarity Control](#) for additional details.

12.4.2 SPI Peripheral Clock

Refer to [Table 12-1: MAX78000 SPI Instances](#) for the specific input clock, f_{INPUT_CLK} , used for each SPI instance. SPI instances assigned to the AHB bus the SPI input clock is the system clock, SYS_CLK. For SPI instances mapped to the APB bus the SPI

input clock is the system peripheral clock, PCLK. The SPI input clock drives the SPIn peripheral clock. The SPIn provides an internal clock, *SPIn_CLK*, that is used within the SPI peripheral for the base clock to control the module and generate the SCK clock when in master mode. Set the SPIn internal clock using the field *SPIn_CLKCTRL.scale* as shown in [Equation 12-1](#). Valid settings for *SPIn_CLKCTRL.scale* are 0 to 8, allowing a divisor of 1 to 256.

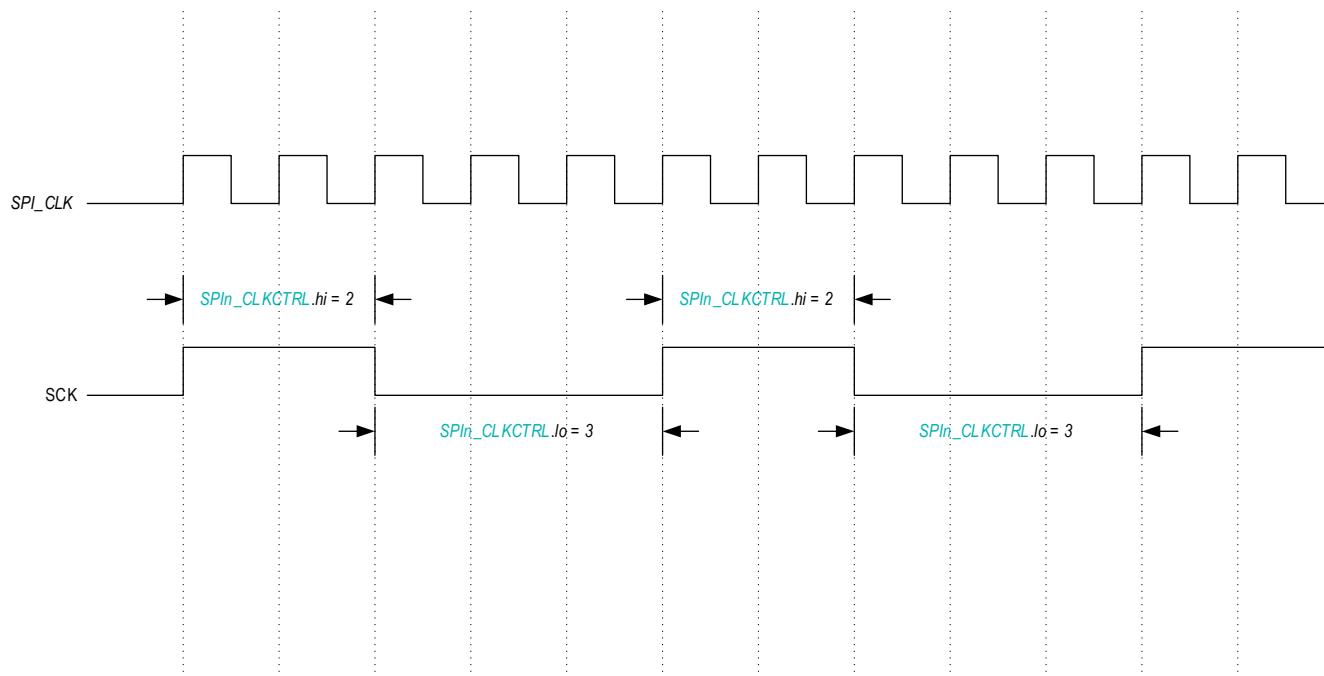
Equation 12-1: SPI Peripheral Clock

$$f_{SPI_CLK} = \frac{f_{INPUT_CLK}}{2^{clkdiv}}$$

12.4.3 Master Mode Serial Clock Generation

In master and multi-master mode the SCK clock is generated by the master. The SPIn provides control for both the high time and low time of the SCK clock. This control allows setting the high and low times for the SCK to duty cycles other than 50% if required. The SCK clock uses the SPIn peripheral clock as a base value and the high and low values are a count of the number of f_{SPI_CLK} clocks. [Figure 12-5](#), visually represents the use of the *SPIn_CLKCTRL.hi* and *SPIn_CLKCTRL.lo* fields for a non-50% duty cycle serial clock generation. See [Equation 12-2](#) and [Equation 12-3](#) for calculating the SCK high and low time from the *SPIn_CLKCTRL.hi* and *SPIn_CLKCTRL.lo* field values.

Figure 12-5: SCK Clock Rate Control



Equation 12-2: SCK High Time

$$t_{SCK_HI} = t_{SPIn_CLK} \times SPIn_CLKCTRL.hi$$

Equation 12-3: SCK Low Time

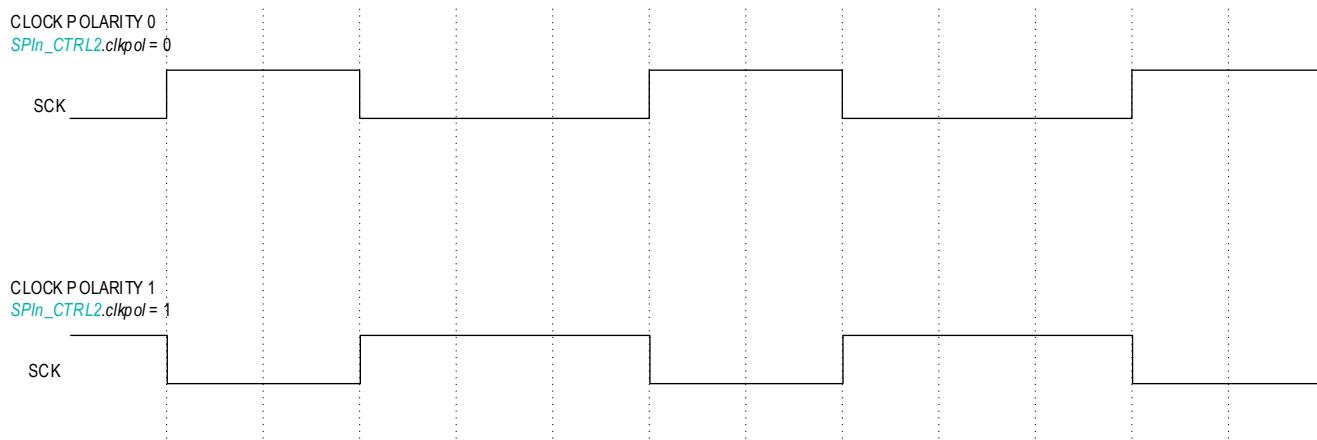
$$t_{SCK_LOW} = t_{SPIn_CLK} \times SPIn_CLKCTRL.lo$$

12.4.4 Clock Phase and Polarity Control

SPIn supports four combinations of clock and phase polarity as shown in [Table 12-5](#). Clock polarity is controlled using the bit *SPIn_CTRL2.clkpol* and determines if the clock is active high or active low as shown in [Figure 12-6](#). Clock polarity does not affect the transfer format for SPI. Clock phase determines when the data must be stable for sampling. Setting the clock

phase to 0, *SPIn_CTRL2.clkpha* = 0, dictates the SPI data is sampled on the initial SPI clock edge regardless of clock polarity. Phase 1, *SPIn_CTRL2.clkpha* = 1, results in data sample occurring on the second edge of the clock regardless of clock polarity.

Figure 12-6: SPI Clock Polarity



For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the SCK edge for the slave to latch the data.

Table 12-5: SPI Modes Clock Phase and Polarity Operation

| SPI Mode | <i>SPIn_CTRL2.clkpha</i> | <i>SPIn_CTRL2.clkpol</i> | SCK Transmit Edge | SCK Receive Edge | SCK Idle State |
|----------|--------------------------|--------------------------|-------------------|------------------|----------------|
| 0 | 0 | 0 | Falling | Rising | Low |
| 1 | 0 | 1 | Rising | Falling | High |
| 2 | 1 | 0 | Rising | Falling | Low |
| 3 | 1 | 1 | Falling | Rising | High |

12.4.5 SPIn FIFOs

The Transmit FIFO hardware is 32 bytes deep. The write data width can be 8-, 16- or 32-bits wide. A 16-bit write queues a 16-bit word to the FIFO hardware. A 32-bit write queues two 16-bit words to the FIFO hardware with the least significant word dequeued first. Bytes must be written to two consecutive byte addresses, with the odd byte as the most significant byte, and the even byte as the least significant byte. The FIFO logic waits for both the odd and even bytes to be written to this register space before dequeuing the 16-bit result to the FIFO.

The Receive FIFO hardware is 32 bytes deep. Read data width can be 8-, 16- or 32-bits. A byte read from this register dequeues one byte from the FIFO. A 16-bit read from this register dequeues two bytes from the FIFO, least significant byte first. A 32-bit read from this register dequeues four bytes from the FIFO, least significant byte first.

12.4.6 SPI Interrupts and Wakeups

The SPIn supports multiple interrupt sources. Status flags for each interrupt are set regardless of the state of the interrupt enable bit for that event. The event happens once when the condition is satisfied. The status flag must be cleared by firmware by writing a 1 to the interrupt flag.

The following FIFO interrupts are supported:

- Transmit FIFO Empty
- Transmit FIFO Threshold
- Receive FIFO Full
- Receive FIFO Threshold

- Transmit FIFO Underrun
 - ◆ Slave mode only, Master mode stalls the serial clock
- Transmit FIFO Overrun
- Receive FIFO Underrun
- Receive FIFO Overrun
 - ◆ Slave Mode only, Master Mode stalls the serial clock
- SPIn supports interrupts for the internal state of the SPI as well as external signals. The following transmission interrupts are supported:
- SS_n asserted or deasserted
- SPI transaction complete
 - ◆ Master mode only
- Slave mode transaction aborted
- Multi-master fault

The SPIn port can wake up the microcontroller from low-power modes when the wake event is enabled. SPIn events that can wake the microcontroller are:

- Receive FIFO full
- Transmit FIFO empty
- Receive FIFO threshold
- Transmit FIFO threshold

12.5 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 12-6: SPIn Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERAL_n_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 12-6: SPIn Register Summary

| Offset | Register Name | Access | Description |
|----------|---------------------|--------|--|
| [0x0000] | <i>SPIn_FIFO32</i> | R/W | SPIn FIFO Data Register |
| [0x0000] | <i>SPIn_FIFO16</i> | R/W | SPIn 16-bit FIFO Data Register |
| [0x0000] | <i>SPIn_FIFO8</i> | R/W | SPIn 8-bit FIFO Data Register |
| [0x0004] | <i>SPIn_CTRL0</i> | R/W | SPIn Master Signals Control Register |
| [0x0008] | <i>SPIn_CTRL1</i> | R/W | SPIn Transmit Packet Size Register |
| [0x000C] | <i>SPIn_CTRL2</i> | R/W | SPIn Static Configuration Register |
| [0x0010] | <i>SPIn_SSTIME</i> | R/W | SPIn Slave Select Timing Register |
| [0x0014] | <i>SPIn_CLKCTRL</i> | R/W | SPIn Master Clock Configuration Register |
| [0x001C] | <i>SPIn_DMA</i> | R/W | SPIn DMA Control Register |
| [0x0020] | <i>SPIn_INTFL</i> | R/W1C | SPIn Interrupt Flag Register |
| [0x0024] | <i>SPIn_INTEN</i> | R/W | SPIn Interrupt Enable Register |
| [0x0028] | <i>SPIn_WKFL</i> | R/W1C | SPIn Wakeup Flags Register |
| [0x002C] | <i>SPIn_WKEN</i> | R/W | SPIn Wakeup Enable Register |
| [0x0030] | <i>SPIn_STAT</i> | RO | SPIn Status Register |

12.5.1 Register Details

Table 12-7: SPIn FIFO32 Register

| SPIn FIFO Data | | | SPIn_FIFO32 | | [0x0000] |
|----------------|------|--------|-------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | SPIn FIFO Data Register This register is used for the SPI Transmit and Receive FIFO. Reading from this register returns characters from the Receive FIFO and writing to this register adds characters to the Transmit FIFO. Read and write this register in either 1-byte, 2-byte or 4-byte widths only. Reading from an empty FIFO or writing to a full FIFO results in undefined behavior. | |

Table 12-8: SPIn 16-bit FIFO Register

| SPIn FIFO Data | | | SPIn_FIFO16 | | [0x0000] |
|----------------|------|--------|-------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:16 | - | R/W | 0 | Reserved Do not modify | |
| 15:0 | data | R/W | 0 | SPIn 16-bit FIFO Data Register This register is used for the SPI Transmit and Receive FIFO. Reading from this register returns characters from the Receive FIFO and writing to this register adds characters to the Transmit FIFO. Read and write this register in 2-byte width only for 16-bit FIFO access. Reading from an empty FIFO or writing to a full FIFO results in undefined behavior. | |

Table 12-9: SPIn 8-bit FIFO Register

| SPIn 8-bit FIFO Data | | | SPIn_FIFO8 | | [0x0000] |
|----------------------|------|--------|------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | SPIn FIFO Data Register This register is used for the SPI Transmit and Receive FIFO. Reading from this register returns characters from the Receive FIFO and writing to this register adds characters to the Transmit FIFO. Read and write this register in either 1-byte, 2-byte or 4-byte widths only. Reading from an empty FIFO or writing to a full FIFO results in undefined behavior. | |
| 7:0 | data | R/W | 0 | SPIn 8-bit FIFO Data Register This register is used for the SPI Transmit and Receive FIFO. Reading from this register returns characters from the Receive FIFO and writing to this register adds characters to the Transmit FIFO. Read and write this register in 1-byte width only for 8-bit FIFO access. Reading from an empty FIFO or writing to a full FIFO results in undefined behavior. | |

Table 12-10: SPIn Control 0 Register

| SPIn Control 0 | | | SPIn_CTRL0 | | [0x0004] |
|----------------|------|--------|------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:20 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |

| SPIn Control 0 | | | | SPIn_CTRL0 | [0x0004] |
|----------------|-----------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 19:16 | ss_active | R/W | 0 | Master Slave Select The SPIn includes up to four slave select lines for each port. This field selects which slave select pin is active when the next SPI transaction is started (<i>SPIn_CTRL0.start</i> = 1). One or more slave select pins can be selected for each SPI transaction by setting the bit for each slave select pin. For example, use SPIn_SS0 and SPIn_SS2 by setting this field to 0b0101 or select all slave selects by setting this field to 0b1111. <i>Note: This field is only used when the SPIn is configured for Master Mode (<i>SPIn_CTRL0.mst_mode</i> = 1).</i> | |
| 15:9 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 8 | ss_ctrl | R/W | 0 | Master Slave Select Control This field controls the behavior of the slave select pins at the completion of a transaction. The default behavior, <i>ss_ctrl</i> = 0, deasserts the slave select pin at the completion of the transaction. Set this field to 1 to leave the slave select pins asserted at the completion of the transaction. If the external device supports this behavior, leaving the slave select pins asserted allows multiple transactions without the delay associated with deassertion of the slave select pin between transactions. 0: Slave Select is deasserted at the end of a transmission 1: Slave Select stays asserted at the end of a transmission | |
| 7:6 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 5 | start | R/W1O | 0 | Master Start Data Transmission Set this field to 1 to start a SPI master mode transaction. 0: No master mode transaction active. 1: Master initiates a data transmission. Ensure that all pending transactions are complete before setting this field to 1. <i>Note: This field is only used when the SPIn is configured for Master Mode (<i>SPIn_CTRL0.master</i> = 1).</i> | |
| 4 | ss_io | R/W | 0 | Master Slave Select Signal Direction Set the I/O direction for 0: Slave Select is an output 1: Slave Select is an input <i>Note: This field is only used when the SPIn is configured for Master Mode (<i>SPIn_CTRL0.mst_mode</i> = 1).</i> | |
| 3:2 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 1 | mst_mode | R/W | 0 | SPI Master Mode Enable This field selects between slave mode and master mode operation for the SPI port. Write this field to 0 to operate as an SPI slave. Setting this field to 1 sets the port as an SPI master. 0: Slave mode SPI operation. 1: Master mode SPI operation. | |

| SPIn Control 0 | | | | SPIn_CTRL0 | [0x0004] |
|----------------|------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 0 | en | R/W | 0 | SPI Enable/Disable This field enables and disables the SPIn port. Disable the SPIn port by setting this field to 0. Disabling the SPIn port does not affect the SPIn FIFOs or register settings. Access to SPIn registers is always available. 0: SPIn port is disabled 1: SPIn port is enabled | |

Table 12-11: SPIn Control 1 Register

| SPIn Transmit Packet Size | | | | SPIn_CTRL1 | [0x0008] |
|---------------------------|-------------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:16 | rx_num_char | R/W | 0 | Number of Receive Characters Number of characters to receive in RX FIFO. <i>Note: If the SPIn port is set to operate in 4-wire mode, this field is ignored and the SPIn_CTRL1.tx_num_chars field is used for both the number of characters to receive and transmit.</i> | |
| 15:0 | tx_num_char | R/W | 0 | Number of Transmit Characters Number of characters to transmit from TX FIFO. <i>Note: If the SPIn port is set to operate in 4-wire mode, this field is used for both the number of characters to receive and transmit.</i> | |

Table 12-12: SPIn Control 2 Register

| SPIn Control 2 | | | | SPIn_CTRL2 | [0x000C] |
|----------------|------------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:20 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 19:16 | ss_pol | R/W | 0 | Slave Select Polarity Controls the polarity of each individual SS signal where each bit position corresponds to a SS signal. SPIn_SS0 is controlled with bit position 0 and SPIn_SS2 is controlled with bit position 2. For each bit position, 0: SS is active low 1: SS is active high | |
| 15 | three_wire | R/W | 0 | Three-Wire SPI Enable Set this field to 1 to enable three-wire SPI communication. Set this field to 0 for four-wire full-duplex SPI communication. 0: Four-wire full-duplex mode enabled. 1: Three-wire mode enabled <i>Note: This field is ignored for Dual SPI, SPIn_CTRL2.data_width =1, and Quad SPI, SPIn_CTRL2.data_width =2.</i> | |
| 14 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |

| SPIn Control 2 | | | | SPIn_CTRL2 | [0x000C] |
|----------------|------------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 13:12 | data_width | R/W | 0b00 | SPI Data Width This field controls the number of data lines used for SPI communications. Three-wire SPI: <i>data_width</i> = 0 Set this field to 0, indicating SPIn_MOSI is used for half-duplex communication. Four-wire full-duplex SPI: <i>data_width</i> = 0 Set this field to 0, indicating SPIn_MOSI and SPIn_MISO are used for the SPI data output and input respectively. Dual Mode SPI: <i>data_width</i> = 1 Set this field to 1, indicating SPIn_SDIO0 and SPIn_SDIO1 are used for half-duplex communication. Quad Mode SPI: <i>data_width</i> = 2 Set this field to 2, indicating SPIn_SDIO0, SPIn_SDIO1, SPIn_SDIO2 and SPIn_SDIO3 are used for half-duplex communication. 0: 1-bit per SCK cycle (Three-wire half-duplex SPI and Four-wire full-duplex SPI) 1: 2-bits per SCK cycle (Dual Mode SPI) 2: 4-bits per SCK cycle (Quad Mode SPI) 3: Reserved <i>Note: When this field is set to 0, use the field SPIn_CTRL2.three_wire to select either Three-Wire SPI or Four-Wire SPI operation.</i> | |
| 11:8 | numbits | R/W | 0 | Number of Bits per Character Set this field to the number of bits per character for the SPI transaction. Setting this field to 0 indicates a character size of 16. 0: 16-bits per character 1: 1-bit per character (not supported) 2: 2-bits per character ... 14: 14-bits per character 15: 15-bits per character <i>Note: 1-bit and 9-bit character lengths are not supported.</i> <i>Note: 2-bit and 10-bit character lengths do not support maximum SCK speeds in Master mode. SPIn_CLK.scale must be > 0</i> <i>Note: For Dual and Quad mode SPI, the character size should be divisible by the number of bits per SCK cycle.</i> | |
| 7:2 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 1 | clkpol | R/W | 0 | Clock Polarity This field controls the SCK polarity. The default clock polarity is for SPI Mode 0 and Mode 1 operation and is active high. Invert the SCK polarity for SPI Mode 2 and Mode 3 operation. 0: Standard SCK for use in SPI Mode 0 and Mode 1 1: Inverted SCK for use in SPI Mode 2 and Mode 3 | |
| 0 | clkpha | R/W | 0 | Clock Phase 0: Data sampled on clock rising edge. Use when in SPI Mode 0 and Mode 2 1: Data sampled on clock falling edge. Use when in SPI Mode 1 and Mode 3 | |

Table 12-13: SPI_n Slave Select Timing Register

| SPI _n Slave Select Timing | | | | SPI _n _SSTIME | [0x0010] |
|--------------------------------------|-------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:24 | - | R/W | 0 | Reserved for Future Use Do not modify this field. <i>Note: The SPI_n_SSTIME register bit settings only apply when SPI_n is operating in Master mode (SPI_n_CTRL0.master = 1)</i> | |
| 23:16 | inact | R/W | 0 | Inactive Stretch This field controls the number of system clocks the bus is inactive between the end of a transaction (Slave Select inactive) and the start of the next transaction (Slave Select active). 0: 256 1: 1 2: 2 3: 3 254: 254 255: 255 | |
| 15:8 | post | R/W | 0 | Slave Select Hold Post Last SCK Number of system clock cycles that SS remains active after the last SCK edge. 0: 256 1: 1 2: 2 3: 3 254: 254 255: 255 | |
| 7:0 | pre | R/W | 0 | Slave Select Delay to First SCK Set the number of system clock cycles the Slave Select is held active prior to the first SCK edge. 0: 256 1: 1 2: 2 3: 3 254: 254 255: 255 | |

 Table 12-14: SPI_n Master Clock Configuration Registers

| SPI _n Master Clock Configuration | | | | SPI _n _CLKCTRL | [0x0014] |
|---|------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:20 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |

| SPIn Master Clock Configuration | | | | SPIn_CLKCTRL | [0x0014] |
|---------------------------------|--------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 19:16 | clkdiv | R/W | 0 | SPI Peripheral Clock Scale Scales the SPI input clock (PCLK) by 2^{scale} to generate the SPIn peripheral clock. $f_{\text{SPInCLK}} = \frac{f_{\text{SPIn_INPUT_CLK}}}{2^{\text{clkdiv}}}$ Valid values for scale are 0 to 8 inclusive. Values greater than 8 are reserved for future use. <i>Note: 1-bit and 9-bit character lengths are not supported.</i> <i>Note: If SPIn_CLKCTRL.clkdiv = 0, SPIn_CLKCTRL.hi = 0, and SPIn_CLKCTRL.lo = 0, character sizes of 2 and 10 bits are not supported.</i> | |
| 15:8 | hi | R/W | 0 | SCK Hi Clock Cycles Control 0: Hi duty cycle control disabled. Only valid if SPIn_CLK.clkdiv = 0. 1 - 15: The number of SPIn peripheral clocks, f_{SPInCLK} , that SCK is high. <i>Note: 1-bit and 9-bit character lengths are not supported.</i> <i>Note: If SPIn_CLKCTRL.clkdiv = 0, SPIn_CLKCTRL.hi = 0, and SPIn_CLKCTRL.lo = 0, character sizes of 2 and 10 bits are not supported.</i> | |
| 7:0 | lo | R/W | 0 | SCK Low Clock Cycles Control This field controls the SCK low clock time and is used to control the overall SCK duty cycle in combination with the SPIn_CLK.hi field. 0: Low duty cycle control disabled. Setting this field to 0 is only valid if SPIn_CLK.clkdiv = 0. 1 to 15: The number of SPIn peripheral clocks, f_{SPInCLK} , that the SCK signal is low. <i>Note: 1-bit and 9-bit character lengths are not supported.</i> <i>Note: If SPIn_CLKCTRL.clkdiv = 0, SPIn_CLKCTRL.hi = 0, and SPIn_CLKCTRL.lo = 0, character sizes of 2 and 10 bits are not supported.</i> | |

Table 12-15: SPIn DMA Control Registers

| SPIn DMA Control | | | SPIn_DMA | | [0x001C] |
|------------------|--------------|--------|----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31 | dma_rx_en | R/W | 0 | RX DMA Enable 0: Disabled. Any pending DMA requests are cleared 1: Enabled | |
| 30 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 29:24 | dma_rx_en | R | 0 | Number of Bytes in the RX FIFO Read returns the number of bytes currently in the RX FIFO | |
| 23 | dma_rx_flush | W | - | Clear the RX FIFO 1: Clear the RX FIFO and any pending RX FIFO flags in SPIn_INTFL. This should be done when the RX FIFO is inactive. Writing a 0 has no effect. | |
| 22 | rx_fifo_en | R/W | 0 | RX FIFO Enabled 0: Disabled 1: Enabled | |
| 21 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |

| SPIn DMA Control | | SPIn_DMA | | | [0x001C] |
|------------------|------------|----------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 20:16 | rx_thd_val | R/W | 0x00 | RX FIFO Threshold Level Set this value to the desired RX FIFO threshold level. When the RX FIFO level crosses above this setting, a DMA request is triggered if enabled by setting SPIn_DMA.dma_tx_en , and SPIn_INTFL.rx_thd_val becomes set. Valid values are 0 to 30. <i>Note: 31 is an invalid setting and reserved for future use.</i> | |
| 15 | dma_tx_en | R/W | 0 | TX DMA Enable 0: Disabled. Any pending DMA requests are cleared 1: TX DMA is enabled | |
| 14 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 13:8 | tx_lvl | RO | 0 | Number of Bytes in the TX FIFO Read this field to determine the number of bytes currently in the TX FIFO. | |
| 7 | tx_flush | R/W | 0 | TX FIFO Clear Set this bit to clear the TX FIFO and all TX FIFO flags in the SPIn_INTFL register. <i>Note: The TX FIFO should be disabled (SPIn_DMA.tx_fifo_en = 0) prior to setting this field.</i> <i>Note: Setting this field to 0 has no effect.</i> | |
| 6 | tx_fifo_en | R/W | 0 | TX FIFO Enabled 0: Disabled 1: Enabled | |
| 5 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 4:0 | tx_thd_val | R/W | 0x10 | TX FIFO Threshold Level Set this value to the desired TX FIFO threshold level. When the TX FIFO count (SPIn_DMA.tx_lvl) falls below this value, a DMA request is triggered if enabled by setting SPIn_DMA.dma_tx_en and SPIn_INTFL.tx_thd_val becomes set. | |

Table 12-16: SPIn Interrupt Status Flags Registers

| SPIn Interrupt Status Flags | | | SPIn_INTFL | | [0x0020] |
|-----------------------------|-------|--------|------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:16 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 15 | rx_un | R/1 | 0 | RX FIFO Underrun Flag Set when a read is attempted from an empty RX FIFO. | |
| 14 | rx_ov | R/W1C | 0 | RX FIFO Overrun Flag Set if SPI is in Slave Mode, and a write to a full RX FIFO is attempted. If the SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is read from the RX FIFO. | |

| SPIn Interrupt Status Flags | | | | SPIn_INTFL | [0x0020] |
|-----------------------------|----------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 13 | tx_un | R/W1C | 0 | TX FIFO Underrun Flag Set if SPI is in Slave Mode, and a read from empty TX FIFO is attempted. If SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is written to the empty TX FIFO. | |
| 12 | tx_ov | R/W1C | 0 | TX FIFO Overrun Flag Set when a write is attempted to a full TX FIFO. | |
| 11 | mst_done | R/W1C | 0 | Master Data Transmission Done Flag Set if SPIn is in Master Mode, and all transactions have completed. <i>SPIn_CTRL1.tx_num_char</i> has been reached. | |
| 10 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 9 | abort | R/W1C | 0 | Slave Mode Transaction Abort Detected Flag Set if the SPI is in Slave Mode, and SS is deasserted before a complete character is received. | |
| 8 | fault | R/W1C | 0 | Multi-Master Fault Flag Set if the SPI is in Master Mode, Multi-Master Mode is enabled, and a Slave Select input is asserted. A collision also sets this flag. | |
| 7:6 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 5 | ssd | R/W1C | 0 | Slave Select Deasserted Flag | |
| 4 | ssa | R/W1C | 0 | Slave Select Asserted Flag | |
| 3 | rx_full | R/W1C | 0 | RX FIFO Full Flag | |
| 2 | rx_thd | R/W1C | 0 | RX FIFO Threshold Level Crossed Flag Set when the RX FIFO exceeds the value in <i>SPIn_DMA.rx_fifo_level</i> . Cleared once RX FIFO level drops below <i>SPIn_DMA.rx_fifo_level</i> . | |
| 1 | tx_em | R/W1C | 1 | TX FIFO Empty Flag The transmit FIFO is empty. | |
| 0 | tx_thd | R/W1C | 0 | TX FIFO Threshold Level Crossed Flag Set when the TX FIFO is less than the value in <i>SPIn_DMA.tx_fifo_level</i> . Cleared once TX FIFO level exceeds <i>SPIn_DMA.tx_fifo_level</i> . | |

Table 12-17: SPIn Interrupt Enable Registers

| SPIn Interrupt Enable | | | | SPIn_INTEN | [0x0024] |
|-----------------------|-------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:16 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 15 | rx_un | R/W | 0 | RX FIFO Underrun Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |

| SPIn Interrupt Enable | | | SPIn_INTEN | | [0x0024] |
|-----------------------|----------|--------|------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 14 | rx_ov | R/W | 0 | RX FIFO Overrun Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 13 | tx_un | R/W | 0 | TX FIFO Underrun Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 12 | tx_ov | R/W | 0 | TX FIFO Overrun Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 11 | mst_done | R/W | 0 | Master Data Transmission Done Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 10 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 9 | abort | R/W | 0 | Slave Mode Abort Detected Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 8 | fault | R/W | 0 | Multi-Master Fault Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 7:6 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 5 | ssd | R/W | 0 | Slave Select Deasserted Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 4 | ssa | R/W | 0 | Slave Select Asserted Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 3 | rx_full | R/W | 0 | RX FIFO Full Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 2 | rx_thd | R/W | 0 | RX FIFO Threshold Level Crossed Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 1 | tx_em | R/W | 0 | TX FIFO Empty Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |
| 0 | tx_thd | R/W | 0 | TX FIFO Threshold Level Crossed Interrupt Enable 0: Interrupt is disabled 1: Interrupt is enabled | |

Table 12-18: SPIn Wakeup Status Flags Registers

| SPIn Wakeup Flags | | | SPIn_WKFL | | [0x0028] |
|-------------------|---------|--------|-----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:4 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 3 | rx_full | R/W1C | 0 | Wake on RX FIFO Full Flag 0: Wake condition has not occurred. 1: Wake condition occurred. | |
| 2 | rx_thd | R/W1C | 0 | Wake on RX FIFO Threshold Level Crossed Flag 0: Wake condition has not occurred. 1: Wake condition occurred. | |
| 1 | tx_em | R/W1C | 0 | Wake on TX FIFO Empty Flag 0: Wake condition has not occurred. 1: Wake condition occurred. | |
| 0 | tx_thd | R/W1C | 0 | Wake on TX FIFO Threshold Level Crossed Flag 0: Wake condition has not occurred. 1: Wake condition occurred. | |

Table 12-19: SPIn Wakeup Enable Registers

| SPIn Wakeup Enable | | | SPIn_WKEN | | [0x002C] |
|--------------------|---------|--------|-----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:4 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 3 | rx_full | R/W | 0 | Wake on RX FIFO Full Enable 0: Wake event is disabled 1: Wake event is enabled. | |
| 2 | rx_thd | R/W | 0 | Wake on RX FIFO Threshold Level Crossed Enable 0: Wake event is disabled 1: Wake event is enabled. | |
| 1 | tx_em | R/W | 0 | Wake on TX FIFO Empty Enable 0: Wake event is disabled 1: Wake event is enabled. | |
| 0 | tx_thd | R/W | 0 | Wake on TX FIFO Threshold Level Crossed Enable 0: Wake event is disabled 1: Wake event is enabled. | |

Table 12-20: SPIn Slave Select Timing Registers

| SPIn Status | | | SPIn_STAT | | [0x0030] |
|-------------|------|--------|-----------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:1 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |

| SPIn Status | | | | SPIn_STAT | [0x0030] |
|-------------|------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 0 | busy | R | 0 | <p>SPI Active Status SPI status flag, see value descriptions for details of each value.</p> <p>0: SPIn is not active. In Master mode the <i>busy</i> flag is cleared when the last character is sent. In Slave mode the <i>busy</i> field is cleared when the configured slave select input is deasserted.</p> <p>1: SPIn is active. In Master mode the <i>busy</i> flag is set when a transaction starts. In Slave mode the <i>busy</i> flag is set when a configured slave select input is asserted.</p> <p><i>Note:</i> <i>SPIn_CTRL0</i>, <i>SPIn_CTRL1</i>, <i>SPIn_CTRL2</i>, <i>SPIn_SS TIME</i>, and <i>SPIn_CLKCTRL</i> should not be configured if this bit is set.</p> | |

13. I²C Master/Slave Serial Communications Peripheral (I²C)

The I²C peripherals can be configured as either an I²C master or I²C slave at standard data rates. For simplicity, I²Cn is used throughout this section to refer to any of the I²C peripherals.

For detailed information on I²C bus operation, refer to Maxim Application Note 4024 “SPI/I²C Bus Lines Control Multiple Peripherals” at <https://www.maximintegrated.com/en/app-notes/index.mvp/id/4024>

13.1 I²C Master/Slave Features

Each I²C Master/Slave is compliant with the I²C Bus Specification and include the following features:

- Communicates via a serial data bus (SDA) and a serial clock line (SCL)
- Operates as either a master or slave device as a transmitter or receiver
- Supports I²C Standard Mode, Fast Mode, Fast Mode Plus and High Speed (Hs) mode
- Transfers data at rates up to:
 - ◆ 100kbps in Standard Mode
 - ◆ 400kbps in Fast Mode
 - ◆ 1Mbps in Fast Mode Plus
 - ◆ 3.4Mbps in Hs Mode
- Supports multi-master systems, including support for arbitration and clock synchronization for Standard, Fast and Fast Plus modes
- Supports 7- and 10-bit addressing
- Supports RESTART condition
- Supports clock stretching
- Provides transfer status interrupts and flags
- Provides DMA data transfer support
- Supports I²C timing parameters fully controllable via firmware
- Provides glitch filter and Schmitt trigger hysteresis on SDA and SCL
- Provides control, status, and interrupt events for maximum flexibility
- Provides independent 8-byte RX FIFO and 8-byte TX FIFO
- Provides TX FIFO preloading
- Provides programmable interrupt threshold levels for the TX and RX FIFO

13.2 Instances

The three instances of the peripheral are shown in *Table 13-1: MAX78000 I²C Peripheral Pins* lists the locations of the SDA and SCL signals for each of the I²Cn peripherals per package.

Table 13-1: MAX78000 I²C Peripheral Pins

| I ² C Instance | Alternate Function | Alternate Function # | Package 81-CTBGA |
|---------------------------|-----------------------|----------------------|---------------------|
| I ² C0 | I ² C0_SCL | AF1 | P0.10 |
| | I ² C0_SDA | AF1 | P0.11 |
| I ² C1 | I ² C1_SCL | AF1 | P0.16 |
| | I ² C1_SDA | AF1 | P0.17 |
| I ² C2 | I ² C2_SCL | AF1 | P0.30 |
| | I ² C2_SDA | AF1 | P0.31 |

13.3 I²C Overview

13.3.1 I²C Bus Terminology

Table 13-2, below, contains terms and definitions used in this chapter for the I²C Bus Terminology.

Table 13-2: I²C Bus Terminology

| Term | Definition |
|------------------|--|
| Transmitter | The device that sends data to the bus. |
| Receiver | The device that receives data from the bus. |
| Master | The device that initiates a transfer, generates clock signals and terminates a transfer. |
| Slave | The device addressed by a master. |
| Multi-master | More than one master can attempt to control the bus at the same time without corrupting the message. |
| Arbitration | Procedure to ensure that, if more than one master simultaneously tries to control the bus, only one can do so and the resulting message is not corrupted. |
| Synchronization | Procedure to synchronize the clock signals of two or more devices. |
| Clock Stretching | When a slave device holds SCL low to pause a transfer until it is ready. This feature is optional according to the I ² C Specification; thus, a master does not have to support slave clock stretching if none of the slaves in the system are capable of clock stretching. |

13.3.2 I²C Transfer Protocol Operation

The I²C protocol operates over a two-wire bus: a clock circuit (SCL) and a data circuit (SDA). I²C is a half-duplex protocol: only one device is allowed to transmit on the bus at a time.

Each transfer is initiated when the bus master sends a START or repeated START condition. It is followed by the I²C slave address of the targeted slave device plus a read/write bit. The master can transmit data to the slave (a ‘write’ operation) or receive data from the slave (a ‘read’ operation). Information is sent most significant bit (MSB) first. Following the slave address, the master indicates a read or write operation and then exchanges data with the addressed slave. An acknowledge bit is sent by the receiving device after each byte is transferred. When all necessary data bytes have been transferred, a STOP or RESTART condition is sent by the bus master to indicate the end of the transaction. After the STOP condition has been sent, the bus is idle and ready for the next transaction. After a RESTART condition is sent, the same master begins a new transmission. The number of bytes that can be transmitted per transfer is unrestricted.

13.3.3 START and STOP Conditions

A START condition occurs when a bus master pulls SDA from high to low while SCL is high, and a STOP condition occurs when a bus master allows SDA to be pulled from low to high while SCL is high. Because these are unique conditions that cannot occur during normal data transfer, they are used to denote the beginning and end of the data transfer.

13.3.4 Master Operation

I²C transmit and receive data transfer operations occur through the *I2Cn_FIFO* register. Writes to the register load the TX FIFO and reads of the register return data from the RX FIFO. If a slave sends a NACK in response to a write operation, the I²C master generates an interrupt. The I²C controller can be configured to issue a STOP condition to free the bus.

The receive FIFO contains the received data. If the receive FIFO is full or the transmit FIFO is empty, the I²C master stops the clock to allow time to read bytes from the receive FIFO or load bytes into the transmit FIFO.

13.3.5 Acknowledge and Not Acknowledge

An acknowledge bit (ACK) is generated by the receiver, whether I²C master or slave, after every byte received by pulling SDA low. The ACK bit is how the receiver tells the transmitter that the byte was successfully received, and another byte might be sent.

A Not Acknowledge (NACK) occurs if the receiver does not generate an ACK when the transmitter releases SDA. A NACK is generated by allowing SDA to float high during the acknowledge time slot. The I²C master can then either generate a STOP condition to abort the transfer, or it can generate a repeated START condition (that is, send a START condition without an intervening STOP condition) to start a new transfer.

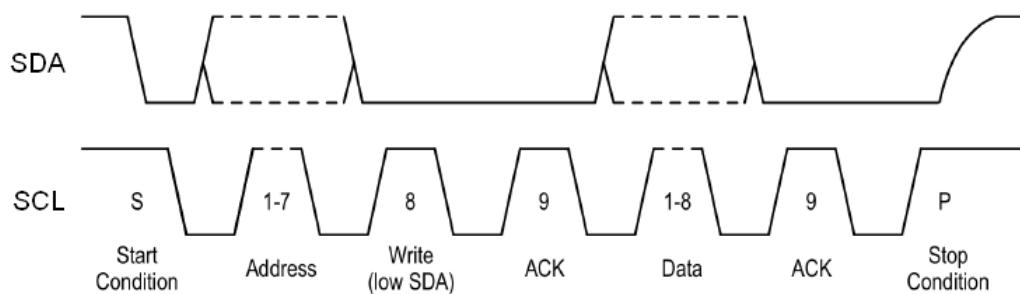
A receiver can generate a NACK after a byte transfer if any of the following conditions occur:

- No receiver is present on the bus with the transmitted address. In that case, no device will respond with an acknowledge signal.
- The receiver is unable to receive or transmit because it is busy and is not ready to start communication with the master.
- During the transfer, the receiver receives data or commands it does not understand.
- During the transfer, the receiver is unable to receive any more data.
- If an I²C master has requested data from a slave, it signals the slave to stop transmitting by sending a NACK following the last byte it requires.

13.3.6 Bit Transfer Process

Both SDA and SCL circuits are open-drain, bidirectional circuits. Each requires an external pullup resistor that ensures each circuit is high when idle. The I²C specification states that during data transfer, the SDA line can change state only when SCL is low, and that SDA is stable and able to be read when SCL is high as shown in *Figure 13-1, below*.

Figure 13-1: I²C Write Data Transfer



An example of an I²C data transfer is as follows:

1. A bus master indicates a data transfer to a slave with a START condition.
2. The master then transmits one byte with a 7-bit slave address and a single read-write bit: a zero for a write or a one for a read.
3. During the next SCL clock following the read-write bit, the master releases SDA. During this clock period, the addressed slave responds with an ACK by pulling SDA low.
4. The master senses the ACK condition and begins transferring data. If reading from the slave, it floats SDA and allows the slave to drive SDA to send data. After each byte, the master drives SDA low to acknowledge the byte. If writing to the slave, the master drives data on the SDA circuit for each of the eight bits of the byte, and then floats SDA during the ninth bit to allow the slave to reply with the ACK indication.
5. After the last byte is transferred, the master indicates the transfer is complete by generating a STOP condition. A STOP condition is generated when the master pulls SDA from a low to high while SCL is high.

13.4 I²C Configuration and Usage

13.4.1 SCL and SDA Bus Drivers

SCL and SDA are open-drain signals. In this device, once the I²C peripheral is enabled and the proper GPIO alternate function is selected, the corresponding pad circuits are automatically configured as open-drain outputs. However, SCL can also be optionally configured as a push-pull driver to conserve power and avoid the need for any pullup resistor. This should only be used in systems where no I²C slave device can hold SCL low, such as for clock stretching. Push-pull operation is enabled by setting `I2Cn_CTRL.sclppm` to 1. SDA, on the other hand, always operates in open-drain mode.

13.4.2 SCL Clock Configurations

The SCL frequency is dependent upon the values of I²C peripheral clock and the values of the external pullup resistor and trace capacitance on the SCL clock line.

Note: An external RC load on the SCL line will affect the target SCL frequency calculation.

13.4.3 SCL Clock Generation for Standard, Fast and Fast-Plus Modes

The master generates the I²C clock on the SCL line. When operating as a master, application code must configure the `I2Cn_CLK_HI` and `I2Cn_CLK_LO` registers for the desired I²C operating frequency.

The SCL high time is configured in the I²C Clock High Time register field `I2Cn_CLK_HI.scl_hi` using Equation 13-2. The SCL low time is configured in the I²C Clock Low Time register field `I2Cn_CLK_LO.scl_lo` using Equation 13-3. Each of these fields is 8-bits. The I²C frequency value is shown in [Equation 13-1, below](#).

Equation 13-1: I²C Clock Frequency

$$f_{I2C_CLK} = \frac{1}{t_{I2C_CLK}} \text{ is either } f_{PCLK} \text{ or } f_{IBRO}$$

Equation 13-2: I²C Clock High Time Calculation

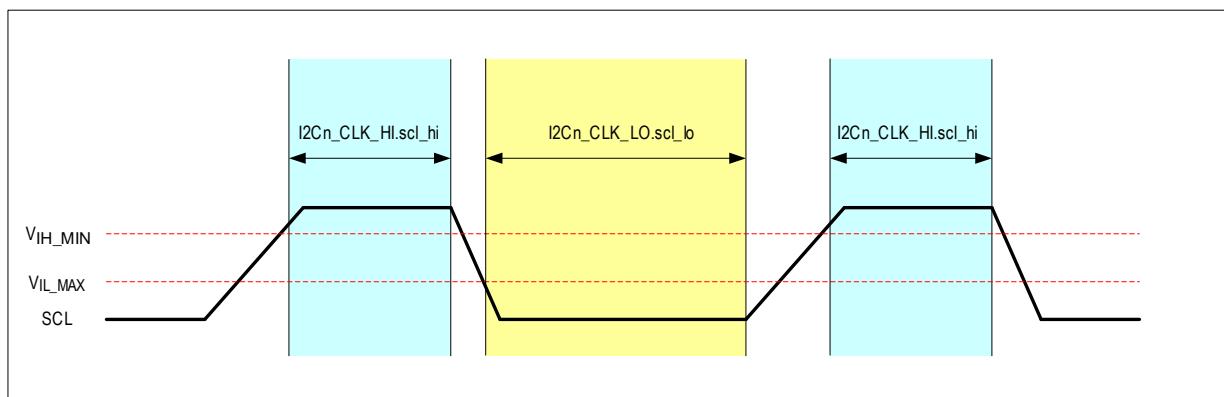
$$t_{SCL_HI} = t_{I2C_CLK} \times (\text{I2Cn_CLK_HI}.scl_hi + 1)$$

Equation 13-3: I²C Clock Low Time Calculation

$$t_{SCL_LO} = t_{I2C_CLK} \times (\text{I2Cn_CLK_LO}.scl_lo + 1)$$

[Figure 13-2](#) shows the association between the SCL clock low and high times for Standard, Fast and Fast Plus I²C frequencies.

Figure 13-2: I²C SCL Timing for Standard, Fast and Fast-Plus Modes



During synchronization, external masters or external slaves may be driving SCL simultaneously. This affects the SCL duty cycle. By monitoring SCL, the controller can determine whether an external master or slave is holding SCL low. In either

case, the controller waits until SCL is high before starting to count the number of SCL high cycles. Similarly, if an external master pulls SCL low before the controller has finished counting SCL high cycles, then the controller starts counting SCL low cycles and releases SCL once the time period, *I2Cn_CLK.lo.scl_lo*, has expired.

Because the controller does not start counting the high/low time until the input buffer detects the new value, the actual clock behavior is based on many factors. These include bus loading, other devices on the bus holding SCL low, and the filter delay time of this device.

13.4.4 SCL Clock Generation for Hs-mode

To operate the I²C interface in Hs-mode at its maximum speed (~3.4MHz), values to be programmed into the *I2Cn_HS_CLK.hs_clk_lo* register and *I2Cn_HS_CLK.hs_clk_hi* register must be determined. Since the Hs-mode operation is entered by first using one of the lower speed modes for pre-amble, a relevant lower speed mode must also be configured. See *SCL Clock Generation for Standard, Fast and Fast-Plus Modes* for information regarding configuration of lower speed modes.

13.4.4.1 Hs-Mode Timing

With I²C bus capacitances less than 100pf, the following specifications are extracted from the I²C-bus Specification User Manual Rev. 6 April 2014 <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>

t_{LOW_MIN} = 160ns, the minimum low time for the I²C bus clock.

t_{HIGH_MIN} = 60ns, the minimum high time for the I²C bus clock.

t_{rCL_MAX} = 40ns, the maximum rise time of the I²C bus clock.

t_{fCL_MAX} = 40ns, the maximum fall time of the I²C bus clock.

13.4.4.2 Hs-Mode Clock Configuration

The maximum Hs-mode bus clock frequency can now be determined. The system clock frequency, *f_{sys_CLK}*, must be known. Hs-mode timing information from *Hs-Mode Timing* must be used.

Equation 13-4: I²C Target SCL Frequency

$$\text{Desired Target Maximum I}^2\text{C Frequency: } f_{SCL} = \frac{1}{t_{SCL}}$$

In Hs-mode, the analog glitch filter (AF_MIN) within the device adds a minimum delay of *t_{AF_MIN}* = 10ns.

Equation 13-5: Determining the *I2Cn_HS_CLK.hs_clk_lo* Register Value

$$I2Cn_HS_CLK.\text{hs_clk_lo} = \text{MAX} \left\{ \left\lfloor \left(\frac{t_{LOW_MIN} + t_{FCL_MAX} + t_{I2C_CLK} - t_{AF_MIN}}{t_{I2C_CLK}} \right) \right\rfloor - 1, \quad \frac{t_{SCL}}{t_{I2C_CLK}} - 1 \right\}$$

Equation 13-6: Determining the *I2Cn_HS_CLK.hs_clk_hi* Register Value

$$I2Cn_HS_CLK.\text{hs_clk_hi} = \left\lfloor \left(\frac{t_{HIGH_MIN} + t_{rCL_MAX} + t_{I2C_CLK} - t_{AF_MIN}}{t_{I2C_CLK}} \right) \right\rfloor - 1$$

Equation 13-7: The Calculated Frequency of the I²C Bus Clock Using the Results of *Equation 13-5* and *Equation 13-6*

$$\text{Calculated Frequency} = ((I2Cn_HS_CLK.\text{hs_clk_hi} + 1) + (I2Cn_HS_CLK.\text{hs_clk_lo} + 1)) * t_{I2C_CLK}$$

Table 13-3: Calculated I²C Bus Clock Frequencies shows the I²C bus clock calculated frequencies given different *f_{sys_CLK}* frequencies.

Table 13-3: Calculated I²C Bus Clock Frequencies

| <i>f_{sys_clk}</i> (MHz) | <i>I2Cn_HS_CLK.hs_clk_hi</i> | <i>I2Cn_HS_CLK.hs_clk_lo</i> | Calculated Frequency (MHz) |
|----------------------------------|------------------------------|------------------------------|----------------------------|
| 100 | 4 | 9 | 3.3 |
| 50 | 2 | 4 | 3.125 |
| 25 | 1 | 2 | 2.5 |

13.4.5 I²C Addressing

After a START condition, an I²C slave address byte is transmitted. The I²C slave address is composed of a slave address followed by a read/write bit.

Table 13-4: I²C Slave Address Format

| Slave Address Bits | | R/W Bit | Description |
|--------------------|-----|---------|-----------------------------------|
| 0000 | 000 | 0 | General Call Address |
| 0000 | 000 | 1 | START Condition |
| 0000 | 001 | x | CBUS Address |
| 0000 | 010 | x | Reserved for different bus format |
| 0000 | 011 | x | Reserved for future purposes |
| 0000 | 1xx | x | HS-mode master code |
| 1111 | 1xx | x | Reserved for future purposes |
| 1111 | 0xx | x | 10-bit slave addressing |

In 7-bit addressing mode, the master sends one address byte. To address a 7-bit address slave, first clear *I2Cn_MASTER_CTRL.sl_ex_addr* = 0, then write the address to the TX FIFO formatted as follows where An is address A6:A0.

Master Writing to Slave: 7-bit address : [A6 A5 A4 A3 A2 A1 A0 0]

Master Reading from Slave: 7-bit address : [A6 A5 A4 A3 A2 A1 A0 1]

In 10-bit addressing mode (*I2Cn_MASTER_CTRL.sl_ex_addr* = 1), the first byte the master sends is the 10-bit Slave Addressing byte which includes the first two bits of the 10-bit address, followed by a 0 for the R/W bit. That is followed by a second byte representing the remainder of the 10-bit address. If the operation is a write, this is followed by data bytes to be written to the slave. If the operation is a read, it is followed by a repeated START. Firmware then writes the 10-bit address again with a 1 for the R/W bit. This I²C then starts receiving data from the slave device.

13.4.6 I²C Master Mode Operation

The peripheral operates in master mode when Master Mode Enable *I2Cn_CTRL.mst* = 1. To initiate a transfer, the master generates a START condition by setting *I2Cn_MASTER_CTRL.start* = 1. If the bus is busy, it does not generate a START condition until the bus is available.

A master can communicate with multiple slave devices without relinquishing the bus. Instead of generating a STOP condition after communicating with the first slave, the master generates a Repeated START condition, or RESTART, by setting *I2Cn_MASTER_CTRL.restart* = 1. If a transaction is in progress, the peripheral finishes the transaction before generating a RESTART. The peripheral then transmits the slave address stored in the TX FIFO. The *I2Cn_MASTER_CTRL.restart* bit is automatically cleared to 0 as soon as the master begins a RESTART condition.

I2Cn_MASTER_CTRL.start is automatically cleared to 0 after the master has completed a transaction and sent a STOP condition.

The master can also generate a STOP condition by setting *I2Cn_MASTER_CTRL.stop* = 1.

If both START and RESTART conditions are enabled at the same time, a START condition is generated first. Then, at the end of the first transaction, a RESTART condition is generated.

If both RESTART and STOP conditions are enabled at the same time, a STOP condition is not generated. Instead, a RESTART condition is generated. After the RESTART condition is generated, both bits are cleared.

If START, RESTART, and STOP are all enabled at the same time, a START condition is first generated. At the end of the first transaction, a RESTART condition is generated. The *I2Cn_MASTER_CTRL.stop* bit is cleared and ignored.

A slave cannot generate START, RESTART, or STOP conditions. Therefore, when Master Mode is disabled, the *I2Cn_MASTER_CTRL.start*, *I2Cn_MASTER_CTRL.restart*, and *I2Cn_MASTER_CTRL.stop* bits are all cleared to 0.

For master mode operation, the following registers should only be configured when either:

1. The I²C peripheral is disabled,
or
2. The I²C bus is guaranteed to be idle/free.

If this peripheral is the only master on the bus, then changing the registers outside of a transaction (*I2Cn_MASTER_CTRL.start* = 0) will satisfy this requirement:

- *I2Cn_CTRL.mst*
- *I2Cn_CTRL.rx_mode*
- *I2Cn_CTRL.scl_pp_mode*
- *I2Cn_CTRL.hs_mode*
- *I2Cn_RX_CTRL1.rx_cnt*
- *I2Cn_MASTER_CTRL.sl_ex_addr*
- *I2Cn_MASTER_CTRL.mcode*
- *I2Cn_CLK_LO.scl_lo*
- *I2Cn_CLK_HI.scl_hi*
- *I2Cn_HS_CLK.hs_clk_lo*
- *I2Cn_HS_CLK.hs_clk_hi*

In contrast to the above set of registers, these registers below can be safely (re)programmed at any time:

- All interrupt flags and interrupt enable bits
- *I2Cn_TX_CTRL0.tx_thresh*
- *I2Cn_RX_CTRL0.rx_thresh*
- *I2Cn_TIMEOUT.to*
- *I2Cn_DMA.rx_en*
- *I2Cn_DMA.tx_en*
- *I2Cn_FIFO.data*
- *I2Cn_MASTER_CTRL.start*
- *I2Cn_MASTER_CTRL.restart*
- *I2Cn_MASTER_CTRL.stop*

13.4.6.1 I²C Master Mode Receiver Operation

When in Master Mode, initiating a Master Receiver operation begins with the following sequence:

1. Write the number of data bytes to receive to the I²C Receive Count field (*I2Cn_RX_CTRL1.rx_cnt*).
2. Write the I²C Slave Address Byte to the *I2Cn_FIFO* register with the R/W bit set to 1
3. Send a START condition by setting *I2Cn_MASTER_CTRL.start* = 1
4. The slave address is transmitted by the controller from the *I2Cn_FIFO* register.
5. The I²C controller receives an ACK from the slave and the controller sets the address ACK interrupt flag (*I2Cn_INT_FLO.addr_ack* = 1).
6. The I²C controller receives data from the slave and automatically ACKs each byte. Firmware must retrieve this data by reading the *I2Cn_FIFO* register.
7. Once *I2Cn_RX_CTRL1.rx_cnt* data bytes have been received, the I²C controller sends a NACK to the slave and sets the Transfer Done Interrupt Status Flag (*I2Cn_INT_FLO.done* = 1).
8. If *I2Cn_MASTER_CTRL.restart* or *I2Cn_M.stop* is set, then the I²C controller sends a repeated START or STOP, respectively.

13.4.6.2 I²C Master Mode Transmitter Operation

When in Master Mode, initiating a Master Transmitter operation begins with the following sequence:

1. Write the I²C Slave Address Byte to the *I2Cn_FIFO* register with the R/W bit set to 0
2. Write the desired data bytes to the *I2Cn_FIFO* register, up to the size of the TX FIFO. (e.g. If the TX FIFO size is 8 bytes, firmware may write one address byte and seven data bytes prior to starting the transaction.)
3. Send a START condition by setting *I2Cn_MASTER_CTRL.start* = 1
4. The controller transmits the slave address byte written to the *I2Cn_FIFO* register.
5. The I²C controller receives an ACK from the slave and the controller sets the address ACK interrupt flag (*I2Cn_INT_FLO.addr_ack* = 1).
6. The *I2Cn_FIFO* register data bytes are transmitted on the SDA line.
 - a. The I²C controller receives an ACK from the slave after each data byte
 - b. As the transfer proceeds, firmware should refill the TX FIFO by writing to the *I2Cn_FIFO* register as needed.
 - c. If the TX FIFO goes empty during this process, the controller will pause at the beginning of the byte and wait for firmware to either write more data or instruct the controller to send a RESTART or STOP condition
7. Once firmware has written all the desired bytes to the *I2Cn_FIFO* register, firmware should set either *I2Cn_MASTER_CTRL.restart* or *I2Cn_MASTER_CTRL.stop*.
8. Once the controller sends all the remaining bytes and empties the TX FIFO, it will set *I2Cn_INT_FLO.done* and proceed to send out either a RESTART condition, if *I2Cn_MASTER_CTRL.restart* was set, or a STOP condition, if *I2Cn_MASTER_CTRL.stop* was set.

13.4.6.3 I²C Multi-Master Operation

The I²C protocol supports multiple masters on the same bus. When the bus is free, it is possible that two (or more) masters might try to initiate communication at the same time. This is a valid bus condition. If this occurs and the two masters want to transmit different data and/or address different slaves, only one master can remain in master mode and complete its transaction. The other master must back off transmission and wait until the bus is idle. This process by which the winning master is determined is called bus arbitration.

To determine which master wins the arbitration, for each address or data bit, the master compares the data being transmitted on SDA to the value observed on SDA. If a master attempts to transmit a 1 on SDA (that is, the master lets SDA

float) but senses a 0 instead, then that master loses arbitration, and the other master that sent a zero continues with the transaction. The losing master cedes the bus by switching off its SDA and SCL drivers.

Note that this arbitration scheme works with any number of bus masters: if more than two masters begin transmitting simultaneously, the arbitration continues as each master cedes the bus until only one master remains transmitting. Data is not corrupted because as soon as each master realizes it has lost arbitration, it stops transmitting on SDA, leaving the following data bits sent on SDA intact.

If the I²C master peripheral detects it has lost arbitration, it stops generating SCL; sets *I2Cn_INT_FLO.areri*; sets *I2Cn_INT_FLO.tx_lock_out*, flushing any remaining data in the TX FIFO; and clears *I2Cn_MASTER_CTRL.start*, *I2Cn_MASTER_CTRL.restart*, and *I2Cn_MASTER_CTRL.stop* to 0. So long as the peripheral is not itself addressed by the winning master, the I²C peripheral stays in master mode (*I2Cn_CTRL.mst* = 1). If at any time another master addresses this peripheral using the address programmed in *I2Cn_SLAVE_ADDR.slave_addr*, then the I²C peripheral clears *I2Cn_CTRL.mst* to 0 and begins responding as a slave. This can even occur during the same address transmission during which the peripheral lost arbitration.

*Note: Arbitration loss is considered an error condition, and like the other error conditions will set *I2Cn_INT_FLO.tx_lock_out*. Therefore, after an arbitration loss, firmware will need to clear *I2Cn_INT_FLO.tx_lock_out* and reload the TX FIFO.*

Also, in a multi-master environment, application firmware does not need to wait for the bus to become free before attempting to start a transaction (writing 1 to *I2Cn_MASTER_CTRL.start*). If the bus is free when *I2Cn_MASTER_CTRL.start* is set to 1, the transaction begins immediately. If instead the bus is busy, then the peripheral will:

1. Wait for the other master to complete the transaction(s) by sending a STOP,
2. Count out the bus free time using $t_{BUF} = t_{SCL_LO}$ (see *Equation 13-3*), and then
3. Send a START condition and begin transmitting the slave address byte(s) in the TX FIFO, followed by the rest of the transfer.

The I²C master peripheral is compliant with all bus arbitration and clock synchronization requirements of the I²C specification; this operation is automatic, and no additional programming is required.

13.4.7 I²C Slave Mode Operation

When in slave mode, the I²Cn peripheral operates as a slave device on the I²C bus and responds to an external master's requests to transmit or receive data. To configure the I²Cn peripheral as a slave, write the *I2Cn_CTRL.mst* bit to zero. The I²Cn clock is driven by the master on the bus, so the SCL device pin will be driven by the external master and *I2Cn_STATUS.clk_mode* remains a zero. The desired slave address must be set by writing to the *I2Cn_SLAVE_ADDR.slave_addr* register.

For slave mode operation, the following register fields should be configured with the I²Cn peripheral disabled:

- *I2Cn_CTRL.mst* = 0 for Slave operation.
- *I2Cn_CTRL.gen_call_addr*
- *I2Cn_CTRL.rx_mode*
 - ◆ The recommended value for this field is 0. Note that a setting of 1 is incompatible with slave mode operation with clock stretching disabled (*I2Cn_CTRL.scl_clk_stretch_dis* = 1).
- *I2Cn_CTRL.scl_clk_stretch_dis*
- *I2Cn_CTRL.hs_mode*
- *I2Cn_RX_CTRL0.dnr*
 - ◆ SMBus/PMBus applications should set this to 0, while other applications should set this to 1.
- *I2Cn_TX_CTRL0.tx_nack_afd*
- *I2Cn_TX_CTRL0.tx_amr_afd*
- *I2Cn_TX_CTRL0.tx_amw_afd*
- *I2Cn_TX_CTRL0.tx_amgc_afd*
- *I2Cn_TX_CTRL0.tx_preload*

- ◆ Recommended value is 0 for applications that can tolerate slave clock stretching (*I2Cn_CTRL.scl_clk_stretch_dis* = 0).
- ◆ Recommend value is 1 for applications that do not allow slave clock stretching (*I2Cn_CTRL.scl_clk_stretch_dis* = 1).
- *I2Cn_CLK_HI.scl_hi*
 - ◆ Applies to Slave Mode when clock stretching is enabled (*I2Cn_CTRL.scl_clk_stretch_dis* = 0)
 - This will be used to satisfy $t_{SU;DAT}$ after clock stretching; program it so that the value defined by [Equation 13-2](#) is $\geq t_{SU;DAT(\min)}$
- *I2Cn_HS_CLK.hs_clk_hi*
 - ◆ Applies to Slave Mode in Hs Mode when clock stretching is enabled (*I2Cn_CTRL.scl_clk_stretch_dis* = 0)
 - This will be used to satisfy $t_{SU;DAT}$ after clock stretching during Hs-Mode operation; program it so that the value defined by [Equation 13-6](#) is $\geq t_{SU;DAT(\min)}$
- *I2Cn_SLAVE_ADDR.slave_addr*
- *I2Cn_SLAVE_ADDR.ex_addr*

In contrast to the above register fields, the following register fields can be safely (re)programmed at any time:

- All Interrupt Flags and Interrupt Enables
- *I2Cn_TX_CTRL0.tx_thresh* and *I2Cn_RX_CTRL0.rx_thresh*
 - ◆ TX and RX FIFO Threshold Levels
- *I2Cn_TX_CTRL1.tx_rdy*
 - ◆ Transmit Ready (Can only be cleared by hardware)
- *I2Cn_TIMEOUT.to*
 - ◆ Time Out Control
- *I2Cn_DMA.rx_en* and *I2Cn_DMA.tx_en*
 - ◆ TX and RX DMA Enables
- *I2Cn_FIFO.data*
 - ◆ FIFO access register

13.4.7.1 Slave Transmitter

The device will operate as a slave transmitter when the received address matches the device slave address with the R/W bit set to 1. The master is then reading from the device slave. There two main modes of slave transmitter operation: just-in-time mode and preload mode.

In just-in-time mode, firmware waits to write the transmit data to the TX FIFO until after the master addresses it for a READ transaction, “just in time” for the data to be sent to the master. This allows firmware to defer the determination of what data should be sent until the time of the address match. As an example, the transmit data could be based off an immediately preceding I²C WRITE transaction that requests a certain block of data to be sent, or the data could represent the latest, most up-to-date value of a sensor reading. Clock stretching *must* be enabled (*I2Cn_CTRL.scl_clk_stretch_dis* = 0) for just-in-time mode operation.

Program flow for transmit operation in just-in-time mode is as follows:

1. With `I2Cn_CTRL.i2c_en` = 0, initialize all relevant registers, including specifically for this mode `I2Cn_CTRL`. `scl_clk_stretch_dis` = 0, `I2Cn_TX_CTRL0[5:2]` = 0x8 and `I2Cn_TX_CTRL0.tx_reload` = 0. Don't forget to program `I2Cn_CLK_HI.scl_hi` and `I2Cn_HS_CLK.hs_clk_hi` with appropriate values satisfying $t_{SU;DAT}$ (and HS $t_{SU;DAT}$).
2. Software sets `I2Cn_CTRL.i2c_en` = 1.
 - a. The controller is now listening for its address. For either a transmit (R/W = 1) or receive (R/W = 0) operation, the peripheral will respond to its address with an ACK.
 - b. When the address match occurs, hardware will set `I2Cn_INT_FLO.addr_match` and `I2Cn_INT_FLO.tx_lock_out`.
3. Software waits for `I2Cn_INT_FLO.addr_match` = 1, either via polling the interrupt flag or setting `I2Cn_INT_EN0.addr_match` to interrupt the CPU.
4. After reading `I2Cn_INT_FLO.addr_match` = 1, Software reads `I2Cn_CTRL.read` to determine whether the transaction is a transmit (read=1) or receive (read=0) operation. In this case we assume read=1, indicating transmit.
 - a. At this point, Hardware will hold SCL low until Software clears `I2Cn_INT_FLO.tx_lock_out` and loads data into the FIFO.
5. Software clears `I2Cn_INT_FLO.addr_match` and `I2Cn_INT_FLO.tx_lock_out`. Now that `I2Cn_INT_FLO.tx_lock_out` is 0, Software can begin loading the transmit data into `I2Cn_FIFO`.
6. As soon as there is data in the FIFO, Hardware will release SCL (after counting out `I2Cn_CLK_HI.scl_hi`) and send out the data on the bus.
7. While the master keeps requesting data and sending ACKs, `I2Cn_INT_FLO.ddone` will remain 0 and Software should continue to monitor the TX FIFO and refill it as needed.
 - a. The FIFO level can be monitored synchronously via the TX FIFO status/interrupt flags, or asynchronously by setting `I2Cn_TX_CTRL0.tx_thresh` and setting `I2Cn_INT_EN0.tx_thresh` interrupt.
 - b. If the TX FIFO ever empties during the transaction, the Hardware will start clock stretching and wait for it to be refilled.
8. The master ends the transaction by sending a NACK. Once this happens the `I2Cn_INT_FLO.done` interrupt flag is set, and Software can stop monitoring the TX FIFO.
9. The transaction is complete, Software should "clean up", including clearing `I2Cn_INT_FLO.done` and clearing `I2Cn_INT_EN0.tx_thresh` interrupt. We return to step 3, waiting on an address match.
10. If Software needs to know how many data bytes were transmitted to the master, it should check the TX FIFO level as soon as Software sees `I2Cn_INT_FLO.done` = 1 and use that to determine how many data bytes were successfully sent.
 - a. *Note that any data remaining in the TX FIFO will be discarded prior to the next transmit operation; it is NOT necessary for Software to manually flush the TX FIFO for this to occur.*

The other mode of operation for slave transmit is preload mode. In this mode, it is assumed that the application firmware knows prior to the transmit operation what data it should send to the master. This data is then "preloaded" into the TX

FIFO. Once the address match occurs, this data can be sent out without any software intervention. Preload mode can be used with clock stretching either enabled or disabled, but it is the only option if clock stretching must be disabled.

To use slave transmit preload mode:

1. With `I2Cn_CTRL.i2c_en` = 0, initialize all relevant registers, including specifically for this mode `I2Cn_CTRL.cl_clk_stretch_dis` = 1, `I2Cn_TX_CTRL0[5:2]` = 0xF and `I2Cn_TX_CTRL0.tx_reload` = 1.
2. Software sets `I2Cn_CTRL.i2c_en` = 1.
 - a. Even though the controller is enabled, at this point it will not ACK an address match with R/W=1 until Software sets `I2Cn_TX_CTRL1.tx_ready` = 1.
3. Software prepares for the transmit operation by loading data into the transmit FIFO, enabling DMA, setting `I2Cn_TX_CTRL0.tx_thresh` and setting `I2Cn_INT_EN0.tx_thresh` interrupt, etc.
 - a. If clock stretching is disabled, then an empty TX FIFO during the transmit operation will cause a TX underrun error. Therefore, firmware should take any necessary steps to avoid an underrun *prior* to setting `I2Cn_TX_CTRL1.tx_ready` = 1.
 - b. If clock stretching is enabled, then an empty TX FIFO will not cause a TX underrun error. However, it is recommended to follow the same preparation steps to minimize the amount of time spent clock stretching, which will let the transaction complete as quickly as possible.
4. Once Software has prepared for the transmit operation, it sets `I2Cn_TX_CTRL1.tx_ready` = 1.
 - a. The controller is now fully enabled and will respond with an ACK to an address match.
 - b. Hardware will set `I2Cn_INT_F0.addr_match` once an address match has occurred. `I2Cn_INT_F0.tx_lock_out` will NOT be set and will remain 0.
5. Software waits for `I2Cn_INT_F0.addr_match` = 1, either via polling the interrupt flag or setting `I2Cn_INT_EN0.amie` to interrupt the CPU.
6. After seeing `I2Cn_INT_F0.addr_match` = 1, Software reads `I2Cn_CTRL.read` to determine whether the transaction is a transmit (read=1) or receive (read=0) operation. In this case we assume `I2Cn_CTRL.read`, indicating transmit.
 - a. At this point, Hardware will begin sending out the data that was preloaded into the TX FIFO.
 - b. Once the first data byte is sent, Hardware will also automatically clear `I2Cn_TX_CTRL1.tx_ready` to 0.
7. While the master keeps requesting data and sending ACKs, `I2Cn_INT_F0.done` will remain 0 and Software should continue to monitor the TX FIFO and refill it as needed.
 - a. The FIFO level can be monitored synchronously via the TX FIFO status/interrupt flags, or asynchronously by setting `I2Cn_TX_CTRL0.tx_thresh` and setting `I2Cn_INT_EN0.tx_thresh` interrupt.
 - b. If clock stretching is disabled and the TX FIFO ever empties during the transaction, the Hardware will set `I2Cn_INT_FL1.tx_underflow` = 1 and send 0xFF for all following data bytes requested by the master.
8. The master ends the transaction by sending a NACK. Once this happens the `I2Cn_INT_F0.done` interrupt flag is set.
 - a. If the TX FIFO goes empty at the same time that the master indicates the transaction is complete by sending a NACK, this is not considered an underrun event, and the `I2Cn_INT_FL1.tx_underflow` flag will remain 0.
9. The transaction is complete, Software should "clean up", which should include clearing `I2Cn_INT_F0.done`. We return to step 3 and prepare for the next transaction.
 - a. If Software needs to know how many data bytes were transmitted to the master, it should check the TX FIFO level as soon as Software sees `I2Cn_INT_F0.done` = 1 and use that to determine how many data bytes were successfully sent.
 - b. By default, any data remaining in the TX FIFO will NOT be discarded, and instead will be reused for the next transmit operation.
 - c. If this is not desired, Software can flush the TX FIFO. The safest way to do this is by clearing and then re-setting `I2Cn_CTRL.i2c_en`. This will flush both the TX and RX FIFOs.

Once a slave starts transmitting out of its *I2Cn_FIFO*, detection of out of sequence STOP, START or RESTART condition will terminate the current transaction. When a transaction is terminated in such manner, *I2Cn_INT_FLO.start_er* or *I2Cn_INT_FLO.stop_er* will be set to 1.

If the TX FIFO is not ready (*I2Cn_TX_CTRL1.tx_ready* = 0) and the I²C controller receives a data read request from the master, the hardware automatically sends a NACK at the end of the first address byte. The setting of the Do Not Respond field is ignored by the hardware in this case because the only opportunity to send a NACK for an I²C read transaction is after the address byte.

13.4.7.2 Slave Receivers

The device will operate as a slave receiver when the received address matches the device slave address with the R/W bit set to 0. The external master is writing to the slave.

Program flow for a receive operation is as follows:

1. With *I2Cn_CTRL.i2c_en* = 0, initialize all relevant registers.
2. Set *I2Cn_CTRL.i2c_en* = 1.
 - a. If an address match with R/W=0 occurs, and the RX FIFO is empty, the peripheral will respond with an ACK and the *I2Cn_INT_FLO.addr_match* flag will be set.
 - b. If the RX FIFO is not empty, then depending on the value of *I2Cn_RX_CTRL0.dnr*, the peripheral will NACK either the address byte (*I2Cn_RX_CTRL0.dnr* = 1) or the first data byte (*I2Cn_RX_CTRL0.dnr* = 0).
3. Wait for *I2Cn_INT_FLO.addr_match* = 1, either by polling or by enabling the *wr_addr_match* interrupt to the CPU. Once a successful address match occurs, hardware will set *I2Cn_INT_FLO.addr_match* =1.
4. Read *I2Cn_CTRL.read* to determine whether the transaction is a transmit (*I2Cn_CTRL.read* = 1) or receive (*I2Cn_CTRL.read* = 0) operation. In this case we assume *I2Cn_CTRL.read* = 0, indicating receive. At this point, the device will begin receiving data into the RX FIFO.
5. Clear *I2Cn_INT_FLO.addr_match*, and while the master keeps sending data, *I2Cn_INT_FLO.done* will remain 0 and software should continue to monitor the RX FIFO and empty it as needed.
 - a. The FIFO level can be monitored synchronously via the RX FIFO status/interrupt flags, or asynchronously by setting *I2Cn_RX_CTRL0.rx_thresh* and enabling the *I2Cn_INT_FLO.rx_thresh* interrupt.
 - b. If the RX FIFO ever fills up during the transaction, then hardware will set *I2Cn_INT_FL1.rx_underflow* and then either:
 - i. If *I2Cn_CTRL.scl_clk_stretch_dis* = 0, start clock stretching and wait for software to read from the RX FIFO, **or**
 - ii. If *I2Cn_CTRL.scl_clk_stretch_dis* = 1, respond to the master with a NACK and the last byte is discarded.
6. The master ends the transaction by sending a RESTART or STOP. Once this happens the *I2Cn_INT_FLO.done* interrupt flag is set, and software can stop monitoring the RX FIFO.
7. Once a slave starts receiving into its RX_FIFO, detection of out of sequence STOP, START or RESTART condition will release the I²C bus to the Idle state and hardware will set *I2Cn_INT_FLO.start_er* or *I2Cn_INT_FLO.stop_er* to 1.

If software has not emptied the data in the RX FIFO from the previous transaction by the time that a master addresses it for another write (i.e. receive) transaction, then the controller will *not* participate in the transaction, and no additional data will be written into the FIFO. Although a NACK *will* be sent to the master, software can control whether the NACK is sent with the initial address match, or if instead it is sent at the end of the first data byte. Setting *I2Cn_RX_CTRL0.dnr* to 1 chooses the former, while setting *I2Cn_RX_CTRL0.dnr* to 0 chooses the latter.

13.4.8 I²C Interrupt Sources

The I²C controller has a very flexible interrupt generator that generates an interrupt signal to the Interrupt Controller on any of several events. On recognizing the I²C interrupt, firmware determines the cause of the interrupt by reading the I²C Interrupt Flags registers *I2Cn_INT_FL0* and *I2Cn_INT_FL1*. Interrupts can be generated for the following events:

- Transaction Complete (Master/Slave)
- Address NACK received from slave (Master)
- Data NACK received from slave (Master)
- Lost arbitration (Master)
- Transaction timeout (Master/Slave)
- FIFO is empty, not empty, full to configurable threshold level (Master/Slave)
- TX FIFO locked out because it is being flushed (Master/Slave)
- Out of sequence START and STOP conditions (Master/Slave)
- Sent a NACK to an external master because the TX or RX FIFO was not ready (Slave)
- Address ACK or NACK received (Master)
- Incoming address match (Slave)
- TX Underflow or RX Overflow (Slave)

Interrupts for each event can be enabled or disabled by setting or clearing the corresponding bit in the *I2Cn_INT_EN0* or *I2Cn_INT_EN1* interrupt enable register.

Note: Disabling the interrupt does not prevent the corresponding flag from being set by hardware but does prevent an IRQ when the interrupt flag is set.

Note: Prior to enabling an interrupt, the status of the corresponding interrupt flag should be checked and, if necessary, serviced or cleared. This prevents a previous interrupt event from interfering with a new I²C communications session.

13.4.9 TX FIFO and RX FIFO

There are separate transmit and receive FIFOs, TX FIFO and RX FIFO. Both are accessed using the FIFO Data register *I2Cn_FIFO*. Writes to this register enqueue data into the TX FIFO. Writes to a full TX FIFO have no effect. Reads from *I2Cn_FIFO* dequeue data from the RX FIFO. Writes to a full TS FIFO have no effect and reads from an empty RX FIFO return 0xFF.

The TX and RX FIFO will only read or write one byte at a time. Transactions larger than 8 bits can still be performed, however. A 16- or 32-bit write to the TX FIFO stores just the lowest 8 bits of the write data. A 16- or 32-bit read from the RX FIFO will have the valid data in the lowest 8 bits and 0's in the upper bits. In any case, the TX and RX FIFOs will only accept 8 bits at a time for either reads or writes.

To offload work from the CPU, the DMA can read and write to each FIFO. See section *I²C DMA Control* for more information on configuring the DMA.

During a receive transaction (which during master operation is a READ, and during slave operation is a WRITE), received bytes are automatically written to the RX FIFO. Software should monitor the RX FIFO level and unload data from it as needed by reading *I2Cn_FIFO*. If the receive FIFO becomes full during a master mode transaction, then the controller sets the *I2Cn_INT_FL1.rx_underflow* the *I2Cn_INT_FL1.rx_underflow* bit and one of two things will happen depending on the value of *I2Cn_CTRL.scl_clk_stretch_dis*:

- If clock stretching is enabled (*I2Cn_CTRL.scl_clk_stretch_dis* = 0), then the controller stretches the clock until software makes space available in the RX FIFO by reading from *I2Cn_FIFO*. Once space is available, the peripheral moves the data byte from the shift register into the RX FIFO, the SCL device pin is released, and the master is free to continue the transaction.
- If clock stretching is disabled (*I2Cn_CTRL.scl_clk_stretch_dis* = 1), then the controller responds to the master with a NACK and the data byte is lost. The master can return the bus to idle with a STOP condition or start a new transaction with a RESTART condition.

During a transmit transaction (which during master operation is a WRITE, and during slave operation is a READ), either user software or the DMA can provide data to be transmitted by writing to the TX FIFO. Once the peripheral finishes transmitting each byte, it removes it from the TX FIFO and, if available, begins transmitting the next byte.

Interrupts can be generated for the following FIFO status:

- TX FIFO level less than or equal to threshold
- RX FIFO level greater than or equal to threshold
- TX FIFO underflow
- RX FIFO overflow
- TX FIFO locked for writing

Both the RX FIFO and TX FIFO are flushed when the I²Cn port is disabled by clearing *I2Cn_CTRL.i2c_en=0*. While the peripheral is disabled, writes to the TX FIFO have no effect and reads from the RX FIFO return 0xFF.

The TX FIFO and RX FIFO can be flushed by setting the Transmit FIFO Flush bit (*I2Cn_TX_CTRL0.tx_flush=1*) or the Receive FIFO Flush bit (*I2Cn_RX_CTRL0.rx_flush=1*), respectively. In addition, under certain conditions the TX FIFO is automatically locked by hardware and flushed so stale data is not unintentionally transmitted. The TX FIFO is automatically flushed, and writes locked out from software under the following conditions:

- General Call Address Match. Automatic flushing and lockout can be disabled by setting *I2Cn_TX_CTRL0.tx_amgc_afd*.
- Slave Address Match Write. Automatic flushing and lockout can be disabled by setting *I2Cn_TX_CTRL0.tx_amw_afd*.
- Slave Address Match Read. Automatic flushing and lockout can be disabled by setting *I2Cn_TX_CTRL0.tx_amr_afd*.
- During operation as a slave transmitter, a NACK is received. Automatic flushing and lockout can be disabled by setting *I2Cn_TX_CTRL0.tx_nack_afd*.
- Any of the following interrupts: Arbitration Error, Timeout Error, Master Mode Address NACK Error, Master Mode Data NACK Error, Start Error, and STOP Error. Automatic flushing cannot be disabled for these conditions.

When the above conditions occur, the TX FIFO is flushed so that data intended for a previous transaction is not transmitted unintentionally for a new transaction. In addition to flushing the TX FIFO, the Transmit Lockout Flag is set (*I2Cn_INT_F0.tx_lock_out = 1*) and writes to the TX FIFO are ignored until firmware acknowledges the external event by clearing *I2Cn_INT_F0.tx_lock_out*.

13.4.10 TX FIFO Preloading

There may be situations during slave mode operation where software wants to preload the TX FIFO prior to a transmission, such as when clock stretching is disabled. In this scenario, rather than responding to an external master requesting data with an ACK and clock stretching while software writes the data to the TX FIFO, the controller will instead respond with a NACK until software has preloaded the requested data into the TX FIFO.

When TX FIFO Preloading is enabled, the software controls ACKs to the external master using the TX Ready (*I2Cn_TX_CTRL1.tx_ready*) bit. When *I2Cn_TX_CTRL1.tx_ready* is set to 0, hardware automatically NACKs all read transactions from the Master. Setting *I2Cn_TX_CTRL1.tx_ready* to 1 sends an ACK to the Master on the next read transaction and transmits the data in the TX FIFO. Preloading the TX FIFO should be complete prior to setting the *I2Cn_TX_CTRL1.tx_ready* field to 1.

The required steps for implementing TX FIFO Preloading in an application are as follow:

1. Enable TX FIFO Preloading by setting `I2Cn_TX_CTRL0.tx_reload` to 1.
This will automatically clear `I2Cn_TX_CTRL1.tx_ready` to 0
2. If the TX FIFO Lockout Flag (`I2Cn_INT_FLO.tx_lock_out`) is set to 1, write 1 to clear the flag and enable writes to the TX FIFO.
3. Enable DMA or Interrupts if required.
4. Load the TX FIFO with the data to send when the Master sends the next read request.
5. Set `I2Cn_TX_CTRL1.tx_ready` to 1 to automatically let the hardware send the preloaded FIFO on the next read from a Master.
6. `I2Cn_TX_CTRL1.tx_ready` is cleared by hardware once it finishes transmitting the first byte, and data is transmitted from the TX FIFO. Once cleared, the application firmware may repeat the Preloading process or disable TX FIFO Preloading.

Note: To prevent the preloaded data from being cleared when the master tries to read it, firmware must at least set `I2Cn_TX_CTRL0.tx_amr_afd` to 1, disabling auto flush on READ address match. The software will determine whether the other auto flush disable bits should be set. For example, if a master uses I2C WRITE transactions to determine what data the slave should send in the following READ transactions, then software can clear `I2Cn_TX_CTRL0.tx_amw_afd` to 0. Then when a WRITE occurs, the TX FIFO is flushed, giving firmware time to load the new data. For the READ transaction, the external master can poll the slave address until the new data has been loaded and `I2Cn_TX_CTRL1.tx_ready` is set, at which point the peripheral responds with an ACK.

13.4.11 Interactive Receive Mode (IRXM)

In some situations, the I2Cn might want to inspect and respond to each byte of received data. In this case, Interactive Receive Mode (IRXM) can be used. Interactive Receive Mode is enabled by setting `I2Cn_CTRL.rx_mode` = 1. If Interactive Receive Mode is enabled, it must occur before any I²C transfer is initiated.

When Interactive Receive Mode (IRXM) is enabled, after every data byte received the I2Cn peripheral automatically holds SCL low before the ACK bit. Additionally, after the 8th SCL falling edge, the I2Cn peripheral sets the IRXM Interrupt Status Flag (`I2Cn_INT_FLO.rx_mode` = 1). Application firmware must read the data and generate a response (ACK or NACK) by setting the IRXM Acknowledge (`I2Cn_CTRL.rx_mode_ack`) bit accordingly. Send an ACK by clearing the `I2Cn_CTRL.rx_mode_ack` bit to 0. Send a NACK by setting the `I2Cn_CTRL.rx_mode_ack` bit to 1.

After setting the `I2Cn_CTRL.rx_mode_ack` bit, clear the IRXM interrupt flag. Write 1 to `I2Cn_INT_FLO.rx_mode` to clear the interrupt flag. When the IRXM interrupt flag is cleared, the I2Cn peripheral hardware releases the SCL line and sends the `I2Cn_CTRL.rx_mode_ack` on the SDA line.

While the I2Cn peripheral is waiting for the application firmware to clear the `I2Cn_INT_FLO.rx_mode` flag, firmware can disable Interactive Receive Mode and, if operating as a master, load the remaining number of bytes to be received for the transaction. This allows firmware to examine the initial bytes of a transaction, which might be a command, and then disable Interactive Receive Mode to receive the remaining bytes in normal operation.

During IRXM, received data is not placed in the RX FIFO. Instead, the `I2Cn_FIFO` address is repurposed to directly read the receive shift register, bypassing the RX FIFO. Therefore, before disabling Interactive Receive Mode, firmware must first read the data byte from `I2Cn_FIFO.data`. If the IRXM byte is not read, the byte is lost and the next read from the RX FIFO returns 0xFF.

Note: Interactive Receive Mode does not apply to address bytes, only to data bytes.

Note: Interactive Receive Mode does not apply to general call address responses or START byte responses.

Note: When enabling Interactive Receive Mode and operating as a slave, clock stretching must remain enabled (`I2Cn_CTRL.scl_clk_stretch_dis` = 0).

13.4.12 Clock Stretching

When the I²Cn peripheral requires some response or intervention from the application firmware in order to continue with a transaction, it will hold SCL low, preventing the transfer from continuing. This is called ‘clock stretching’ or ‘stretching the clock’. While the I²C Bus Specification defines the term ‘clock stretching’ to only apply to a slave device holding the SCL line low, this section describes situations where the I²Cn peripheral holds the SCL line low in either slave *or* master mode and refers to *both* as clock stretching.

When the I²Cn peripheral stretches the clock, it typically does so in response to either a full RX FIFO during a receive operation, or an empty TX FIFO during a transmit operation. Necessarily, this occurs before the next data byte begins, either between the ACK bit and the first data bit or, if at the beginning of a transaction, immediately after a START or RESTART condition. However, when operating in Interactive Receive Mode (*I2Cn_CTRL.rx_mode* = 1), the peripheral can also clock stretch *before* the ACK bit, allowing firmware to decide whether to send an ACK or NACK.

For a transmit operation (as either master or slave), when the TX FIFO is empty, SCL is automatically held low after the ACK bit and before the next data byte begins. To stop clock stretching and continue the transaction, firmware must write data to *I2Cn_FIFO.data*. If operating in master mode, however, instead of sending more data, firmware may also set either *I2Cn_MASTER_CTRL.stop* or *I2Cn_MASTER_CTRL.restart* to send a STOP or RESTART condition, respectively.

For a receive operation (as either master or slave), when both the RX FIFO and the receive shift register are full, SCL is automatically held low until at least one data byte is read from the RX FIFO. To stop clock stretching and continue the transaction, firmware must read data from *I2Cn_FIFO.data*. If operating in master mode and this is the final byte of the transaction, as determined by *I2Cn_RX_CTRL1.rx_cnt*, then application firmware must also set either *I2Cn_MASTER_CTRL.stop* or *I2Cn_MASTER_CTRL.restart* to send a STOP or RESTART condition, respectively. This must be done in addition to reading from the RX FIFO, since the peripheral cannot start sending the STOP or RESTART until the last data byte has been moved from the RX shift register into the RX FIFO. (This will occur automatically once there is space in the RX FIFO.)

*Note: Since some masters do not support other devices stretching the clock, it is possible to completely disable all clock stretching during slave mode by setting *I2Cn_CTRL.scl_clk_stretch_dis* to 1 and clearing *I2Cn_CTRL.rx_mode* to 0. In this case, instead of clock stretching the I²Cn peripheral sends a NACK if receiving data or sends 0xFF if transmitting data.*

Note: The clock synchronization required to support other I²C master or slave devices stretching the clock is built into the peripheral and requires no intervention from software to operate correctly.

13.4.13 I²C Bus Timeout

The Timeout register, *I2Cn_TIMEOUT.to*, is used to detect bus errors. [Equation 13-8](#) and [Equation 13-9](#) show equations for calculating the maximum and minimum timeout values based on the value loaded into the *I2Cn_TIMEOUT.to* field.

Equation 13-8: I²C Timeout Maximum

$$t_{TIMEOUT} \leq \left(\frac{1}{f_{I2C_CLK}} \right) \times ((\text{I2Cn_TIMEOUT.to} \times 32) + 3)$$

Due to clock synchronization, the timeout is guaranteed to meet the following minimum time calculation shown in [Equation 13-9](#).

Equation 13-9: I²C Timeout Minimum

$$t_{TIMEOUT} \leq \left(\frac{1}{f_{I2C_CLK}} \right) \times ((\text{I2Cn_TIMEOUT.to} \times 32) + 2)$$

The timeout feature is disabled when *I2Cn_TIMEOUT.to* = 0 and is enabled for any non-zero value. When the timeout is enabled, the timeout timer starts counting when the I²Cn peripheral hardware drives SCL low and is reset by the I²Cn peripheral hardware when the SCL line is released.

The timeout counter only monitors if the I²Cn peripheral hardware is driving the SCL line low. It does not monitor if an external I²Cn device is actively holding the SCL line low. The timeout counter also does not monitor the status of the SDA line.

If the timeout timer expires, a bus error condition has occurred. When a timeout error occurs, the I²Cn peripheral hardware releases the SCL and SDA lines and sets the timeout error interrupt flag to 1 (*I2Cn_INT_FLO.to_er* = 1).

For applications where the device may hold the SCL line low longer than the maximum timeout supported, the timeout can be disabled by setting the timeout field to 0 (*I2Cn_TIMEOUT.to* = 0).

13.4.14 I²C DMA Control

There are independent DMA channels for each TX FIFO and each RX FIFO. DMA activity is triggered by the TX FIFO (*I2Cn_TX_CTRL0.tx_thresh*) and RX FIFO (*I2Cn_RX_CTRL0.rx_thresh*) threshold levels.

When the TX FIFO byte count (*I2Cn_TX_CTRL1.tx_fifo*) is less than or equal to the TX FIFO Threshold Level *I2Cn_TX_CTRL0.tx_thresh*, then the DMA transfers data into the TX FIFO according to the DMA configuration.

To ensure the DMA does not overflow the TX FIFO, the DMA burst size should be set as follows:

Equation 13-10: DMA Burst Size Calculation for I²C Transmit

$$\text{DMA Burst Size} \leq \text{TX FIFO Depth} - \text{I2Cn_TX_CTRL0.tx_thresh} = 8 - \text{I2Cn_TX_CTRL0.tx_thresh}$$

where $0 \leq \text{I2Cn_TX_CTRL0.tx_thresh} \leq 7$

Applications trying to avoid transmit underflow and/or clock stretching should use a smaller burst size and higher *I2Cn_TX_CTRL0.tx_thresh* setting. This fills up the FIFO more frequently but increases internal bus traffic.

When the RX FIFO count (*I2Cn_RX_CTRL1.rx_fifo*) is greater than or equal to the RX FIFO Threshold Level *I2Cn_RX_CTRL0.rx_thresh*, the DMA transfers data out of the RX FIFO according to the DMA configuration. To ensure the DMA does not underflow the RX FIFO, the DMA burst size should be set as follows:

Equation 13-11: DMA Burst Size Calculation for I²C Receive

$$\text{DMA Burst Size} \leq \text{I2Cn_RX_CTRL0.rx_thresh}$$

where $1 \leq \text{I2Cn_RX_CTRL0.rx_thresh} \leq 8$

Applications trying to avoid receive overflow and/or clock stretching should use a smaller burst size and lower *I2Cn_RX_CTRL0.rx_thresh*. This results in reading from the Receive FIFO more frequently but increases internal bus traffic.

Note for receive operations, the length of the DMA transaction (in bytes) must be an integer multiple of I2Cn_RX_CTRL0.rx_thresh. Otherwise, the receive transaction will end with some data still in the RX FIFO, but not enough to trigger an interrupt to the DMA, leaving the DMA transaction incomplete. One easy way to ensure this for all transaction lengths is to set burst size to 1 (I2Cn_RX_CTRL0.rx_thresh = 1).

To enable DMA transfers, enable the TX DMA channel (*I2Cn_DMA.tx_en*) and/or the RX DMA channel (*I2Cn_DMA.rx_en*).

13.5 Registers

See *Table 2-4: APB Peripheral Base Address Map* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in *Table 13-5: I²C Register*

Summary. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 13-5: I²C Register Summary

| Offset | Register | Description |
|----------|-------------------------|---|
| [0x0000] | <i>I2Cn_CTRL</i> | I ² C Control Register |
| [0x0004] | <i>I2Cn_STATUS</i> | I ² C Status Register |
| [0x0008] | <i>I2Cn_INT_FLO</i> | I ² C Interrupt Flags 0 Register |
| [0x000C] | <i>I2Cn_INT_EN0</i> | I ² C Interrupt Enable 0 Register |
| [0x0010] | <i>I2Cn_INT_FL1</i> | I ² C Interrupt Flags 1 Register |
| [0x0014] | <i>I2Cn_INT_EN1</i> | I ² C Interrupt Enable 1 Register |
| [0x0018] | <i>I2Cn_FIFO_LEN</i> | I ² C FIFO Length Register |
| [0x001C] | <i>I2Cn_RX_CTRL0</i> | I ² C Receive Control 0 Register |
| [0x0020] | <i>I2Cn_RX_CTRL1</i> | I ² C Receive Control 1 Register |
| [0x0024] | <i>I2Cn_TX_CTRL0</i> | I ² C Transmit Control 0 Register |
| [0x0028] | <i>I2Cn_TX_CTRL1</i> | I ² C Transmit Control 1 Register |
| [0x002C] | <i>I2Cn_FIFO</i> | I ² C Transmit and Receive FIFO Register |
| [0x0030] | <i>I2Cn_MASTER_CTRL</i> | I ² C Master Control Register |
| [0x0034] | <i>I2Cn_CLK_LO</i> | I ² C Clock Low Time Register |
| [0x0038] | <i>I2Cn_CLK_HI</i> | I ² C Clock High Time Register |
| (0x003C) | <i>I2Cn_HS_CLK</i> | I ² C Hs-Mode Clock Control Register |
| [0x0040] | <i>I2Cn_TIMEOUT</i> | I ² C Timeout Register |
| [0x0048] | <i>I2Cn_DMA</i> | I ² C DMA Enable Register |
| [0x004C] | <i>I2Cn_SLAVE_ADDR</i> | I ² C Slave Address Register |

13.5.1 Register Details

Table 13-6: I²C Control Register

| I ² C Control | | | I2Cn_CTRL | | [0x0000] |
|--------------------------|-------------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 15 | hs_mode | R/W | 0 | Hs-Mode Enable I ² C high speed mode operation 0: Disable 1: Enable | |
| 14 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 13 | scl_pp_mode | R/W | 0 | Single Master Only When set to 1, the device MUST ONLY be used in a single master application with slave devices that are NOT going to hold SCL low (i.e. the slave devices will never clock stretch) | |

| I ² C Control | | | I2Cn_CTRL | | [0x0000] |
|--------------------------|---------------------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 12 | scl_clk_stretch_dis | R/W | 0 | Slave Mode Clock Stretching 0: Enabled 1: Disabled | |
| 11 | read | R | 0 | Slave Read/Write Bit Status Returns the logic level of the R/W bit on a received address match (<i>I2Cn_INT_FLO.addr_match = 1</i>) or general call match (<i>I2Cn_INT_FLO.gen_call_addr = 1</i>). This bit is valid three system clock cycles after the address match status flag is set. | |
| 10 | sw_out_en | R/W | 0 | Software Output Control Enabled Setting this field to 1 enables software bit-bang control of the I2Cn Bus. 0: The I2C controller manages the SDA and SCL pins in hardware. 1: SDA and SCL are controlled by firmware using the <i>I2Cn_CTRL.sda_out</i> and <i>I2Cn_CTRL.scl_out</i> fields. | |
| 9 | sda | R | - | SDA Status 0: SDA pin is logic low. 1: SDA pin is logic high. | |
| 8 | scl | R | - | SCL Status 0: SCL pin is logic low. 1: SCL pin is logic high. | |
| 7 | sda_out | R/W | 0 | SDA Pin Output Control Set the state of the SDA hardware pin (actively pull low or float). 0: Pull SDA Low 1: Release SDA <i>Note: Only valid when I2Cn_CTRL.sw_out_en=1</i> | |
| 6 | scl_out | R/W | 0 | SCL Pin Output Control Set the state of the SCL hardware pin (actively pull low or float). 0: Pull SCL low 1: Release SCL <i>Note: Only valid when I2Cn_CTRL.sw_out_en =1</i> | |
| 5 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 4 | rx_mode_ack | R/W | 0 | Interactive Receive Mode (IRXM) Acknowledge If IRXM is enabled (<i>I2Cn_CTRL.rx_mode = 1</i>), this field determines if the hardware sends an ACK or a NACK to an IRM transaction. 0: Respond to IRXM with ACK 1: Respond to IRXM with NACK | |
| 3 | rx_mode | R/W | 0 | Interactive Receive Mode (IRXM) When receiving data, allows for an Interactive Receive Mode (IRM) interrupt event after each received byte of data. The I2Cn peripheral hardware can be enabled to send either an ACK or NACK for IRXM. See <i>Interactive Receive Mode</i> section for detailed information. 0: Disable 1: Enable <i>Note: Only set this field when the I2C bus is inactive.</i> | |

| I ² C Control | | | I2Cn_CTRL | | [0x0000] |
|--------------------------|---------------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 2 | gen_call_addr | R/W | 0 | General Call Address Enable 0: Ignore General Call Address 1: Acknowledge General Call Address | |
| 1 | mst | R/W | 0 | Master Mode Enable 0: Slave mode enabled. 1: Master mode enabled. | |
| 0 | i2c_en | R/W | 0 | I²C Peripheral Enable 0: Disabled 1: Enabled | |

Table 13-7: I²C Status Register

| I ² C Status | | | I2Cn_STATUS | | [0x0004] |
|-------------------------|----------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:6 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 5 | clk_mode | RO | 0 | Master Mode I²C Bus Transaction Active The peripheral is operating in Master mode and a valid transaction beginning with a START command is in progress on the I ² C bus. This bit will read 1 until the master ends the transaction with a STOP command. This bit will continue to read 1 while a slave performs clock stretching. 0: Device not actively driving SCL clock cycles. 1: Device operating as master and actively driving SCL clock cycles. | |
| 4 | tx_full | RO | 0 | TX FIFO Full 0: Not full 1: Full | |
| 3 | tx_empty | RO | 1 | TX FIFO Empty 0: Not empty 1: Empty | |
| 2 | rx_full | RO | 0 | RX FIFO Full 0: Not full 1: Full | |
| 1 | rx_empty | RO | 1 | RX FIFO Empty 0: Not empty 1: Empty | |
| 0 | bus | RO | 0 | Master or Slave Mode I²C Bus Transaction Active The peripheral is operating in Master or Slave mode and a valid transaction beginning with a START command is in progress on the I ² C bus. This bit will read 1 until the peripheral acting as a master or an external master ends the transaction with a STOP command. This bit will continue to read 1 while a slave performs clock stretching. 0: I ² C bus is idle. 1: I ² C bus transaction in progress. | |

Table 13-8: I²C Interrupt Flag 0 Register

| I ² C Interrupt Flag 0 | | | | I2Cn_INT_F0 | [0x0008] |
|-----------------------------------|----------------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:24 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 23 | wr_addr_match | R/W1C | 0 | Slave Write Address Match Interrupt Flag If set, the device has been accessed for a write (i.e. receive) transaction in slave mode and the address received matches the device slave address. 0: No address match. 1: Address match. | |
| 22 | rd_addr_match | R/W1C | 0 | Slave Read Address Match Interrupt Flag If set, the device has been accessed for a read (i.e. transmit) transaction in slave mode and the address received matches the device slave address. 0: No address match. 1: Address match. | |
| 21:16 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 15 | tx_lock_out | R/W1C | 0 | TX FIFO Locked Interrupt Flag If set, the TX FIFO is locked and writes to the TX FIFO are ignored. When set, the TX FIFO is automatically flushed. Writes to the TX FIFO are ignored until this flag is cleared. Write 1 to clear. 0: TX FIFO not locked. 1: TX FIFO is locked and all writes to the TX FIFO are ignored. | |
| 14 | stop_er | R/W1C | 0 | Out of Sequence STOP Interrupt Flag This flag is set if a STOP condition occurs out of expected sequence. Write 1 to clear this field. Writing 0 has no effect. 0: Error condition has not occurred. 1: Out of sequence STOP condition occurred. | |
| 13 | start_er | R/W1C | 0 | Out of Sequence START Interrupt Flag This flag is set if a START condition occurs out of expected sequence. Write 1 to clear this field. Writing 0 has no effect. 0: Error condition has not occurred. 1: Out of sequence START condition occurred. | |
| 12 | do_not_resp_er | R/W1C | 0 | Slave Mode Do Not Respond Interrupt Flag This occurs if an address match is made, but the TX FIFO or RX FIFO is not ready. Write 1 to clear this field. Writing 0 has no effect. 0: Error condition has not occurred. 1: I ² C address match has occurred and either the TX or RX FIFO is not configured. | |
| 11 | data_er | R/W1C | 0 | Master Mode Data NACK from External Slave Interrupt Flag This flag is set by hardware if a NACK is received from a slave. This flag is only valid if the I2Cn peripheral is configured for Master Mode operation. Write 1 to clear. Write 0 has no effect. 0: Error condition has not occurred. 1: Data NACK received from a slave. | |

| I ² C Interrupt Flag 0 | | | | I2Cn_INT_F0 | [0x0008] |
|-----------------------------------|---------------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 10 | addr_nack_er | R/W1C | 0 | Master Mode Address NACK from Slave Error Flag This flag is set by hardware if an Address NACK is received from a slave bus. This flag is only valid if the I2Cn peripheral is configured for Master Mode operation. Write 1 to clear. Write 0 has no effect. 0: Error condition has not occurred. 1: Address NACK received from a slave. | |
| 9 | to_er | R/ W1C | 0 | Timeout Error Interrupt Flag This occurs when this device holds SCL low longer than the programmed timeout value. Applies to both Master and Slave Mode. Write 1 to clear. Write 0 has no effect. 0: Timeout error has not occurred. 1: Timeout error occurred. | |
| 8 | arb_er | R/ W1C | 0 | Master Mode Arbitration Lost Interrupt Flag Write 1 to clear. Write 0 has no effect. 0: Condition has not occurred. 1: Condition occurred. | |
| 7 | addr_ack | R/ W1C | 0 | Master Mode Address ACK from External Slave Interrupt Flag This field is set when a slave address ACK is received. Write 1 to clear. Write 0 has no effect. 0: Condition has not occurred. 1: The slave device ACK for the address was received. | |
| 6 | stop | R/ W1C | 0 | Slave Mode STOP Condition Interrupt Flag This flag is set by hardware when a STOP condition is detected. Write 1 to clear. Write 0 has no effect. 0: Condition has not occurred. 1: Condition occurred. | |
| 5 | tx_thresh | RO | 1 | TX FIFO Threshold Level Interrupt Flag This field is set by hardware if the number of bytes in the Transmit FIFO is less than or equal to the Transmit FIFO threshold level. Write 1 to clear. This field is automatically cleared by hardware when the TX FIFO contains fewer bytes than the TX threshold level. 0: TX FIFO contains more bytes than the TX threshold level. 1: TX FIFO contains TX threshold level or fewer bytes (Default). | |
| 4 | rx_thresh | R/W1C | 1 | RX FIFO Threshold Level Interrupt Flag This field is set by hardware if the number of bytes in the Receive FIFO is greater than or equal to the Receive FIFO threshold level. This field is automatically cleared when the RX FIFO contains fewer bytes than the RX threshold setting. 0: RX FIFO contains fewer bytes than the RX threshold level. 1: RX FIFO contains at least RX threshold level of bytes (Default). | |
| 3 | addr_match | R/W1C | 0 | Slave Mode Incoming Address Match Status Interrupt Flag Write 1 to clear. Writing 0 has no effect. 0: Slave address match has not occurred. 1: Slave address match occurred. | |
| 2 | gen_call_addr | R/W1C | 0 | Slave Mode General Call Address Match Received Interrupt Flag Write 1 to clear. Writing 0 has no effect. 0: General call address match has not occurred. 1: General call address match occurred. | |

| I ² C Interrupt Flag 0 | | | | I2Cn_INT_F0 | [0x0008] |
|-----------------------------------|---------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 1 | rx_mode | R/W1C | 0 | Interactive Receive Mode Interrupt Flag Write 1 to clear. Writing 0 is ignored. 0: Interrupt condition has not occurred. 1: Interrupt condition occurred. | |
| 0 | done | R/W1C | 0 | Transfer Complete Interrupt Flag This flag is set for both Master and Slave mode once a transaction completes. Write 1 to clear. Writing 0 has no effect. 0: Transfer is not complete. 1: Transfer complete. | |

Table 13-9: I²C Interrupt Enable 0 Register

| I ² C Interrupt Enable 0 | | | | I2Cn_INT_EN0 | [0x000C] |
|-------------------------------------|----------------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:24 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 23 | wr_addr_match | R/W | 0 | Slave Write Address Match Interrupt Enable This bit is set to enable interrupts when the device is accessed in slave mode and the address received matches the device slave addressed for a write transaction. 0: Disabled. 1: Enabled. | |
| 22 | rd_addr_match | R/W | 0 | Slave Read Address Match Interrupt Enable This bit is set to enable interrupts when the device is accessed in slave mode and the address received matches the device slave addressed for a read transaction. 0: Disabled. 1: Enabled. | |
| 21:16 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 15 | tx_lock_out | R/W | 0 | TX FIFO Lock Out Interrupt Enable 0: Disabled. 1: Enabled. | |
| 14 | stop_er | R/W | 0 | Out of Sequence STOP Condition Detected Interrupt Enable 0: Disabled. 1: Enabled. | |
| 13 | start_er | R/W | 0 | Out of Sequence START Condition Detected Interrupt Enable 0: Disabled. 1: Enabled. | |
| 12 | do_not_resp_er | R/W | 0 | Slave Mode Do Not Respond Interrupt Enable Set this field to enable interrupts in Slave Mode when the “Do Not Respond” condition occurs. 0: Interrupt disabled. 1: Interrupt enabled. | |
| 11 | data_er | R/W | 0 | Master Mode Received Data NACK from Slave Interrupt Enable 0: Disabled. 1: Enabled. | |

| I ² C Interrupt Enable 0 | | | | I2Cn_INT_EN0 | [0x000C] |
|-------------------------------------|---------------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 10 | addr_nack_er | R/W | 0 | Master Mode Received Address NACK from Slave Interrupt Enable 0: Disabled. 1: Enabled. | |
| 9 | to_er | R/W | 0 | Timeout Error Interrupt Enable 0: Disabled. 1: Enabled. | |
| 8 | arb_er | R/W | 0 | Master Mode Arbitration Lost Interrupt Enable 0: Disabled. 1: Enabled. | |
| 7 | addr_ack | R/W | 0 | Received Address ACK from Slave Interrupt Enable Set this field to enable interrupts for Master Mode slave device address ACK events. 0: Interrupt disabled. 1: Interrupt enabled. | |
| 6 | stop | R/W | 0 | STOP Condition Detected Interrupt Enable 0: Disabled. 1: Enabled. | |
| 5 | tx_thresh | R/W | 0 | TX FIFO Threshold Level Interrupt Enable 0: Disabled. 1: Enabled. | |
| 4 | rx_thresh | R/W | 0 | RX FIFO Threshold Level Interrupt Enable 0: Disabled. 1: Enabled. | |
| 3 | addr_match | R/W | 0 | Slave Mode Incoming Address Match Interrupt Enable 0: Disabled. 1: Enabled. | |
| 2 | gen_call_addr | R/W | 0 | Slave Mode General Call Address Match Received Interrupt Enable 0: Disabled. 1: Enabled. | |
| 1 | rx_mode | R/W | 0 | Interactive Receive Interrupt Enable 0: Disabled. 1: Enabled. | |
| 0 | done | R/W | 0 | Transfer Complete Interrupt Enable 0: Disabled. 1: Enabled. | |

 Table 13-10: I²C Interrupt Flag 1 Register

| I ² C Interrupt Status Flags 1 | | | | I2Cn_INT_FL1 | [0x0010] |
|---|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:3 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 2 | start | R/W1C | 0 | START Condition Status Flag If set, a device START condition has been detected. 0: START condition not detected. 1: START condition detected. | |

| I ² C Interrupt Status Flags 1 | | | | I2Cn_INT_FL1 | [0x0010] |
|---|--------------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 1 | tx_underflow | R/W1C | 0 | Slave Mode: TX FIFO Underflow Status Flag In Slave Mode operation, the hardware sets this flag automatically if the TX FIFO is empty and the master requests more data by sending an ACK after the previous byte transferred. 0: Slave mode TX FIFO underflow condition has not occurred. 1: Slave mode TX FIFO underflow condition occurred. | |
| 0 | rx_underflow | R/W1C | 0 | Slave Mode: RX FIFO Overflow Status Flag In Slave Mode operation, the hardware sets this flag automatically when an RX FIFO overflow occurs. Write 1 to clear. Writing 0 has no effect. 0: Slave mode RX FIFO overflow event has not occurred. 1: Slave mode RX FIFO overflow condition occurred (data lost). | |

Table 13-11: I²C Interrupt Enable 1 Register

| I ² C Interrupt Enable 1 | | | | I2Cn_INT_EN1 | [0x0014] |
|-------------------------------------|--------------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:3 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 2 | start | R/W | 0 | START Condition Interrupt Enable 0: Disabled. 1: Enabled. | |
| 1 | tx_underflow | R/W | 0 | Slave Mode TX FIFO Underflow Interrupt Enable 0: Disabled. 1: Enabled. | |
| 0 | rx_underflow | R/W | 0 | Slave Mode RX FIFO Overflow Interrupt Enable 0: Disabled. 1: Enabled. | |

Table 13-12: I²C FIFO Length Register

| I ² C FIFO Length | | | | I2Cn_FIFO_LEN | [0x0018] |
|------------------------------|--------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 15:8 | tx_len | RO | 8 | TX FIFO Length Returns the length of the TX FIFO. 8: 8-byte TX FIFO. | |
| 7:0 | rx_len | RO | 8 | RX FIFO Length Returns the length of the RX FIFO. 8: 8-byte RX FIFO. | |

Table 13-13: I²C Receive Control 0 Register

| I ² C Receive Control 0 | | | I2Cn_RX_CTRL0 | | [0x001C] |
|------------------------------------|-----------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:12 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 11:8 | rx_thresh | R/W | 0 | RX FIFO Threshold Level Set this field to the required number of bytes to trigger a RX FIFO threshold event. When the number of bytes in the RX FIFO is equal to or greater than this field, the hardware sets the <i>I2Cn_INT_FLO.rx_thresh</i> bit indicating an RX FIFO threshold level event. 0: 0 bytes or more in the RX FIFO causes a threshold event. 1: 1+ bytes in the RX FIFO triggers a receive threshold event (recommended minimum value). ... 8: RX FIFO threshold event only occurs when the RX FIFO is full. | |
| 7 | rx_flush | R/W1O | 0 | Flush RX FIFO Write 1 to this field to initiate a RX FIFO flush, clearing all data in the RX FIFO. This field is automatically cleared by hardware when the RX FIFO flush completes. Writing 0 has no effect. 0: RX FIFO flush complete or not active. 1: Flush the RX FIFO | |
| 6:1 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 0 | dnr | R/W | 0 | Slave Mode Do Not Respond Slave mode operation only. If the device has been addressed for a write operation, and there is still data in the RX_FIFO then: 0: Always respond to an address match with an ACK but then always respond to data bytes with a NACK. 1: NACK the address. | |

Table 13-14: I²C Receive Control 1 Register

| I ² C Receive Control 1 | | | I2Cn_RX_CTRL1 | | [0x0020] |
|------------------------------------|---------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:12 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 11:8 | rx_fifo | R | 0 | RX FIFO Byte Count Status 0: No data in the RX FIFO. 1: 1 byte in the RX FIFO. 2: 2 bytes in the RX FIFO. 3: 3 bytes in the RX FIFO. 4: 4 bytes in the RX FIFO. 5: 5 bytes in the RX FIFO. 6: 6 bytes in the RX FIFO. 7: 7 bytes in the RX FIFO. 8: 8 bytes in the RX FIFO (max value). | |

| I ² C Receive Control 1 | | | I2Cn_RX_CTRL1 | | [0x0020] |
|------------------------------------|--------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 7:0 | rx_cnt | R/W | 1 | RX FIFO Transaction Byte Count Configuration When in Master Mode, write the number of bytes to be received in a transaction from 1 to 256. 0x00 represents 256. 0: 256 byte receive transaction. 1: 1 byte receive transaction. 2: 2 byte receive transaction. ... 255: 255 byte receive transaction. <i>This field is ignored when I2Cn_CTRL.rx_mode = 1. To receive more than 256 bytes, use I2Cn_CTRL.rx_mode = 1</i> | |

Table 13-15: I²C Transmit Control 0 Register

| I ² C Transmit Control 0 | | | I2Cn_TX_CTRL0 | | [0x0024] |
|-------------------------------------|-------------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:12 | - | RO | 0 | Reserved | |
| 11:8 | tx_thresh | R/W | 0 | TX FIFO Threshold Level Sets the level for a Transmit FIFO threshold event interrupt. If the number of bytes remaining in the TX FIFO falls to this level or lower the interrupt flag <i>I2Cn_INT_FLO.tx_thresh</i> is set indicating a TX FIFO Threshold Event occurred. 0: 0 bytes remaining in the TX FIFO triggers a TX FIFO threshold event. 1: 1 byte or fewer remaining in the TX FIFO triggers a TX FIFO threshold event (recommended minimum value). ... 7: 7 or fewer bytes remaining in the TX FIFO triggers a TX FIFO threshold event | |
| 7 | tx_flush | R/W1O | 0 | TX FIFO Flush A TX FIFO flush clears all remaining data from the transmit FIFO. 0: TX FIFO flush is complete or not active. 1: Flush the TX FIFO <i>Note: Hardware automatically clears this bit to 0 after it is written to 1 when the flush is completed.</i> <i>If I2Cn_INT_FLO.tx_lock_out = 1, then I2Cn_TX_CTRL0.tx_flush = 1.</i> | |
| 6 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 5 | tx_nack_afd | R/W | 0 | TX FIFO received NACK Auto Flush Disable Various situations or conditions are described in this user guide that lead to the Transmit FIFO being flushed and locked out (<i>I2Cn_INT_FLO.tx_lock_out</i> = 1). 0: Received NACK at end of Slave Transmit operation enabled 1: Received NACK at end of Slave Transmit operation disabled. <i>Note: upon entering TX Preload Mode, hardware will automatically set this bit to 0. Software can subsequently set to any value desired (i.e. Hardware does not continuously force the bitfield to this value).</i> | |

| I ² C Transmit Control 0 | | | | I2Cn_TX_CTRL0 | [0x0024] |
|-------------------------------------|---------------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 4 | tx_amr_afd | R/W | 0 | TX FIFO Slave Address Match Read Auto Flush Disable Various situations or conditions are described in this user guide that lead to the Transmit FIFO being flushed and locked out (<i>I2Cn_INT_FLO.tx_lock_out</i> = 1). 0: Enabled. 1: Disabled. <i>Note: upon entering TX Preload Mode, hardware will automatically set this bit to 1. Software can subsequently set to any value desired (i.e. Hardware does not continuously force the bitfield to this value).</i> | |
| 3 | tx_amw_afd | R/W | 0 | TX FIFO Slave Address Match Write Auto Flush Disable Various situations or conditions are described in this user guide that lead to the Transmit FIFO being flushed and locked out (<i>I2Cn_INT_FLO.tx_lock_out</i> = 1). 0: Enabled. 1: Disabled. <i>Note: upon entering TX Preload Mode, hardware will automatically set this bit to 1. Software can subsequently set to any value desired (i.e. Hardware does not continuously force the bitfield to this value).</i> | |
| 2 | tx_amgc_afd | R/W | 0 | TX FIFO General Call Address Match Auto Flush Disable Various situations or conditions are described in this user guide that lead to the Transmit FIFO being flushed and locked out (<i>I2Cn_INT_FLO.tx_lock_out</i> = 1). 0: Enabled. 1: Disabled. <i>Note: upon entering TX Preload Mode, hardware will automatically set this bit to 1. Software can subsequently set to any value desired (i.e. Hardware does not continuously force the bitfield to this value).</i> | |
| 1 | tx_ready_mode | R/W | 0 | TX FIFO Ready Manual Mode 0: Hardware will control <i>I2Cn_TX_CTRL1.tx_ready</i> 1: Software control of <i>I2Cn_TX_CTRL1.tx_ready</i> | |
| 0 | tx_preload | R/W | 0 | TX FIFO Preload Mode Enable 0: Normal operation. An address match in Slave Mode, or a General Call address match, will flush and lock the TX FIFO so it cannot be written and set <i>I2Cn_INT_FLO.tx_lock_out</i> . 1: TX FIFO Preload Mode. An address match in Slave Mode, or a General Call address match, will not lock the TX FIFO and will not set <i>I2Cn_INT_FLO.tx_lock_out</i> . This allows firmware to preload data into the TX FIFO. The status of the I ² C is controllable at <i>I2Cn_TX_CTRL1.tx_ready</i> . | |

Table 13-16: I²C Transmit Control 1 Register

| I ² C Transmit Control Register 1 | | | | I2Cn_TX_CTRL1 | [0x0028] |
|--|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:12 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |

| I ² C Transmit Control Register 1 | | | I2Cn_TX_CTRL1 | | [0x0028] |
|--|----------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 11:8 | tx_fifo | R | 0 | Transmit FIFO Byte Count Status 0: No data in the TX FIFO. 1: 1 byte in the TX FIFO. 2: 2 bytes in the TX FIFO. 3: 3 bytes in the TX FIFO. 4: 4 bytes in the TX FIFO. 5: 5 bytes in the TX FIFO. 6: 6 bytes in the TX FIFO. 7: 7 bytes in the TX FIFO. 8: 8 bytes in the TX FIFO (max value). | |
| 7:1 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 0 | tx_ready | R/1 | 1 | Transmit FIFO Preload Ready Status When TX FIFO Preload Mode is enabled, <i>I2Cn_TX_CTRL0.tx_preload</i> = 1, this bit is automatically cleared to 0. While this bit is 0, if the I2Cn hardware receives a slave address match a NACK is sent. Once the I2Cn hardware is ready (firmware has preloaded the TX FIFO, configured the DMA, etc.) application firmware must set this bit to 1 so the I2Cn hardware will send an ACK on a slave address match. When TX FIFO Preload Mode is disabled, <i>I2Cn_TX_CTRL0.tx_preload</i> = 1, this bit is forced to 1 and the I2Cn hardware behaves normally. | |

Table 13-17: I²C Data Register

| I ² C Data | | | I2Cn_FIFO | | [0x002C] |
|-----------------------|-------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 7:0 | data | R/W | 0xFF | I²C FIFO Data Register Reads from this register pops data off the RX FIFO. Writes to this register pushes data onto the TX FIFO. Reading from an empty RX FIFO returns 0xFF. Writes to a full TX FIFO are ignored. | |

Table 13-18: I²C Master Control Register

| I ² C Master Control | | | I2Cn_MASTER_CTRL | | [0x0030] |
|---------------------------------|------------|--------|------------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:11 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 10:8 | mcode | R/W | 0 | MCODE These bits set the master code used in Hs-mode operation | |
| 7 | sl_ex_addr | R/W | 0 | Slave Extended Addressing Enable 0: Send a 7-bit address to the slave 1: Send a 10-bit address to the slave | |
| 6:3 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |

| I ² C Master Control | | | I2Cn_MASTER_CTRL | | [0x0030] |
|---------------------------------|---------|--------|------------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 2 | stop | R/W1O | 0 | Send STOP Condition 1: Send a STOP Condition at the end of the current transaction. <i>Note: This bit is automatically cleared by hardware when the STOP condition begins.</i> | |
| 1 | restart | R/W1O | 0 | Send Repeated START Condition After sending data to a slave, the master may send another START to retain control of the bus. 1: Send a Repeated START condition to Slave instead of sending a STOP condition at the end of the current transaction. <i>Note: This bit is automatically cleared by hardware when the repeated START condition begins.</i> | |
| 0 | start | R/W1O | 0 | Start Master Mode Transfer 1: Start Master Mode Transfer <i>Note: This bit is automatically cleared by hardware when the transfer is completed or aborted.</i> | |

Table 13-19: I²C SCL Low Control Register

| I ² C Clock Low Control | | | I2Cn_CLK_LO | | [0x0034] |
|------------------------------------|--------|--------|-------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:9 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 8:0 | scl_lo | R/W | 0x001 | Clock Low Time In Master Mode, this configures the SCL low time. $t_{SCL_LO} = f_{I2C_CLK} \times (scl_lo + 1)$ <i>Note: 0 is not a valid setting for this field.</i> | |

Table 13-20: I²C SCL High Control Register

| I ² C Clock High Control | | | I2Cn_CLK_HI | | [0x0038] |
|-------------------------------------|--------|--------|-------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:9 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 8:0 | scl_hi | R/W | 0x001 | Clock High Time In Master Mode, this configures the SCL high time. $t_{SCL_HI} = 1/f_{I2C_CLK} \times (scl_hi + 1)$ In both Master and Slave Mode, this also configures the time SCL is held low after new data is loaded from the TX FIFO or after firmware clears I2Cn_INT_FLO.rx_mode during Interactive Receive Mode. <i>Note: 0 is not a valid setting for this field.</i> | |

Table 13-21: I²C Hs-Mode Clock Control Register

| I ² C Hs-Mode Clock Control | | I2Cn_HS_CLK | | | [0x003C] |
|--|-----------|-------------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | R/W | 0 | Reserved for Future Use Do not modify | |
| 15:8 | hs_clk_hi | R/W | 0 | Hs-Mode Clock High Time This field sets the Hs-Mode clock high count. In Slave mode, this is the time SCL is held high after data is output on SDA. <i>Note: See section 13.4.4 SCL Clock Generation for Hs-mode for details on the requirements for the Hs-mode clock high and low times.</i> | |
| 7:0 | hs_clk_lo | R/W | 0 | Hs-Mode Clock Low Time This field sets the Hs-Mode clock low count. In Slave mode, this is the time SCL is held low after data is output on SDA. <i>Note: See section 13.4.4 SCL Clock Generation for Hs-mode for details on the requirements for the Hs-mode clock high and low times.</i> | |

 Table 13-22: I²C Timeout Register

| I ² C Timeout | | I2Cn_TIMEOUT | | | [0x0040] |
|--------------------------|-------|--------------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved | |
| 15:0 | to | R/W | 0 | Bus Error SCL Timeout Period Set this value to the number of I ² C clock cycles desired to cause a bus timeout error. The peripheral timeout timer starts when it pulls SCL low. After the peripheral releases the line, if the line is not pulled high prior to the timeout number of I ² C clock cycles, a bus error condition is set (<i>I2Cn_INT_FLO.to_er = 1</i>) and the peripheral releases the SCL and SDA lines 0: Timeout disabled. All other values result in a timeout calculation of: $t_{BUS_TIMEOUT} = \frac{1}{f_{I2C_CLK}} \times to$ <i>Note: The timeout counter monitors the I2Cn peripheral's driving of the SCL pin, not an external I²C device driving the SCL pin.</i> | |

 Table 13-23: I²C DMA Register

| I ² C DMA | | | I2Cn_DMA | | [0x0048] |
|----------------------|-------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:2 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 1 | rx_en | R/W | 0 | RX DMA Channel Enable 0: Disable 1: Enable | |
| 0 | tx_en | R/W | 0 | TX DMA Channel Enable 0: Disable 1: Enable | |

Table 13-24: I²C Slave Address Register

| I ² C Slave Address | | | I2Cn_SLAVE_ADDR | | [0x004C] |
|--------------------------------|------------|--------|-----------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 15 | ex_addr | R/W | 0 | Slave Mode Extended Address Length Select 0: 7-bit addressing. 1: 10-bit addressing. | |
| 14:10 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 9:0 | slave_addr | R/W | 0 | Slave Mode Slave Address In Slave Mode Operation, (<i>I2Cn_CTRL.mstr</i> = 0), set this field to the slave address for the I2Cn port. For 7-bit addressing, the address occupies the least significant 7 bits. For 10-bit addressing, the 9-bits of address occupies the most significant 9 bit and the R/W bit occupies the least significant bit. <i>Note: I2Cn_SLAVE_ADDR.ex_addr controls if this field is a 7-bit or 10-bit address.</i> | |

14. Inter-Integrated Sound Interface (I²S)

I²S (Inter-IC Sound) is a serial audio interface for communicating PCM data information between devices.

Both master and slave modes are supported. The peripheral provides separate data in and data out signals supporting full-duplex communication for higher throughput.

Key features:

1. Supports both master and slave modes
2. Stereophonic (2 channel) and monophonic (left or right channel option) formats
3. Each channel supports full duplex operations
4. Two DMA requests for each I²S channel can be connected (one to the RX FIFO, one to the TX FIFO)
5. Flexible timing
 - a. Configurable sampling rate from $1/65536$ to 1 of the I²S input clock
 - b. BCLK source selection in Master and Slave mode
 - c. LRCLK source selection in Master and Slave mode
6. Flexible data format
 - a. Programmable word size (byte, half word and word) can be saved to memory
 - b. Number of bits per data word can be selected from 1 to 32 (typically 8, 16, 24, 32-bit data frame)
 - c. Word select polarity control, not in I²S Specification
 - d. First bit position selection, not in I²S Specification
 - e. MSB/LSB aligned, not in I²S Specification
 - f. Sign extended control, not in I²S Specification
7. Full-duplex serial communication with separate I²S Data In (SDI) and I²S Data Out (SDO) pins.

14.1 Instances

Table 14-1: MAX78000 I²S Instances

| Instance | Supported Channels | I ² S_CLK Clock Options | | RX FIFO Depth | TX FIFO Depth |
|----------|--------------------|------------------------------------|------------------------|---------------|---------------|
| I2S0 | Stereo | PCLK | IS2_CLKEXT (P0.14 AF2) | 8 | 8 |

Table 14-2: MAX78000 I²S Pin Mapping

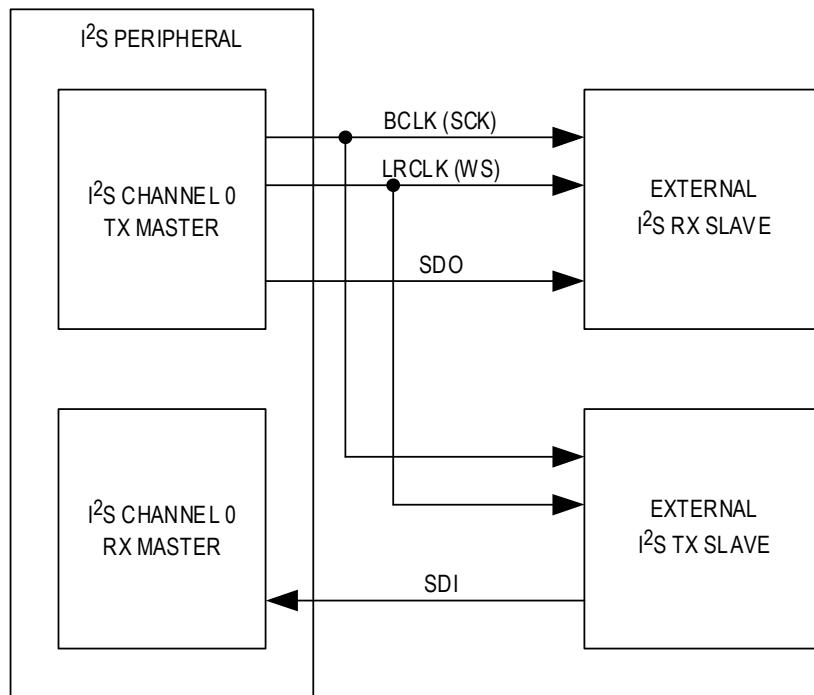
| Instance | I ² S Signal | 81 CTBGA | |
|----------|-------------------------|----------|--------------------|
| | | Pin | Alternate Function |
| I2S0 | BCLK (SCK) | P1.2 | AF1 |
| | LRCLK (WS) | P1.3 | AF1 |
| | SDI | P1.4 | AF1 |
| | SDO | P1.5 | AF1 |

14.2 Details

The I²S supports full-duplex serial communication with separate I²S data in (SDI) and data out (SDO) pins. [Figure 14-1: Full Duplex Connection, I²S Master Mode](#) shows an interconnect between a peripheral configured in host mode, communicating

with an external I²S slave receiver and an external I²S transmitter. In master mode the BCLK and LRCLK signals are generated by the peripheral and sent to each slave device.

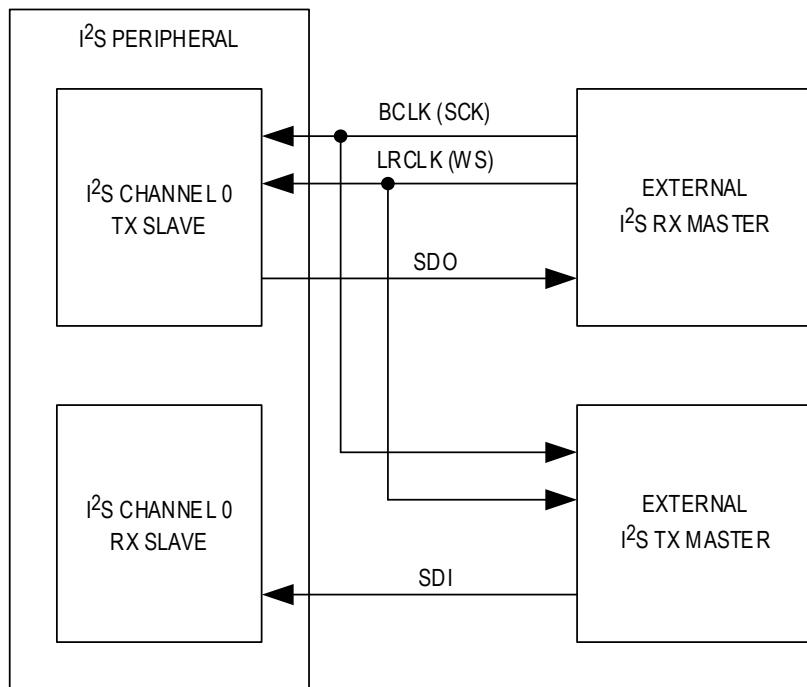
Figure 14-1: Full Duplex Connection, I²S Master Mode



Note: When in Master mode, LRCLK (WS) and BCLK(SCK) signals are generated by the MAX78000's I²S peripheral.

Figure 14-2: Full Duplex Connection, I²S Slave Mode shows a peripheral configured in slave mode. An external I²S system clock, from one of the external I²S master devices, or an independent source, provides BCLK and LRCLK signals to the I²S peripheral.

Figure 14-2: Full Duplex Connection, I²S Slave Mode



Note: When operating in slave mode, the LRCLK (WS) and BCLK (SCK) signals are generated externally and are inputs to the MAX78000 I²S peripheral.

14.3 I²S Clocking

I²S communication is synchronized by two clocks: the bit clock (BCLK, also known as SCK) and the left/right channel clock (LRCLK, also known as WS.) In master mode the internal timing generator creates the BCLK and LRCLK signals. The pins are configured as outputs to supply the clock to slave devices.

In slave mode the BCLK and LRCLK pins are configured as inputs. An external master generates the BCLK and LRCLK signals which the peripheral uses to synchronize itself to the I²S bus.

A typical bit clock rate is between 512kHz for 8kHz sampling, and 12.288MHz for 192kHz sampling rate. [Equation 14-1](#) shows the bit clock rate for CD stereo audio using a sample frequency of 44.1kHz and 16-bits per sample.

Equation 14-1: CD Audio Bit Frequency Calculation

$$f_{\text{BCLK}} = 44.1\text{kHz} \times 16 \times 2 \Rightarrow 1.4122\text{MHz}$$

Clock switching must be done while the I²S clock is disabled.

14.3.1 BCLK Generation for Master

The required BCLK frequency is determined using:

1. Audio sample frequency
2. Number of bits per sample
3. Number of channels (2 for stereo)

Equation 14-2, below, shows the formula to determine the target BCLK frequency.

Equation 14-2: BCLK Target Frequency based on Audio Frequency, Bits per Word and Channels

$$f_{BCLK} = f_{SAMPLE} \times \left(\frac{\text{Bits}}{\text{Sample}} \right) \times \text{Number of Channels}$$

Calculate BCLK frequency using *Equation 14-3* using the selected I2S_CLK from *Table 14-1*.

Equation 14-3: BLCK Frequency Equation for the I²S Peripheral

$$f_{BCLK} = \frac{f_{I2S_CLK}}{(I2Sn_CTRL1CH0.clkdiv + 1)}$$

14.3.2 LRCLK Generation

An I²S data stream can carry monophonic (either left or right channel) or stereophonic (one left and one right channel) data. The audio sample rate is determined by the LRCLK frequency.

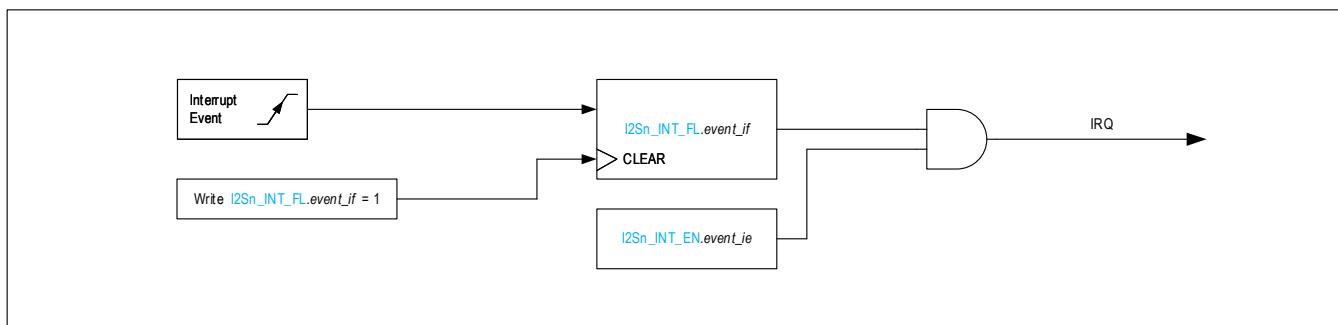
LRCLK selects the right or left data channel. The polarity of the LRCLK is programmable. Left channel data is transferred when LRCLK is low, and right channel data transferred when LRCLK is high.

$$f_{LRCLK} = \frac{f_{BCLK}}{(1 + \text{bits_word})}$$

14.4 Interrupt Events

The peripheral generates interrupts for the events shown in *Table 14-3: I2S Interrupt Events*. Unless noted otherwise, each instance has its own independent set of interrupt flag and enable fields. *Figure 14-3: I2S Interrupt Functional Diagram* shows how the interrupts work.

Figure 14-3: I²S Interrupt Functional Diagram



Unless stated otherwise hardware sets interrupt flags on the occurrence of on the event (“edge-triggered”), rather than the condition (“level-triggered”). Multiple events may set the same flag. An interrupt request (IRQ) will be generated if the corresponding interrupt enable field is set. The interrupt flags remain set until cleared by software, usually in their respective interrupt service routine (ISR).

Table 14-3: I²S Interrupt Events (Channel 0 Example)

| Event | Local Interrupt Flag | Local Interrupt Enable |
|--------------------|------------------------------|------------------------------|
| RX Overrun | <i>I2Sn_INTFL.rx_ov_ch0</i> | <i>I2Sn_INTEN.rx_ov_ch0</i> |
| RX Threshold | <i>I2Sn_INTFL.rx_thd_ch0</i> | <i>I2Sn_INTEN.rx_thd_ch0</i> |
| TX FIFO Half-Empty | <i>I2Sn_INTFL.tx_he_ch0</i> | <i>I2Sn_INTEN.tx_he_ch0</i> |

14.4.1 RX FIFO Overrun

An RX overrun event occurs if the number of data words in a FIFO (*I2Sn_DMACH0.rx_lvl*) is equal to 8 and a valid character has been shifted in. The received character is discarded, and the FIFO remains unchanged.

14.4.2 RX FIFO Threshold

An RX FIFO threshold event occurs when a valid character is shifted in and the number of data words in an RX FIFO (*I2Sn_DMACH0.rx_lvl*) increments from *.rx_thd -1* to *.rx_thd*. The event does not occur if the opposite transition occurs.

14.4.3 TX FIFO Half-Empty (Fixed)

A TX FIFO half-empty event occurs when the number of bytes in the TX FIFO (*I2Sn_DMACH0.tx_lvl*) transitions from:

$$\frac{RX\ FIFO\ DEPTH}{2} \xrightarrow{\text{transitions to}} \left(\frac{RX\ FIFO\ DEPTH}{2}\right) + 1$$

14.5 Wakeup Events

The peripheral does not support wakeup events from low power modes.

14.6 Direct Memory Access

The I²S supports DMA transfers in both the transmit and receive directions; separate DMA channels can be connected to the RX and TX FIFO buffers. The followings describe the behavior of the RX and TX DMA requests when the DMA function is enabled.

1. RX DMA request is asserted when the number of valid bytes in the RX FIFO is greater than or equal to the RX FIFO threshold.
2. TX DMA request is asserted when the number of valid bytes in the TX FIFO is less than the TX FIFO threshold.
3. The IP support DMA transfers in both the transmit and receive directions; separate DMA channels can be connected to the RX and TX FIFO buffers. The interrupts are not rerouted to the DMA controller and must still be managed by the CPU in the usual manner.

14.7 Block Operation

After exiting a power-on reset, the IP is disabled by default. It must be enabled and configured by software to establish the I²S serial communication. A typical software sequence is shown below.

1. Clear *PERCKCN1.i2sd* to 0 to enable the I²S clock source shown in *Table 14-1: MAX78000 I²S Instances*.
2. Wait for all transmissions and receptions to complete based on the customer-implemented protocol.
3. Disable the peripheral clock.
4. Set *I2Sn_CTRLCHO.rst* to 1 to clear the remaining value in the counter.
5. Set *I2Sn_CTRLCHO.flush* to 1 to flush the FIFO buffers.
6. Configure the *I2Sn_CTRLCHO.ch_mode* to select master or slave configuration.
7. If in master mode configure the baud rate by programming the *I2Sn_CTRLCHO* register.
8. Configure the threshold of the RX FIFO by programming the *I2Sn_CTRLCHO.rx_thd*. The threshold of the TX FIFO is a fixed value, which is half of the TX FIFO depth.
9. Enable DMA function by programming *I2Sn_DMACHO* register if DMA function is needed.
10. Enable interrupt function by programming *I2Sn_INTEN* register if interrupt function is needed.
11. Program the *clkdiv* bits in *I2Sn_CTRLCHO* register for the new baudrate.
12. Re-enable the baud clock by setting *I2Sn_CTRLCHO.en*.
13. Start I²S operations.

14.8 I²S Master/Slave Configuration

Both master and slave modes are supported. In master mode the BCLK and LRCK signals are generated internally and output on the BCLK and LRCLK pins. In slave mode the BCLK and LRCLK pins are configured as inputs and the peripheral timing is determined by the external master clock source.

Table 14-4: I²S Channel mode Configuration (Channel 0 Example)

| Mode | <i>I2Sn_CTRLCHO.ch_mode</i> | <i>I2Sn_CTRLCHO.ext_sel</i> | LRCLK | BCLK |
|--------|-----------------------------|-----------------------------|-------------------|-------------------|
| Master | 0b00 | x | Output to Slave | Output to Slave |
| Slave | 0b11 | 0 | Input from Master | Input from Master |

14.9 Data Formatting

Data is signed two's complement, with controllable data direction allowing users to select MSB first or LSB first. The data frame can have arbitrary number of bits per channel.

The audio data word is programmable from 1 to 32 bits long (typically 8, 16, 24, and 32 bits). The transfer size, I²S data frame can be from 1 to 32 bits, is set by writing to the *I2Sn_CTRLCHO.wsizer* register. For word sizes less than 32 bits, the data frame may still be 64 bits and the unused bits are driven low by the transmitting device.

TX data is written to the SDO pin on the falling edge of BCLK, and RX data is read from the SDI pin on the rising edge of BCLK. The most significant bit (MSB) is always transmitted first.

14.9.1 Word Select Polarity

The Word Select polarity can be configured by the *I2Sn_CTRLCHO.wspol* register. By default, LRCLK low is for the left channel, high is for the right channel. Setting the register bit will invert the LRCLK signal definition.

14.9.2 MSB Location Control

By default the first bit of the transmit data is located at the second BCLK cycle as required by the I²SBUS specification. Optionally the MSB of the transmit data can be configured to be at the first BCLK cycle of the data word via [*I2Sn_CTRLCHO.msb_loc*](#). The waveform shows the transmit data with the register bit set.

14.9.3 Data Alignment

The I²S data can be left aligned (aligned to the MSB) or right aligned (aligned to the LSB) via the [*I2Sn_CTRLCHO.align*](#) field.

Left aligned: [*I2Sn_CTRLCHO.align*](#) = 0

- Number of data bits (BCLK pulses) per half-frame is higher than the FIFO data width
- RX: All bits after LSB of FIFO data width will be discarded
- TX: All bits after LSB of FIFO data width will be sent as 0
- Number of data bits (BCLK pulses) per half-frame is lower than the FIFO data width
- RX & TX: Lower bits of TX and RX FIFO will be removed first

Right aligned: [*I2Sn_CTRLCHO.align*](#) = 1

- Number of data bits (samples) per half-frame is higher than the FIFO data width
- RX: All bits before MSB of the sample value will be discarded
- TX: All bits before MSB of the sample value will be 0
- Number of data bits (samples) per half-frame is lower than the FIFO data width
- The data will be sign extended and saved to FIFO
- Lower bits of TX FIFO will be removed first, same as left aligned case

14.9.4 FIFO Word Control

The data width of the TX/RX FIFO can be configured to be byte, half word, or word size by using the [*I2Sn_CTRLCHO.wsize*](#) field. The data always begins with left channel first from the lowest address. The tables below describe the data ordering based on the [*I2Sn_CTRLCHO.wsize*](#) setting.

When a channel is reconfigured for a new setup, the FIFOs must be flushed, and the channel is reset by setting the [*I2Sn_CTRLCHO.rst*](#) field to 1.

Table 14-5: Byte Ordering when [*I2Sn_CTRLCHO.wsize*](#) = 0 (Byte)

| Address | MSByte | | | LSByte |
|----------------------|-------------------------|------------------------|---------------------------|--------------------------|
| Low Address | Right Channel Byte 1 | Left Channel Byte 1 | Right Channel Byte 0 | Left Channel Byte 0 |
| Low Address + 4 | Right Channel Byte 3 | Left Channel Byte 3 | Right Channel Byte 2 | Left Channel Byte 2 |
| | ... | ... | ... | ... |
| High Address + (n+1) | Right Channel Byte n | Left Channel Byte n | Right Channel Byte n-1 | Left Channel Byte n-1 |

Table 14-6: Half-Word Ordering when $I2Sn_CTRLOCHO.wsiz = 1$

| Address | MSByte | LSByte |
|--------------|---------------------------|--------------------------|
| Low Address | Right Channel Half-Word 0 | Left Channel Half Word 0 |
| | Right Channel Half-Word 1 | Left Channel Half Word 1 |
| ... | ... | ... |
| High Address | Right Channel Half Word n | Left Channel Half Word n |

Table 14-7: Word Ordering when $I2Sn_CTRLOCHO.wsiz = 2$ or 3

| Address | Data Word |
|--------------|----------------------|
| Low Address | Left Channel Word 0 |
| | Right Channel Word 0 |
| | Left Channel Word 1 |
| | Right Channel Word 1 |
| | ... |
| High Address | |

14.9.5 Samples Data

Table 14-8: Number of BCLK per Half Frame and Sample Number shows the relationship between the Number of BCLKs per half frame and the number of samples per half frame. *Equation 14-4* shows the required relationship between the Number of Samples and Number of BCLK per Half Frame.

Equation 14-4: Number of Samples Relationship to Number of BCLK

$$\frac{\# \text{Samples}}{\text{Half Frame}} \leq \frac{\# \text{BCLK}}{\text{Half Frame}}$$

The $I2Sn_CTRL1CHO.bits_word$ column in *Table 14-8, below*, is determined by $\frac{\# \text{BCLK}}{\text{Half Frame}} - 1$.

The $I2Sn_CTRL1CHO.smp_size$ column is the sample number captured from the I²S bus and is determined by $\frac{\# \text{Samples}}{\text{Half Frame}} - 1$.

Table 14-8: Number of BCLK per Half Frame and Sample Number

| | # BCLK Half Frame | # Samples Half Frame | FIFO Depth | <i>I2Sn_CTRL1CHO</i> | | | Sign extend when <i>I2Sn_CTRLOCHO.align=1</i> & <i>smp_size < FIFO Depth</i> |
|------------|----------------------|-------------------------|---------------|-----------------------|-----------------|-------------|--|
| | | | | <i>bits_word</i> | <i>smp_size</i> | <i>wsiz</i> | |
| 8-bit / 16 | 8 | 8 | 8 | 7 | 7 | 0 | |
| 8-bit / 32 | 16 | 8 | 8 | 15 | 7 | 0 | |
| 8-bit / 40 | 20 | 8 | 8 | 19 | 7 | 0 | |
| 8-bit / 48 | 24 | 8 | 8 | 23 | 7 | 0 | |

| | # BCLK Half Frame | # Samples Half Frame | FIFO Depth | <i>I2Sn_CTRL1CHO</i> | | | Sign extend when <i>I2Sn_CTRL0CHO.align=1</i> & <i>smp_size</i> < FIFO Depth |
|-------------|----------------------|-------------------------|---------------|----------------------|-----------------|--------------|--|
| | | | | <i>bits_word</i> | <i>smp_size</i> | <i>wsize</i> | |
| 8-bit / 64 | 32 | 8 | 8 | 31 | 7 | 0 | |
| 16-bit / 32 | 16 | 16 | 16 | 15 | 15 | 1 | |
| 16-bit / 40 | 20 | 16 | 16 | 19 | 15 | 1 | |
| 16-bit / 48 | 24 | 16 | 16 | 23 | 15 | 1 | |
| 16-bit / 64 | 32 | 16 | 16 | 31 | 15 | 1 | |
| 20-bit / 16 | 8 | 8 | 8 | 7 | 7 | 0 | |
| 20-bit / 32 | 16 | 16 | 16 | 15 | 15 | 1 | |
| 20-bit / 40 | 20 | 20 | 32 | 19 | 19 | 2 | sign |
| 20-bit / 48 | 24 | 20 | 32 | 23 | 19 | 2 | sign |
| 20-bit / 64 | 32 | 20 | 32 | 31 | 19 | 2 | sign |
| 24-bit / 16 | 8 | 8 | 8 | 7 | 7 | 0 | |
| 24-bit / 32 | 16 | 16 | 16 | 15 | 15 | 1 | |
| 24-bit / 40 | 20 | 20 | 32 | 19 | 19 | 2 | sign |
| 24-bit / 48 | 24 | 24 | 32 | 23 | 23 | 2 | sign |
| 24-bit / 64 | 32 | 24 | 32 | 31 | 23 | 2 | sign |
| 32-bit / 16 | 8 | 8 | 8 | 7 | 7 | 0 | |
| 32-bit / 32 | 16 | 16 | 16 | 15 | 15 | 1 | |
| 32-bit / 40 | 20 | 20 | 32 | 19 | 19 | 2 | sign |
| 32-bit / 48 | 24 | 24 | 32 | 23 | 23 | 2 | sign |
| 32-bit / 64 | 32 | 32 | 32 | 31 | 31 | 2 | |

14.10 Stereo/Mono Configuration

The I²S can transfer stereo or monophonic data based on the *I2Sn_CTRL0CHO.stereo* field. In stereo mode, both left and right channels contain data, which is the default setting. In mono mode, only the left or right channel contains data and the channel is selected using the *I2Sn_CTRL0CHO.stereo* field settings of 2 or 3.

14.11 FIFOs

Data to be transferred must be written to the TX FIFO by writing to the *I2Sn_FIFO0CHO.data* register (either directly or using a DMA channel). This data will be transmitted out by the hardware automatically a character (word/half word/byte) at a time, in the order that it was received. The status flags and associated I²S interrupts can be used to monitor the FIFO status and determine when the transfer cycle or cycles have completed.

Received data is loaded into the RX FIFO, and it can then be unloaded by reading from the *I2Sn_FIFO0CHO.data* register. An overrun event occurs if the RX FIFO is full and another character is received.

In the case of TX FIFO empty, FIFO read point will increase when the read the last non-empty FIFO data and point to the first nonvalid data and keep sending that data till new data is written. For the exception case of FIFO over run or under run, the channel should be reset and reconfigured.

Data is written into the TX FIFO to start the I²S transmission. The I²S data is continuously sent after the TX is enabled and the I²S clock divider is ready. The I²S will send previous (duplicate) data when the FIFO is empty. The TX FIFO operates in dual clock domains and supports asynchronous data movements from the write port to the read port. Synchronization is done by converting the pointer values from binary to gray code, and double registers are used for clock domain crossing.

Before data transmit, the format transfer block will fetch data from TX FIFO based on the word size and write to I²S data buffer based on the format definition (LSB first, MSB location, align, stereo, etc.). The I²S block will serially shift out the data.

14.12 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in

[Table 14-9: I²S Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 14-9: I²S Register Summary

| Offset | Register Name | Description |
|----------|-------------------------------|---------------------------------|
| [0x0000] | I2Sn_CTRL0CHO | Global Mode Control 0 Channel 0 |
| [0x0010] | I2Sn_CTRL1CHO | Local Setup Channel 0 Register |
| [0x0030] | I2Sn_DMACHO | DMA Control Channel 0 |
| [0x0040] | I2Sn_FIFOCHO | FIFO Channel 0 Register |
| [0x0050] | I2Sn_INTFL | Interrupt Status Register |
| [0x0054] | I2Sn_INTEN | Interrupt Enable Register |
| [0x0058] | I2Sn_EXTSETUP | Control Register |

14.12.1 Register Details

Table 14-10: Global Mode Control Channel 0

| Global Mode Control 0 Channel | | | I2Sn_CTRL0CHO | | 0x0000 |
|-------------------------------|------------|--------|-------------------------------|---|--------|
| Bits | Name | Access | Reset | Description | |
| 31:24 | rx_thd_val | R/W | 0 | RX FIFO Interrupt Threshold Event This field specifies the depth of receive FIFO for threshold interrupt generation. Values equal to 0 or greater than the FIFO depth are ignored. | |
| 23:21 | - | RO | 0 | Reserved Do not modify this field. | |
| 20 | fifo_lsb | R/W | 0 | Bit Field Control Only used if the FIFO size is larger than the sample size and I2Sn_CTRL0CHO.align = 0. For transmit, the LSB part is sent from the FIFO. For receive, store the LSB part in the FIFO without sign extension. 0: Disabled 1: Enabled | |
| 19 | rst | W1C | 0 | Channel Reset 0: No action 1: Reset channel | |
| 18 | flush | W1C | 0 | FIFO Buffer Flush Write 1 to initiate a flush of the RX and TX FIFOs. Hardware clears this bit to 0 when the operation is complete. 0: No operation. 1: Flush RX FIFO and TX FIFO | |

| Global Mode Control 0 Channel | | | I2Sn_CTRL0CHO | | 0x0000 |
|-------------------------------|-----------|--------|---------------|--|--------|
| Bits | Name | Access | Reset | Description | |
| 17 | rx_en | R/W | 0 | RX Enable 0: Disabled 1: Enabled | |
| 16 | tx_en | R/W | 0 | TX Enable 0: Disabled 1: Enabled | |
| 15:14 | wsize | R/W | 0x3 | Data Size When Writing to FIFO 0: Byte 1: Half-word (16 bits) 2: Word (32 bits) 3: Word (32 bits) | |
| 13:12 | stereo | R/W | 0 | I²S Mode 0: Stereo 1: Stereo 2: Monophonic left channel 3: Monophonic right channel | |
| 11 | ext_sel | R/W | 0 | External Clock Select 0: External source 0 (BCLK0/ws0) 1: External source 1 (BCLK1/ws1) | |
| 10 | align | R/W | 0 | Data Alignment 0: MSB 1: LSB | |
| 9 | msb_loc | R/W | 0 | MSB Position Within Field 0: Second bit of the word 1: First bit of the word | |
| 8 | ws_pol | R/W | 0 | LRCLK Polarity Select This setting has no effect if LRCLK is generated from a paired channel. 0: LRCLK low for left channel 0: LRCLK high for left channel | |
| 7:6 | ch_mode | R/W | 0 | Channel Mode 0: Master mode, internal generation of BCLK/LRCLK 1: Reserved 2: Reserved 3: Slave mode, external generation of BCLK/LRCLK | |
| 5:2 | - | DNM | 0 | Reserved Do not modify this field. | |
| 1 | lsb_first | R/W | 0 | LSB transmit/received first in SDO/SDI 0: Disabled 1: Enabled | |
| 0 | - | RO | 0 | Reserved Do not modify | |

Table 14-11: Local Setup Channel 0 Register

| Local Setup Channel 0 | | | I2Sn_CTRL1CH0 | | 0x0010 |
|-----------------------|--------|--------|---------------|---|--------|
| Bits | Name | Access | Reset | Description | |
| 31:16 | clkdiv | R/W | 0 | I²S Frequency Divisor | |
| 15 | adjst | R/W | 0 | Data Justification 0: Sample MSB of the data. 1: Sample LSB of the data. | |

| Local Setup Channel 0 | | | I2Sn_CTRL1CH0 | | 0x0010 |
|-----------------------|-----------|--------|---------------|--|--------|
| Bits | Name | Access | Reset | Description | |
| 14 | - | RO | 0 | Reserved Do not modify | |
| 13:9 | smp_size | R/W | 0 | Sample Size This field is the Word length of the sample size per half frame – 1 The Word length is equal to <i>I2Sn_CTRL1CH0.bits_word</i> when <i>smp_size</i> = 0 or <i>smp_size</i> > <i>I2Sn_CTRL1CH0.bits_word</i> . | |
| 8 | en | R/W | 0 | I²S Clock Enable Should be enabled when <i>I2Sn_CTRL1CH0.msb_loc</i> = 1. 0: Disabled 1: Enabled | |
| 7:5 | - | RO | 0 | Reserved Do not modify | |
| 4:0 | bits_word | R/W | 0 | I²S Word Length Per Half Frame This field is defined as the I ² S data bits per half frame minus 1. <i>Example: If the bit number is 16 per half frame, bits_word is 15.</i> | |

Table 14-12: I²S DMA Channel 0 Register

| I ² S DMA Channel 0 | | | I2Sn_DMACH0 | | 0x0030 |
|--------------------------------|----------------|--------|-------------|---|--------|
| Bits | Name | Access | Reset | Description | |
| 31:24 | rx_lvl | RO | 0 | Receive FIFO Level This field is the number of data words in the RX FIFO. | |
| 23:16 | tx_lvl | RO | 0 | Transmit FIFO Level This field is the number of data words in the TX FIFO. | |
| 15 | dma_rx_en | RW | 0 | DMA RX Channel Enable 0: Disabled 1: Enabled | |
| 14:8 | dma_rx_thd_val | RW | 0 | DMA RX FIFO Event Threshold If the RX FIFO level is greater than this value, then the RX FIFO DMA interface will send a signal to the system DMA to notify that RX FIFO has characters to transfer to memory. | |
| 7 | dma_tx_en | RW | 0 | DMA TX Channel Enable 0: Disabled 1: Enabled | |
| 6:0 | dma_tx_thd_val | RO | 0 | DMA TX FIFO Event Threshold If the TX FIFO level is less than this value, then the TX FIFO DMA interface will send a signal to system DMA to notify that TX FIFO is ready to receive data from memory. | |

Table 14-13: I²S FIFO Channel 0 Register

| I ² S FIFO Channel 0 | | | I2Sn_FIFOCH0 | | 0x0040 |
|---------------------------------|------|--------|--------------|---|--------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | I²S FIFO Writing to this field loads the next character into the TX FIFO and increments the <i>I2Sn_DMACH0.tx_lvl</i> . Writes are ignored if the TX FIFO is full. Reads of this field returns the next character available from the RX FIFO and decrements the <i>I2Sn_DMACH0.rx_lvl</i> . The value 0x0000 0000 is returned if <i>I2Sn_DMACH0.rx_lvl</i> = 0. | |

 Table 14-14: I²S Interrupt Flag Register

| I ² S Interrupt Flag | | | I2Sn_INTFL | | 0x0050 |
|---------------------------------|------------|--------|------------|---|--------|
| Bits | Name | Access | Reset | Description | |
| 31:4 | - | DNM | 0 | Reserved, Do Not Modify Do not modify this field. | |
| 3 | tx_he_ch0 | W1C | 0 | TX FIFO Half-Empty Event Interrupt Flag Channel 0 If this field is set to 1 the event has occurred. Write 1 to clear. 0: No event 1: Event occurred | |
| 2 | tx_ob_ch0 | W1C | 0 | TX FIFO One Byte Remaining Event Interrupt Flag Channel 0 If this field is set to 1 the event has occurred. Write 1 to clear. 0: No event 1: Event occurred | |
| 1 | rx_thd_ch0 | W1C | 0 | RX Threshold Event Interrupt Flag Channel 0 If this field is set to 1 the event has occurred. Write 1 to clear. 0: No event 1: Event occurred | |
| 0 | rx_ov_ch0 | W1C | 0 | RX FIFO Overrun Event Interrupt Flag Channel 0 If this field is set to 1 the event has occurred. Write 1 to clear. 0: No event 1: Event occurred | |

 Table 14-15: I²S Interrupt Enable Register

| I ² S Interrupt Enable | | | I2Sn_INTEN | | 0x0054 |
|-----------------------------------|------------|--------|------------|---|--------|
| Bits | Name | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved, Do Not Modify Reserved for Future Use, do not modify this field. | |
| 3 | tx_he_ch0 | R/W | 0 | TX FIFO Half-Empty Event Interrupt Enable Channel 0 0: Disabled 1: Enabled | |
| 2 | tx_ob_ch0 | R/W | 0 | TX FIFO One Byte Remaining Event Interrupt Enable Channel 0 0: Disabled 1: Enabled | |
| 1 | rx_thd_ch0 | R/W | 0 | RX FIFO Threshold Event Interrupt Enable Channel 0 0: Disabled 1: Enabled | |

| I ² S Interrupt Enable | | | I2Sn_INTEN | | 0x0054 |
|-----------------------------------|-----------|--------|------------|--|--------|
| Bits | Name | Access | Reset | Description | |
| 0 | rx_ov_ch0 | R/W | 0 | RX FIFO Overrun Event Interrupt Enable Channel 0 0: Disabled 1: Enabled | |

Table 14-16: I²S External Setup Register

| I ² S External Setup | | | I2Sn_EXTSETUP | | 0x0058 |
|---------------------------------|---------------|--------|---------------|--|--------|
| Bits | Name | Access | Reset | Description | |
| 31:5 | - | RO | 0 | Reserved Do not modify | |
| 4:0 | ext_bits_word | R/W | 0 | Extended Word Length If <i>I2Sn_CTRL0CH0.ch_mode</i> = 2, use this field to set the word length. | |

15. Parallel Camera Interface (PCIF)

The Parallel Camera Interface (PCIF) is a peripheral designed to read data from camera sensors.

Key features:

- Reads 8-bit, 10-bit, or 12-bit parallel data from an external camera sensor
- Supports multiple synchronization timing modes
 - ◆ Horizontal and Vertical Synchronization Timing Mode using the PCIF_HSYNC and PCIF_VSYNC pins
 - ◆ Start Active Video (SAV) and End Active Video (EAV) embedded timing codes within the data stream
- 8×32-bit word FIFO depth
- Interrupt support for
 - ◆ FIFO not empty
 - ◆ FIFO threshold
 - ◆ FIFO full
 - ◆ Image complete
- Supports either single image capture mode or continuous image capture mode

15.1 Instances

There is one instance of the Parallel Camera Interface, shown in [Table 15-1](#). The pins and alternate functions for the PCIF are shown in [Table 15-2: MAX78000 PCIF Signals](#).

Table 15-1: MAX78000 PCIF Instances

| Instance | PCIF Peripheral Clock Clock Options | RX FIFO Depth |
|----------|--|------------------|
| PCIF | PCLK | 8 |

Table 15-2: MAX78000 PCIF Signals

| Signal Name | 81-CTBGA Pin | Alternate Function | Signal Direction | Description |
|-------------|-----------------|-----------------------|---------------------|----------------------------------|
| PCIF_PCLK | P1.9 | AF1 | Input | Pixel Clock Input |
| PCIF_HSYNC | P1.8 | AF1 | Input | Horizontal Synchronization Input |
| PCIF_VSYNC | P0.15 | AF2 | Input | Vertical Synchronization Input |
| PCIF_D0 | P0.20 | AF2 | Input | Pixel Data Input 0 |
| PCIF_D1 | P0.21 | AF2 | Input | Pixel Data Input 1 |
| PCIF_D2 | P0.22 | AF2 | Input | Pixel Data Input 2 |
| PCIF_D3 | P0.23 | AF2 | Input | Pixel Data Input 3 |
| PCIF_D4 | P0.24 | AF2 | Input | Pixel Data Input 4 |
| PCIF_D5 | P0.25 | AF2 | Input | Pixel Data Input 5 |
| PCIF_D6 | P0.26 | AF2 | Input | Pixel Data Input 6 |
| PCIF_D7 | P0.27 | AF2 | Input | Pixel Data Input 7 |
| PCIF_D8 | P0.30 | AF2 | Input | Pixel Data Input 8 |
| PCIF_D9 | P0.31 | AF2 | Input | Pixel Data Input 9 |
| PCIF_D10 | P1.6 | AF2 | Input | Pixel Data Input 10 |
| PCIF_D11 | P1.7 | AF2 | Input | Pixel Data Input 11 |

15.2 Capture Modes

The PCIF supports either single image capture mode or continuous capture mode. Each mode and the PCIF configuration is described in the following sections.

15.2.1 Single Image Capture

In this mode the PCIF waits for one image from the sensor, then stops reading data. Configure the PCIF for this mode by setting the `PCIF_CFG.read_mode` field to 1. The `PCIF_CFG.read_mode` field remains set prior to and while receiving image data from the camera. Once the image is complete, the hardware automatically sets the `PCIF_CFG.read_mode` field to 0 and sets the `PCIF_STFLG.img_done` status to 1.

15.2.2 Continuous Capture

In this mode the PCIF continues to read image data as long as the connected camera sensor continues to provide image data. Configure the PCIF for continuous capture mode by setting the `PCIF_CFG.read_mode` field to 2. Disable continuous mode capture by setting the `PCIF_CFG.read_mode` field to 0.

15.3 Timing Modes

There are two different timing modes. Horizontal and Vertical Synchronization Mode and Data Streaming Mode. Both timing modes can be combined with single image capture or continuous capture read modes.

15.3.1 Horizontal and Vertical Synchronization Timing Mode

In this timing mode the PCIF uses the PCIF_HSYNC and the PCIF_VSYNC input pins to determine the beginning and end of image data. The PCIF begins to accept image data on the PCIF_Dx pins once the PCIF_VSYNC input pin is transitioned from 0 to 1 and the PCIF_HSYNC input pin reads 1. The PCIF_VSYNC pin only needs to remain high for one PCIF_PCLK period for detection of the start of video signal. The PCIF_HSYNC signal is used to frame a complete set of pixel data. Re-assertion of the PCIF_VSYNC signal indicates to the PCIF that the image is complete.

Set the bit `PCIF_CFG.timing_sel` to 0 to configure the PCIF for Horizontal and Vertical Synchronization mode.

15.3.2 Data Stream Timing Mode

In this timing mode the PCIF_HSYNC and PCIF_VSYNC input pins are ignored. The PCIF uses embedded timing codes to determine the start and end of a single image or continuous stream. These codes can be configured by setting the Start Active Video (SAV) code (`PCIF_TMG_CODES.sav`) and the End Active Video (EAV) code (`PCIF_TMG_CODES.eav`). These two codes must match the codes sent by the connected camera respectively and cannot be identical. Set `PCIF_CFG.timing_sel` to 1 to configure the PCIF for embedded timing codes mode.

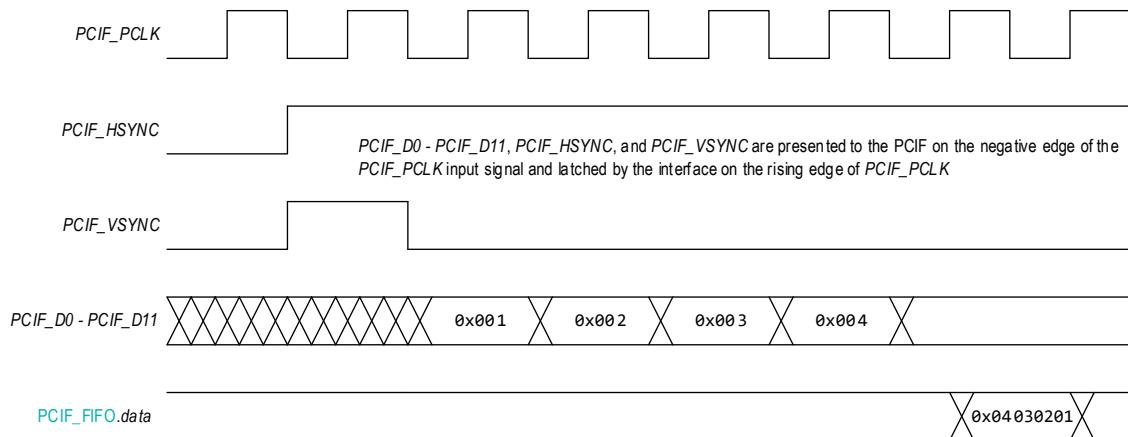
15.4 Data Width

The width of the pixel data can be configured as 8-bit, 10-bit or 12-bit. Pixel data is read from the PCIF_Dx input pins on the rising edge of the PCIF_PCLK input pixel clock. It is assumed that PCIF_Dx will change on the negative edge of PCIF_PCLK.

15.4.1 8-bit Width

Setting `PCIF_CFG.data_width` to 0 sets the recognized pixel width on the PCIF_Dx bus to 8 bits. The upper 4 bits of PCIF_Dx inputs are ignored. Pixel data will be framed as 32-bit words before these words are transferred to the 32-bit wide Data FIFO `PCIF_FIFO` and made ready to be read. The 32-bit Data FIFO word is oriented as having the most significant byte the most recently received 8-bit PCIF_Dx data. See [Figure 15-1: Horizontal and Vertical Synchronization Timing Mode with 8-Bit Data Width](#) and [Figure 15-2: Data Stream Timing Mode with 8-Bit Data Width](#) examples.

Figure 15-1: Horizontal and Vertical Synchronization Timing Mode with 8-Bit Data Width



PCIF_FIFO is the internal FIFO register.

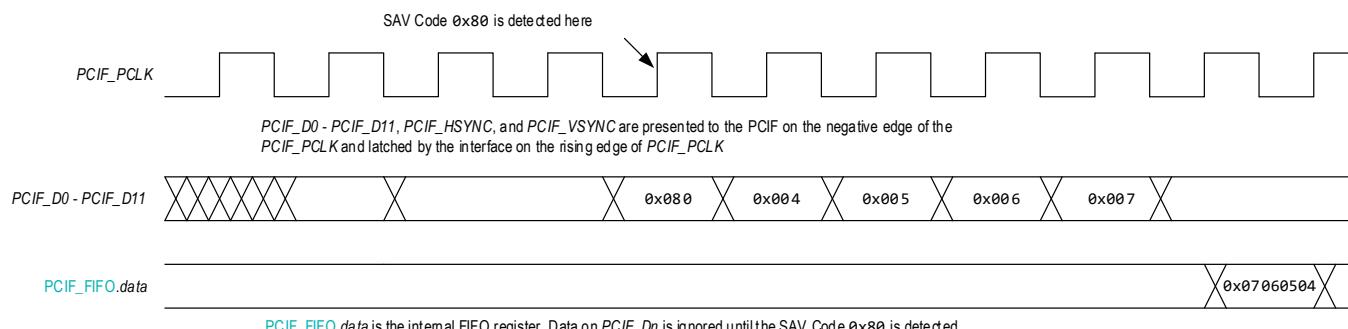
PCIF Data Input pins, PCIF_D8 – PCIF_D11, are ignored in 8-bit Data Width Mode

PCIF Register Configuration

PCIF_CFG.data_width = 0b00

PCIF_CFG.timing_sel = 0b0

Figure 15-2: Data Stream Timing Mode with 8-Bit Data Width



PCIF_FIFO is the internal FIFO register. Data on PCIF_Dn is ignored until the SAV Code 0x80 is detected.

PCIF Register Configuration

PCIF_TMG_CODES.sav = 0x80

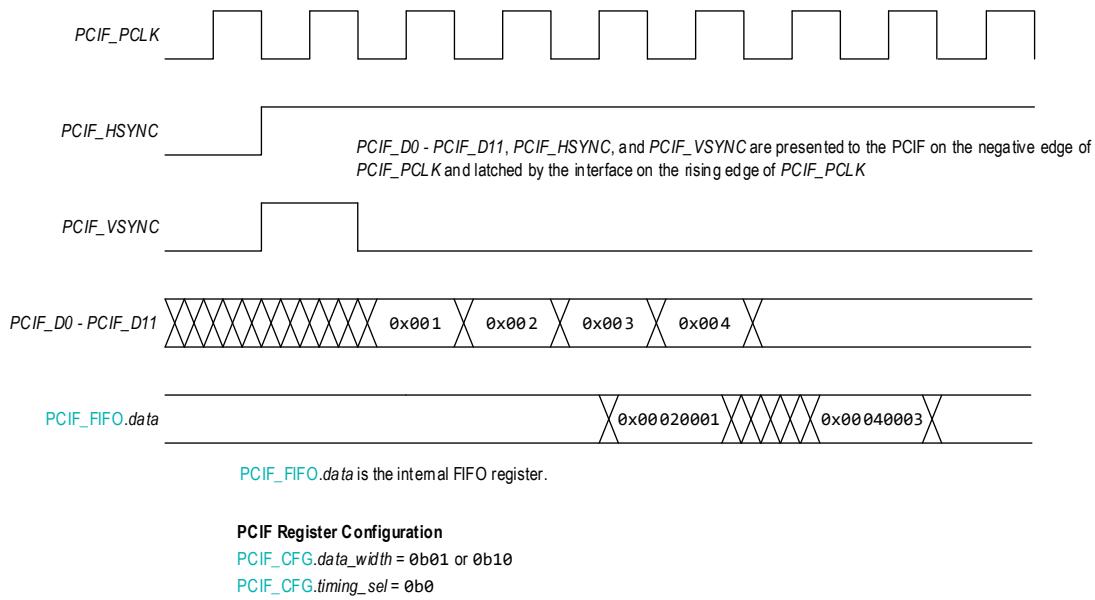
PCIF_CFG.data_width = 0b00

PCIF_CFG.timing_sel = 0b1

15.4.2 10 and 12-bit Width

Setting **PCIF_CFG.data_width** to 1 sets the recognized pixel width on the PCIF_Dx bus to 10-bits. Setting **PCIF_CFG.data_width** to 2 sets the recognized pixel width on the PCIF_Dx bus to 12-bits. As with the 8-bit width setting, the pixel data will be framed as 32-bit words before these words are transferred to the 32-bit wide Data FIFO **PCIF_FIFO** and made ready to be read. These pixel widths are MSB zero padded to 16-bits and then two 16-bit pixels are concatenated to form the 32-bit word, the most recently received PCIF_Dx data is the most significant 16-bits of the FIFO data. See [Figure 15-3](#) for PCIF_VSYNC/PCIF_HSYNC Timing example.

Figure 15-3: 10 or 12-bit PCIF_VSYNC/PCIF_HSYNC



15.5 Data FIFO

The Data FIFO **PCIF_FIFO** is a 32-bit wide 8-word deep buffer that contains data read from the PCIF_Dx pixel data input pins. The Data FIFO threshold can be configured by setting **PCIF_CFG fifo_thresh**. The **PCIF_STFLG fifo_thresh** will be set if the Data FIFO depth becomes greater than or equal to **PCIF_CFG fifo_thresh**. An interrupt can be generated when this condition happens if **PCIF_INT_EN fifo_thresh** is set. The Data FIFO also provides status flags for FIFO Full (**PCIF_STFLG fifo_full**) and FIFO Not Empty (**PCIF_STFLG fifo_not_empty**). Both status flags have associated interrupts (**PCIF_INT_EN fifo_full** and **PCIF_INT_EN fifo_not_empty**) that can be enabled and triggered when the status flags become set.

15.6 Usage

15.6.1 DMA

1. Set **PCIF_CFG.data_width** and **PCIF_CFG.timing_sel** as required by the camera sensor attached.
2. Enable the **PCIF_INT_EN.img_done** to generate an interrupt once the image is complete.
3. Set **PCIF_CFG.read_mode** for single image or continuous capture. Triggering the camera sensor to output an image will start the PCI automatically.
4. As data is read from the camera sensor by the PCIF it will trigger a read request whenever it has a full 32-bit word in the Data FIFO. Once the camera sensor has finished transmitting data, signaled by a rising edge on **PCIF_VSYNC** or a data stream EAV code. The PCIF will trigger the **PCIF_INT_EN.img_done** interrupt.
5. The interrupt handler can then reset the interrupt flag by writing 1 to **PCIF_STFLG.img_done**.

15.6.2 Interrupts

1. Set *PCIF_CFG.data_width* and *PCIF_CFG.timing_sel* as required by the camera sensor attached.
2. Set *PCIF_CFG fifo_thresh* to the desired level to allow the interrupt to service the FIFO before it fills.
3. Enable the *PCIF_INT_EN.img_done* and the *PCIF_INT_EN fifo_thresh* interrupts, generating an ISR when the image is complete or the FIFO has been filled to the threshold level set in the *PCIF_CFG fifo_thresh* field.
4. Set *PCIF_CFG.read_mode* for single image or continuous capture. When the camera sensor is triggered to output an image the PCIF will automatically start receiving data.
5. As data is read from the camera sensor by the PCIF, the hardware will trigger an ISQ whenever the FIFO threshold *PCIF_CFG fifo_thresh* has been met. The interrupt handler should perform a burst read from the FIFO (*PCIF_FIFO.data*). When the camera sensor finishes transmitting image data, signaled either by a rising edge on PCIF_VSYNC or a Data Stream EAV code, the hardware generates an *PCIF_INT_EN.img_done* interrupt.
6. After servicing an image done interrupt, the interrupt handler must reset the image done interrupt flag by writing 1 to the *PCIF_STFLG.img_done*.
7. Software should check *PCIF_STFLG fifo_not_empty* and perform a read of *PCIF_FIFO.data* to receive the remainder of the words of data that occupy the FIFO less than *PCIF_CFG fifo_thresh*. When all of the data is read from the FIFO, hardware clears the *PCIF_STFLG fifo_not_empty* flag automatically.

15.7 Parallel Camera Registers

See *Table 2-4: APB Peripheral Base Address Map* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own independent set of the registers shown in *Table 15-3: Parallel Camera Interface Register Summary*. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See *Table 2-1: Field Access Definitions* for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 15-3: Parallel Camera Interface Register Summary

| Offset | Register | Name |
|----------|------------------------|--------------------------------|
| [0x0008] | <i>PCIF_CFG</i> | PCIF Configuration Register |
| [0x000C] | <i>PCIF_INT_EN</i> | PCIF Interrupt Enable Register |
| [0x0010] | <i>PCIF_STFLG</i> | PCIF Status Flag Register |
| [0x0014] | <i>PCIF_TIMG_CODES</i> | PCIF Timing Code Register |
| [0x0030] | <i>PCIF_FIFO</i> | PCIF FIFO Data Register |

15.7.1 Parallel Camera Register Details

Table 15-4: PCIF Configuration Register

| PCIF Configuration | | PCIF_CFG | | | [0x0008] |
|--------------------|-------|----------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:9 | - | RO | 0 | Reserved Do not modify this field. | |

| PCIF Configuration | | PCIF_CFG | | | [0x0008] |
|--------------------|-------------|----------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 8:5 | fifo_thresh | R/W | 0 | Data FIFO Threshold Setting If the number of words in the FIFO is greater than or equal to this value, then PCIF_STFLG fifo_thresh will be set. 0b0000: FIFO threshold equals 0 words 0b0001: FIFO threshold equals 1 word 0b1000: FIFO threshold equals 8 words 0b1001 - 0b1111: Reserved | |
| 4 | timing_sel | R/W | 0 | Camera Timing Select This field selects the camera timing synchronization to either HSYNC/VSYNC mode or embedded timing codes in the camera data. 0: VSYNC/HSYNC timing-controlled images 1: Embedded timing codes via start of video (SAV) and end of video (EAV) codes. | |
| 3:2 | data_width | R/W | 0 | Camera Data Width Set this field to the width of the camera's data. 0b00: 8-bit data 0b01: 10-bit data 0b10: 12-bit data 0b11: Reserved <i>Note: Unused PCIF_Dx pins are ignored.</i> | |
| 1:0 | read_mode | R/W | 0 | Camera Read Mode Set this field to the required camera read mode or set to 0 to disable the PCIF. 0b00: PCIF disabled 0b01: Single Image Capture 0b10: Continuous Capture 0b11: Reserved | |

Table 15-5: PCIF Interrupt Enable Register

| PCIF Interrupt Enable | | PCIF_INT_EN | | | [0x000C] |
|-----------------------|----------------|-------------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved Do not modify this field. | |
| 3 | fifo_not_empty | R/W | 0 | FIFO Not Empty Interrupt Enable Set this field to 1 to generate an interrupt when the FIFO is not empty (PCIF_STFLG fifo_not_empty = 1), indicating data is available to read from the FIFO. 0: Interrupt Disabled 1: Interrupt Enabled | |
| 2 | fifo_thresh | R/W | 0 | FIFO Threshold Interrupt Enable Set this field to 1 to generate an interrupt when the FIFO threshold is reached (PCIF_STFLG fifo_thresh = 1). 0: Interrupt Disabled 1: Interrupt Enabled | |

| PCIF Interrupt Enable | | | PCIF_INT_EN | | [0x000C] |
|-----------------------|-----------|--------|-------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 1 | fifo_full | R/W | 0 | FIFO Full Interrupt Enable Set this bit to 1 to generate an interrupt when the FIFO is full (<i>PCIF_STFLG fifo_full</i> = 1). 0: Interrupt Disabled 1: Interrupt Enabled | |
| 0 | img_done | R/W | 0 | Image Complete Interrupt Enable Set this bit to 1 to generate an interrupt when the image is done (<i>PCIF_STFLG img_done</i> = 1). 0: Interrupt Disabled 1: Interrupt Enabled | |

Table 15-6: PCIF Status Flags Register

| PCIF Status Flags | | | PCIF_STFLG | | [0x0010] |
|-------------------|----------------|--------|------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved Do not modify this field. | |
| 3 | fifo_not_empty | RO | 0 | FIFO Not Empty Status Flag This status is set by hardware when the FIFO level is 1 or greater. This flag is automatically cleared by hardware when all data has been read from the FIFO. 0: The FIFO is empty 1: The FIFO is not empty | |
| 2 | fifo_thresh | RO | 0 | FIFO Threshold Status Flag This status is set by hardware when the FIFO level is greater than or equal to the <i>PCIF_CFG fifo_thresh</i> field. When the level in the FIFO falls below the set threshold, this will be automatically cleared by hardware. 0: FIFO threshold not exceeded 1: FIFO threshold exceeded | |
| 1 | fifo_full | RO | 0 | FIFO Full Status Flag This status is set by hardware when the FIFO has reached its full capacity of eight 32-bit words. The interrupt flag is cleared by hardware automatically when data is read from the FIFO. 0: The FIFO is not full 1: The FIFO is full | |
| 0 | img_done | R/W1C | 0 | Image Complete Status Flag This status is set by hardware when either the PCIF_VSYNC device pin has transitioned logic level during a triggered camera sensor read, or the EAV code, <i>PCIF_TMG_CODES.eav</i> , is detected. 0: End of image not detected 1: End of image detected | |

Table 15-7: PCIF Timing Codes Register

| PCIF Camera Timing Codes | | | PCIF_TMG_CODES | | [0x0014] |
|--------------------------|-------|--------|----------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved Do not modify this field. | |
| 15:8 | eav | R/W | 0x9D | End Active Video The end active video field is an 8-bit code that is camera-dependent. This value cannot be equal to PCIF_TMG_CODES.sav . Set this field to the camera's end active video code, which may differ from the reset default of 0x9D. | |
| 7:0 | sav | R/W | 0x80 | Start Active Video The start active video field is an 8-bit code that is camera-dependent. This value cannot be equal to PCIF_TMG_CODES.eav . Set this field to the camera's start active video field, which may differ from the reset default of 0x80. | |

Table 15-8: PCIF FIFO Data Register

| PCIF FIFO | | | PCIF_FIFO | | [0x0030] |
|-----------|-------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | R | 0 | Data Data from the FIFO to be read. Once read, the next value in the FIFO becomes immediately available to read. | |

16. 1-Wire Master (OWM)

The device provides a 1-Wire master (OWM) that you can use to communicate with one or more external 1-Wire slave devices using a single-signal, combined clock, data protocol. The OWM is contained in the OWM module. The OWM module handles the lower-level details (including timing and drive modes) required by the 1-Wire protocol, allowing the CPU to communicate over the 1-Wire bus at a logical data level.

16.1 1-Wire Master Features

The OWM provides the following features:

- Flexible, 1-Wire timing generation (required 1MHz timing base) using the OWM module clock frequency, which is in turn derived from the current system clock source. You can also pre-scale the OWM module clock to allow proper 1-Wire timing generation using a range of base frequencies.
- Automatic generation of proper 1-Wire time slots for both standard and overdrive timing modes.
- Flexible configuration for 1-Wire line pullup modes: options for internal pullup, external fixed pullup, and optional external strong pullup are available.
- Long-line compensation and bit banging (direct firmware drive) modes.
- 1-Wire reset generation and presence-pulse detection.
- Generation of 1-Wire read and write time slots for single-bit and eight-bit byte transmissions.
- Search ROM Accelerator (SRA) mode, which simplifies the generation of multiple-bit time slots and discrepancy resolution required when completing the Search ROM function to determine the IDs of multiple, unknown 1-Wire slaves on the bus.
- Transmit data completion, received data available, presence pulse detection, and 1-Wire line-error condition interrupts.

For more information about the Maxim 1-Wire protocol and supporting devices, refer to the following resources:

- *AN937: The Book of iButton Standards*
 - ◆ www.maximintegrated.com/AN937
- *AN1796: Overview of 1-Wire Technology and its Use*
 - ◆ www.maximintegrated.com/AN1796
- *AN187: 1-Wire Search Algorithm*
 - ◆ www.maximintegrated.com/AN187

iButton is a registered trademark of Maxim Integrated Products, Inc.

16.2 1-Wire Pins and Configuration

The one instance of the peripheral is shown in *Table 16-1: MAX78000 1-Wire Master Peripheral Pins* lists the locations of the OWM_IO and OWM_PE signals for each of the OWMn peripherals per package.

Table 16-1: MAX78000 1-Wire Master Peripheral Pins

| OWMn | ALTERNATE FUNCTION | ALT FUNCTION # | 81-CTBGA |
|------|--------------------|----------------|----------|
| OWM0 | OWM_IO | AF1 | P0.6 |
| | OWM_PE | AF1 | P0.7 |

16.2.1 1-Wire I/O (OWM_IO)

The OWM_IO pin is a bidirectional I/O that is used to directly drive the external 1-Wire bus. As described in the *Book of iButton Standards*, this I/O is generally driven as an open-drain output. The 1-Wire bus requires a common pullup to return the 1-Wire bus line to an idle high state when no master or slave device is actively driving the line low. This pullup can consist of a fixed resistor pullup (connected to the 1-Wire bus outside the microcontroller), an internal pullup enabled by setting *OWM_CFG.int_pullup_enable* to 1, or an OWM module controlled external pullup enabled by setting *OWM_CFG.ext_pullup_mode* to 1.

16.2.2 Pullup Enable (OWM_PE)

The 1-Wire pullup enable (PE) signal is an active high output used to enable an optional external pullup on the 1-Wire bus. This pullup is intended to provide a stronger (lower impedance) pullup on the 1-Wire bus under certain circumstances, such as during overdrive mode.

16.2.3 Clock Configuration

To correctly generate the timing required by the 1-Wire protocol in Standard or Overdrive timing modes, the OWM clock must be set to achieve $f_{owmclk} = 1\text{MHz}$. This clock generates both the Standard and Overdrive timing, so it does not need adjustment when transitioning from Standard to Overdrive mode or vice versa.

The OWM peripheral uses the system peripheral clock, PCLK, divided by the value in the *OWM_CLK_DIV_1US.divisor* field as shown in *Equation 16-1, below*, where $f_{PCLK} = f_{SYSCLK}/2$:

Equation 16-1: OWM 1MHz Clock Frequency

$$f_{owmclk} = 1\text{MHz} = \frac{f_{PCLK}}{\text{OWM_CLK_DIV_1US}.divisor}$$

16.3 1-Wire Protocol

The general timing and communication protocols used by the OWM interface are those standardized for the 1-Wire network.

Because the 1-Wire interface is a master interface, it initiates and times all communication on the 1-Wire bus. Except for the present pulse generation when a device first connects to the 1-Wire bus, 1-Wire slave devices complete 1-Wire bus communication only as directed by the 1-Wire bus master. From a firmware perspective, the lowest-level timing and electrical details of how the 1-Wire network operates are unimportant. The application can configure the OWM module properly and direct it to complete low-level operations such as reset, read, and write bit/byte operations. Thus, the OWM module on the microcontroller is designed to interface to the 1-Wire bus at a low level.

16.3.1 Networking Layers

In the *Book of iButton Standards*, the 1-Wire communication protocol is described in terms of the ISO-OSI model (International Organization of Standardization (ISO) Open System Interconnection (OSI) network layer model). Network

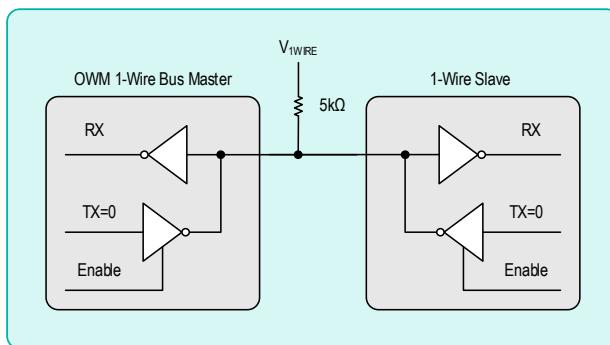
layers that apply to this description are the Physical, Link, Network, and Transport layers. The Transport layer consists of the software that transfers memory data other than ROM ID contents to and from the individual 1-Wire network nodes. The Presentation layer would correspond to higher-level application software functions (such as library layers) that implement communication protocols using the 1-Wire layers as a foundation. This document describes the details of the Physical, Link and Network layers with regards to the OSI network layer model. The Transport and Presentation layers are beyond the scope of this document.

16.3.1.1 Physical Layer

The 1-Wire communication bus consists of a single data/power line plus ground. devices (either master or slave) that interface to the 1-Wire communication bus using an open-drain (active low) connection, which means that the 1-Wire bus normally idles in a high state.

An external pullup resistor is used to pull the 1-Wire line high when no master or slave device is driving the line. This means that 1-Wire devices do not actively drive the 1-Wire line high. Instead, they either drive the line low or release it (set their output to high impedance) to allow the external resistor to pull the line high. This allows the 1-Wire bus to operate in a wired-AND manner as shown in [Figure 16-1](#) and avoids bus contention if more than one device attempts to drive the 1-Wire bus at the same time.

Figure 16-1: 1-Wire Signal Interface



16.3.1.2 Link Layer

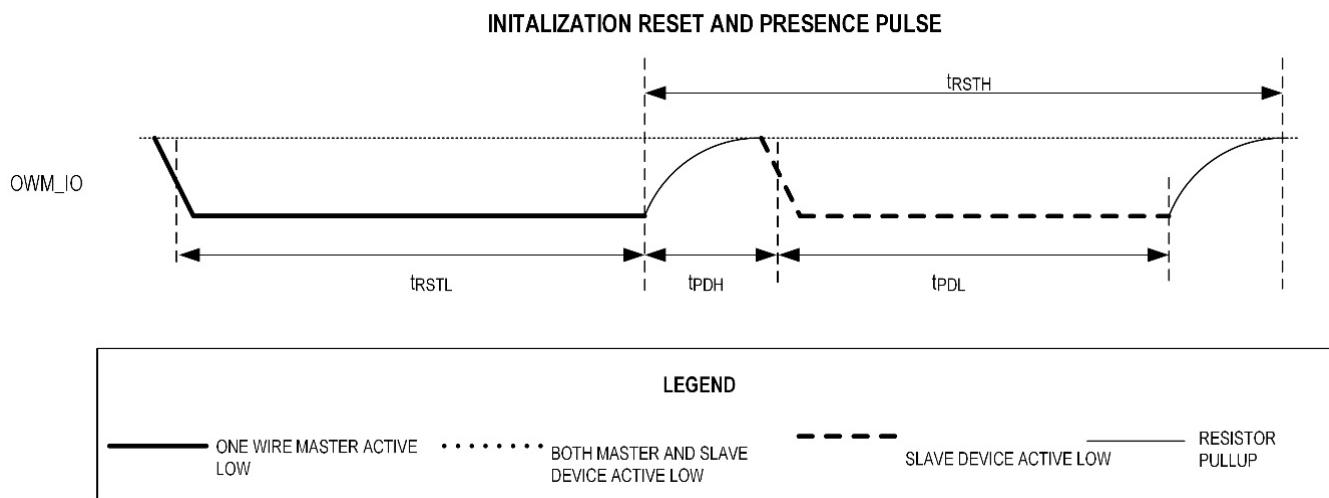
The 1-Wire Bus supports a single master and one or more slave devices (multidrop). Slave devices can connect to and disconnect from the 1-Wire Bus dynamically (as is typically the case with iButton devices that operate using an intermittent touch contact interface), which means that it is the master's responsibility to poll the bus as needed to determine the number and types of 1-Wire devices that are connected to the bus.

All communication sequences on the 1-Wire Bus are initiated by the OWM. The OWM determines when 1-Wire data transmissions begin, as well as the overall communication speed that is used. There are three different communication speeds supported by the 1-Wire specification: standard speed, overdrive speed, and hyperdrive speed. However, only standard speed and overdrive speed are supported by the OWM peripheral in the devices.

16.3.1.2.1 OWM Reset and Presence Detect

The OWM begins each communication sequence by sending a reset pulse as shown in [Figure 16-2](#). This pulse resets all 1-Wire slave devices on the line to their initial states and causes them all to begin monitoring the line for a command from the OWM. Each 1-Wire slave device on the line responds to the reset pulse by sending out a presence pulse. These pulses from multiple 1-Wire slave devices are combined in wired-AND fashion, resulting in a pulse whose length is determined by the slowest 1-Wire slave device on the bus.

Figure 16-2: 1-Wire Reset Pulse



In general, the 1-Wire line must idle in a high state when communication is not taking place. It is possible for the master to pause communication in between time slots. There is not an overall "timeout" period that causes a slave to revert to the reset state if the master takes too long between one time slot and the next time slot.

The 1-Wire communication protocol relies on the fact that the maximum allowable length for a bit transfer (write 0/1 or read bit) time slot is less than the minimum length for a 1-Wire reset. At any time, if the 1-Wire line is held low (by the master or by any slave device) for more than the minimum reset pulse time, all slave devices on the line interpret this as a 1-Wire reset pulse.

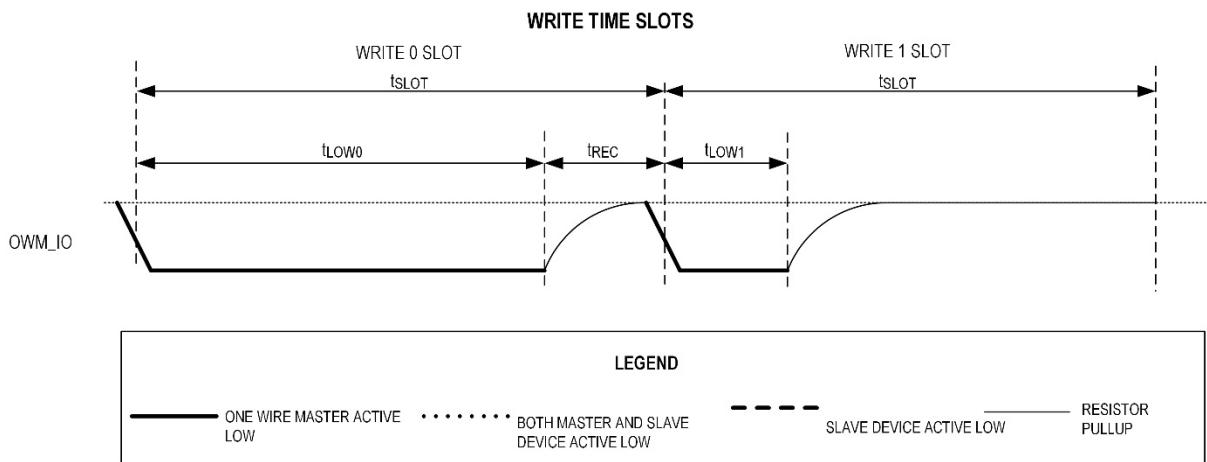
16.3.1.2.2 OWM Write Time Slot

All 1-Wire bit time slots are initiated by the 1-Wire bus master and begin with a single falling edge. There is no indication given by the beginning of a time slot whether a read bit or write bit operation is intended, as the time slots all begin in the same manner. Rather, the 1-Wire command protocol enforces agreement between the OWM and slave as to which time slots are used for bit writes and which time slots are used for bit reads.

When multiple bits of a value are transmitted (or read) in sequence, the least significant bit of the value is always sent or received first. The 1-Wire bus is a half-duplex bus, so data is transmitted in only one direction (from master to slave or from slave to master) at any given time.

As shown in *Figure 16-3*, the time slots for writing a 0 bit and writing a 1 bit begin identically, with the falling edge and a minimum-width low pulse sent by the master. To write a one bit, the master releases the line after the minimum low pulse, allowing it to be pulled high. To write a zero bit, the master continues to hold the line low until the end of the time slot.

Figure 16-3: 1-Wire Write Time Slot



From the slave's perspective, the initial falling edge of the time slot triggers the start of an internal timer, and when the proper amount of time has passed, the slave samples the 1-Wire line that is driven by the master. This sampling point is in between the end of the minimum-width low pulse and the end of the time slot.

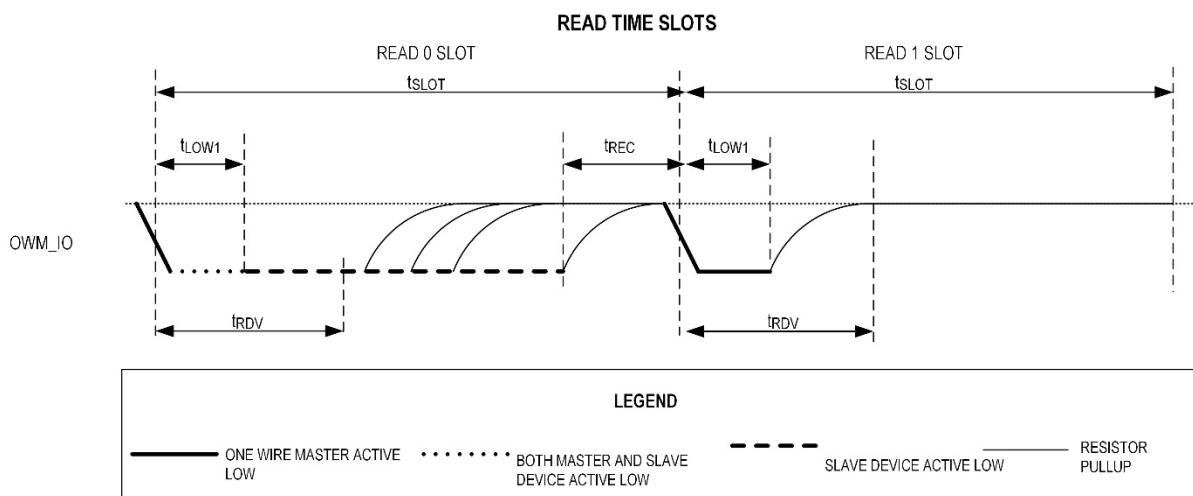
16.3.1.2.3 OWM Read Time Slot

As with all 1-Wire transactions, the master initiates all bit read time slots. Like the bit write time slots, the bit read time slot begins with a falling edge. From the master's perspective, this time slot is transmitted identically to the "Write 1 Bit" time slot shown in *Figure 16-3*. The master begins by transmitting a falling edge, holds the line low for a minimum-width period, and then releases the line.

The difference here is that instead of the slave sampling the line, the slave begins transmitting either a 0 (by holding the line low) or a 1 (by leaving the line to float high) after the initial falling edge. The master then samples the line to read the bit value that is transmitted by the slave device.

As an example, *Figure 16-4* shows a sequence in which the slave device transmits data back to the 1-Wire bus master upon request. Note that to transmit a 1 bit, the slave device does not need to do anything. It simply leaves the line alone (to float high) and waits for the next time slot. To transmit a 0 bit, the slave device holds the line low until the end of the time slot.

Figure 16-4: 1-Wire Read Time Slot



16.3.1.2.4 Standard Speed and Overdrive Speed

By default, all 1-Wire communications following reset begin at the lowest rate of speed (that is, standard speed). For 1-Wire devices that support it, it is possible for the OWM to increase the rate of communication from standard speed to overdrive speed by sending the appropriate command.

The protocols and time slots operate identically for standard and overdrive speeds. The difference comes in the widths of the time slots and pulses. The OWM automatically adjusts the timings based on the setting of the *OWM_CFG.overdrive* field.

If a 1-Wire slave device receives a standard speed reset pulse, it resets and reverts to standard speed communication. If the device is already communicating in overdrive mode, and it receives a reset pulse at the overdrive speed, it resets but remains in overdrive mode.

16.3.1.3 Network Layer

16.3.1.3.1 ROM Commands

Following the initial 1-Wire reset pulse on the bus, all slave 1-Wire devices are active, which means that they are monitoring the bus for commands. Because the 1-Wire bus can have multiple slave devices present on the bus at any time, the OWM must go through a process (defined by the 1-Wire command protocol) to activate only the 1-Wire slave device that it intends to communicate with and deactivate all others. This is the purpose of the ROM commands (network layer) shown in *Table 16-2*.

Table 16-2: 1-Wire ROM Commands

| ROM Command | Hex Value |
|----------------------|-----------|
| Read ROM | 0x33 |
| Match ROM | 0x55 |
| Search ROM | 0xF0 |
| Skip ROM | 0xCC |
| Overdrive Skip ROM | 0x3C |
| Overdrive Match ROM | 0x69 |
| Resume Communication | 0xA5 |

The ROM command layer relies on the fact that all 1-Wire slave devices are assigned a globally unique, 64-bit ROM ID. This ROM ID value is factory programmed to ensure that no two 1-Wire slave devices have the same value.

16.3.1.3.2 ROM ID

Figure 16-5 is a visual representation of the 1-Wire ROM ID fields and shows the organization of the fields within the 64-bit ROM ID for a device.

Figure 16-5: 1-Wire ROM ID Fields

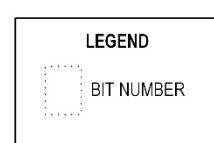
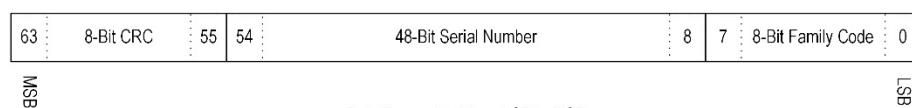


Table 16-3 provides a detailed description of each of the ROM ID fields.

Table 16-3: 1-Wire Slave Device ROM ID Field

| Field | Bit Number | Description |
|-------------|------------|--|
| Family code | 0-7 | This 8-bit value is used to identify the type of a 1-Wire slave device. |
| Unique ID | 8-55 | This 48-bit value is factory-programmed to give each 1-Wire slave device (within a given family code group) a globally unique identifier. |
| CRC | 56-63 | This is the 8-bit, 1-Wire CRC as defined in the <i>Book of iButton Standards</i> . The CRC is generated using the polynomial $(x^8 + x^5 + x^4 + 1)$. |

Note: For certain operations that consist only of writing from the OWM to the slave, it is technically possible for the master to communicate with more than one slave at a time on the same 1-Wire bus. For this to work, the exact same data must be transmitted to all slave devices, and any values read back from the slaves must either be identical as well or must be disregarded by the master device (because different slaves can attempt to transmit different values). The descriptions below assume, however, that the master is communicating with only one slave device at a time because this is the method that is normally used.

As explained above, the ROM ID contents play a key role in addressing and selecting devices on the 1-Wire bus. All devices except one are in an idle/inactive state after the Match ROM command or the Search ROM command is executed. They return to the active state only after receiving a 1-Wire reset pulse.

Devices with overdrive capability are distinguished from others by their family code and two additional ROM commands: Overdrive Skip ROM and Overdrive Match ROM. The first transmission of the ROM command itself takes place at the normal speed that is understood by all 1-Wire devices. After a device with overdrive capability is addressed and set into overdrive mode (that is, after the appropriate ROM command is received), further communication to that device must occur at overdrive speed. Because all deselected devices remain in the idle state if no reset pulse of regular duration is detected, even multiple overdrive components can reside on the same 1-Wire bus. A reset pulse of regular duration resets all 1-Wire devices on the bus and simultaneously sets all overdrive-capable devices back to the default standard speed.

16.3.2 Read ROM Command

The Read ROM command allows the OWM to obtain the 8-byte ROM ID of any slave device connected to the 1-Wire bus. Each slave device on the bus responds to this command by transmitting all eight bytes of its ROM ID value starting with the least significant byte (Family Code) and ending with the most significant byte (CRC).

Because this command is addressed to all 1-Wire devices on the bus, if more than one slave is present on the bus there is a data collision as multiple slaves attempt to transmit their ROM IDs at once. This condition is detectable by the OWM because the CRC value does not match the ROM ID value received. In this case, the OWM should reset the 1-Wire bus and select a single slave device on the bus to continue either by using the Match ROM command (if the ROM ID values are already known) or the Search ROM command (if the master has not yet identified some or all devices on the bus).

After the Read ROM command is complete, all slave devices on the 1-Wire bus are selected or active, and communication proceeds to the Transport layer.

16.3.3 Skip ROM and Overdrive Skip ROM Commands

The Skip ROM command is used to activate all slave devices present on the 1-Wire bus regardless of their ROM ID. Normally, this command is used when only a single 1-Wire slave device is connected to the bus. After the Skip ROM command is complete, all slave devices on the 1-Wire bus are selected or active and communication proceeds to the Transport layer.

The Overdrive Skip ROM command operates in an identical manner except that running it also causes the receiving slave devices to shift communication speed from standard speed to overdrive speed. The Overdrive Skip ROM command byte itself (0x3C) is transmitted at standard speed. All subsequent communication is sent at overdrive speed.

16.3.4 Match ROM and Overdrive Match ROM Commands

The Match ROM command is used by the OWM to select one and only one slave 1-Wire device when the ROM ID of the device has already been determined. When transmitting this command, the master sends the command byte (that is, 55h for standard speed and 69h for overdrive speed) and then sends the entire 64-bit ROM ID for the device selected, least significant bit first.

During the transmission of the ROM ID by the master, all slave devices monitor the bus. As each bit is transmitted, each of the slave devices compares it against the corresponding bit of their ROM ID. If the bits match, the slave device continues to monitor the bus. If the bits do not match, the slave device transitions to the inactive state (waiting for a 1-Wire reset) and stops monitoring the bus.

At the end of the transmission, at most one slave device is active, which is the slave device whose ROM ID matched the ROM ID that was transmitted. All other slave devices are inactive. Communication then proceeds to the Transport layer for the device that was selected.

The Overdrive Match ROM command operates in an identical manner except that it also causes the slave device that is selected by the command to shift communication speed from standard speed to overdrive speed. The Overdrive Match ROM command byte (69h) and the 64-bit ROM ID bits are transmitted at standard speed. All subsequent communication is sent at overdrive speed.

16.3.5 Search ROM Command

The Search ROM command allows the OWM to determine the ROM ID values of all 1-Wire slave devices connected to the bus using an iterative search process. Each execution of the Search ROM command reveals the ROM ID of one slave device on the bus.

The operation of the Search ROM command resembles a combination of the Read ROM and Match ROM commands. First, all slaves on the bus transmit the least significant bit (Bit 0) of their ROM IDs. Next, all slaves on the bus transmit a complement of the same bit. By analyzing the two bits received, the master can determine whether the Bit 0 values were 0 for all slaves, 1 for all slaves, or a combination of the two. Next, the master selects which slaves remain activated for the next step in the Search ROM process by transmitting the Bit 0 value for the slaves it selects. All slaves whose Bit 0 matches the value transmitted by the master remain active, while slaves with a different Bit 0 value go to the inactive state and do not participate in the remainder of the Search ROM command.

Next, the same process is followed for Bit 1, then Bit 2, and so on until the 63rd bit (most significant bit) of the ROM ID is transmitted. At this point only one slave device remains active, and the master can either continue with communication at the Transport layer or issue a 1-Wire reset pulse to go back for another pass at the Search ROM command.

The [Book of iButton Standards](#) goes into more detail about the process that is used by the master to obtain ROM IDs of all devices on the 1-Wire bus using multiple executions of the Search ROM command. The algorithm resembles a binary tree search and is used regardless of how many devices are on the bus.

There is no overdrive equivalent version of the Search ROM command.

16.3.6 Search ROM Accelerator Operation

To allow the Search ROM command to process more quickly, the OWM module provides a special accelerator mode for use with the Search ROM command. This mode is activated by setting `OWM_CTRL_STAT.sra_mode` to 1.

When this mode is active, ROM IDs being processed by the Search ROM command are broken into 4-bit nibbles where the current 64-bit ROM ID varies with each pass through the search algorithm. Each 4-bit processing step is initiated by writing the 4-bit value to `OWM_DATA.tx_rx`. This causes the generation of twelve 1-Wire time slots by the OWM as each bit in the 4-bit value (starting with the LSB) results in a read of two bits (all active slaves transmitting bit N of their ROM IDs, then all active slaves transmitting the complement of bit N of their ROM ID), and then a write of a single bit by the OWM.

After the 4-bit processing stage is complete, the return value loaded into `OWM_DATA.tx_rx` consists of 8 bits. The low nibble (bits 0 through 3) contains the four discrepancy flags: one for each ID bit processed. If the discrepancy bit is set to 1, it means that either two slaves with differing ID bits in that position both responded (the 2 bits read were both zero), or that no slaves responded (the 2 bits read were both 1). If the discrepancy bit is set to 0, then the 2 bits read were complementary (either 0, 1 or 1, 0) meaning that there was no bus conflict.

In this way, at each step in the Search ROM command, the master either follows the ID of the responding slaves or deselects some of the slaves on the bus in case of a conflict. By the time the end of the 64-bit ROM ID is reached (the sixteenth 4-bit group processing step), the combination of all bits from the high nibbles of the received data are equal to the ROM ID of one of the slaves remaining on the bus. Subsequent passes through the Search ROM algorithm are used to determine additional slave ROM ID values until all slaves are identified. Refer to the *Book of iButton Standards* for a detailed explanation of the search function and possible variants of the search algorithm applicable to specific circumstances.

16.3.7 Resume Communication Command

If more than one 1-Wire slave device is on the bus, then the master must specify which one it wishes to communicate with each time a new 1-Wire command (starting with a reset pulse) begins. Using the commands discussed previously, this would normally involve sending the Match ROM command each time, which means the master must explicitly specify the full 64-bit ROM ID of the part it communicates with for each command.

The Resume Communication command provides a shortcut for this process by allowing the master to repeatedly select the same device for multiple commands without having to transmit the full ROM ID each time.

When the OWM selects a single device (using the Match ROM or Search ROM commands), an internal flag called the RC (for Resume Communication) flag is set in the slave device. (Only one device on the bus has this flag set at any one time; the Skip ROM command selects multiple devices, but the RC flag is not set by the Skip ROM command.)

When the master resets the 1-Wire bus, the RC flag remains set. At this point, it is possible for the master to send the Resume Communication command. This command does not have a ROM ID attached to it, but the device that has the RC flag set responds to this command by going to the active state while all other devices deactivate and drop off the 1-Wire bus.

Issuing any other ROM command clears the RC flag on all devices. So, for example, if a Match ROM command is issued for device A, its RC flag is set. The Resume Communication command can then be used repeatedly to send commands to device A. If a Match ROM command is then sent with the ROM ID of device B, the RC flag on device A will clear to 0, and the RC flag on device B is set.

16.4 1-Wire Operation

Once the OWM peripheral is correctly configured, then using the OWM peripheral to communicate with the 1-Wire network involves directing the OWM to generate the proper reset, read, and write operations to communicate with the 1-Wire slave devices used in a specific application.

The OWM manages the following 1-Wire protocol primitives directly in either Standard or Overdrive mode:

- 1-Wire bus reset (including detection of presence pulse from responding slave devices)
- Write single bit (a single write time slot)
- Write 8-bit byte, least significant bit first (eight write time slots)
- Read single bit (a single write-1 time slot)
- Read 8-bit byte, least significant bit first (eight write-1 time slots)
- Search ROM Acceleration Mode allowing the generation of four groups of three time slots (read, read, and write) from a single 4-bit register write to support the Search ROM command

16.4.1 Resetting the OWM

The first step in any 1-Wire communication sequence is to reset the 1-Wire bus. To direct the OWM module to complete a 1-Wire reset, write `OWM_CTRL_STAT.start_ow_reset` to 1. This generates a reset pulse and checks for a replying presence pulse from any connected slave devices.

Once the reset time slot is complete, the `OWM_CTRL_STAT.start_ow_reset` field is automatically cleared to zero. Then, the interrupt flag `OWM_INTFL.ow_reset_done` is set to 1 by hardware. This flag must be cleared by writing a 1 bit to the flag.

If a presence pulse is detected on the 1-Wire bus during the reset sequence (that should normally be the case unless no 1-Wire slave devices are present on the bus), the `OWM_CTRL_STAT.presence_detect` flag is also set to 1. This flag does not result in the generation of an interrupt.

16.5 1-Wire Data Reads

16.5.1 Reading a Single Bit Value from the 1-Wire Bus

The procedure for reading a single bit is like the procedure for writing a single bit because the operation is completed by writing a 1 bit that the slave device either leaves unchanged (to transmit a 1 bit) or overrides by forcing the line low (to transmit a 0 bit).

To read a single bit value from the 1-Wire Bus, complete the following steps:

1. Set `OWM_CFG.single_bit_mode` to 1. This setting causes the OWM to transmit/receive a single bit of data at a time instead of the default 8 bits.
2. Write `OWM_DATA.tx_rx` to 1. Only bit 0 of this field is used in this instance; the other bits in the field are ignored. Writing to the `OWM_DATA` register initiates the read of the bit on the 1-Wire bus.
3. Once the single-bit transmission is complete, hardware sets the interrupt flag `OWM_INTFL.tx_data_empty` to 1. This flag (that triggers an OWM module interrupt if `OWM_INTEN.tx_data_empty` is also set to 1) is cleared by writing a 1 to the flag.
4. As the hardware shifts the bit value out, it also samples the value returned from the slave device. Once this value is ready to read, the interrupt flag `OWM_INTFL.rx_data_ready` is set to 1. If `OWM_INTEN.rx_ready` is set to 1, an OWM module interrupt occurs.
5. Read `OWM_DATA.tx_rx` (only bit 0 is used) to determine the value returned by the slave device. Note that if no slave devices are present or the slaves are not communicating with the master, bit 0 remains set to 1.

16.5.2 Reading an 8-Bit Value from the 1-Wire Bus

The procedure for reading an 8-bit byte is like the procedure for writing an 8-bit byte because the operation is completed by writing eight 1 bits that the slave device either leaves unchanged (to transmit 1 bits) or overrides by forcing the line low (to transmit 0 bits).

1. Set `OWM_CFG.single_bit_mode` to 0. This setting causes the OWM to transmit/receive in the default 8-bit mode.
2. Write `OWM_DATA.tx_rx` to 0xFFh.
3. Once the 8-bit transmission completes, hardware sets the interrupt flag `OWM_INTFL.tx_data_empty` to 1. This flag (that triggers an OWM module interrupt if `OWM_INTEN.tx_data_empty` is also set to 1) is cleared by writing a 1 to the flag.
4. As the hardware shifts the bit values out, it also samples the values returned from the slave device. Once the full 8-bit value is ready to be read, the interrupt flag `OWM_INTFL.rx_data_ready` is set to 1. If `OWM_INTEN.rx_ready` is set to 1, an OWM module interrupt occurs.
5. Read `OWM_DATA.tx_rx` to determine the 8-bit value returned by the slave device. *Note that if no slave devices are present or the slave devices are not communicating with the master, the return value 0xFF is the same as the transmitted value.*

16.6 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 16-4: OWM Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 16-4: OWM Register Summary

| Offset | Register | Description |
|----------|---------------------------------|-------------------------------|
| [0x0000] | OWM_CFG | OWM Configuration Register |
| [0x0004] | OWM_CLK_DIV_1US | OWM Clock Divisor Register |
| [0x0008] | OWM_CTRL_STAT | OWM Control/Status Register |
| [0x000C] | OWM_DATA | OWM Data Buffer Register |
| [0x0010] | OWM_INTFL | OWM Interrupt Flag Register |
| [0x0014] | OWM_INTEN | OWM Interrupt Enable Register |

16.6.1 Register Details

Table 16-5: OWM Configuration Register

| OWM Configuration Register | | OWM_CFG | | | [0x0000] |
|----------------------------|-------------------|---------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 7 | int_pullup_enable | R/W | 0 | Internal Pullup Enable Set this field to enable the internal pullup resistor. 0: Internal pullup disabled. 1: Internal pullup enabled. | |
| 6 | overdrive | R/W | 0 | Overdrive Enable Set this field to 1 to enable overdrive mode for 1-Wire communications. Clearing this field sets 1-Wire communications to standard speed. 0: Overdrive mode disabled, standard speed mode. 1: Overdrive mode enabled. | |
| 5 | single_bit_mode | R/W | 0 | Bit Mode Enable When set to 1, only a single bit at a time is transmitted and received (LSB of OWM_DATA) rather than the whole byte. 0: Byte mode enabled, single bit mode disabled. 1: Single bit mode enabled, byte mode disabled. | |
| 4 | ext_pullup_enable | R/W | 0 | External Pullup Enable Enables external FET pullup when the 1-Wire master is idle. FET is designed to pull the wire high regardless of its enable state (that is, high or low). Idle means the 1-Wire master is idle, and there are no 1-Wire accesses in progress. 0: External pullup pin is not driven to high. 1: External pullup pin is driven high when the 1-Wire bus is idle, actively pulling the 1-Wire IO high. | |

| OWM Configuration Register | | | OWM_CFG | | [0x0000] |
|----------------------------|-----------------|--------|---------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 3 | ext_pullup_mode | R/W | 0 | External Pullup Mode Provides an extra output to control an external pullup. For long wires, a pullup resistor strong enough to pull the wire high in a reasonable amount of time might need to be so strong that it would be difficult to drive the line low. In this case, implement an external FET to actively drive the wire high for a brief amount of time. Then, let the resistor keep the line high. | |
| 2 | bit_bang_en | R/W | 0 | Bit-Bang Mode Enable Enable bit-bang control of the I/O pin. If this bit is set to 1, OWM_CTRL_STAT.bit_bang_oe controls the state of the I/O pin. 0: Bit-bang mode disabled. 1: Bit-bang mode enabled. | |
| 1 | force_pres_det | R/W | 1 | Presence Detect Force Setting this bit to 1 drives the OWM_IO pin low during presence detection. Use this bit field to prevent a large number of 1-Wire slaves on the bus from all responding at different times, which might cause ringing. When this bit is set to 1, the OWM_CTRL_STAT.presence_detect bit is always set as the result of a 1-Wire reset even if no slave devices are present on the bus. 0: OWM_IO pin floats during presence detection portion of 1-Wire reset. 1: OWM_IO pin is driven low during presence detection portion of 1-Wire reset. | |
| 0 | long_line_mode | R/W | 0 | Long Line Mode Enable Selects alternate timings for 1-Wire communication. The recommended setting depends on the length of the wire. For lines less than 40 meters, 0 should be used. Setting this bit to 0 leaves the write one release, the data sampling, and the time-slot recovery times at approximately 5µs, 15µs, and 7µs, respectively. Setting this bit to 1 enables long line mode timings during standard mode communications. This mode moves the write one release, the data sampling, and the time-slot recovery times out to approximately 8µs, 22µs, and 14µs, respectively. 0: Standard operation for lines less than 40 meters. 1: Long Line mode enabled. | |

Table 16-6: OWM Clock Divisor Register

| OWM Clock Divisor | | | OWM_CLK_DIV_1US | | [0x0004] |
|-------------------|---------|--------|-----------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R | 0 | Reserved for Future Use Do not modify this field. | |
| 7:0 | divisor | R/W | 0 | OWM Clock Divisor Divisor for the OWM peripheral clock. The target is to achieve a 1MHz clock. See the Clock Configuration section for details. 0x00: OWM clock disabled. $0x01: f_{owmclk} = \frac{f_{PCLK}}{1}$ $0x02: f_{owmclk} = \frac{f_{PCLK}}{2}$... $0xFF: f_{owmclk} = \frac{f_{PCLK}}{255}$ | |

Table 16-7: OWM Control Status Register

| OWM Control Status | | OWM_CTRL_STAT | | | [0x0008] |
|--------------------|-----------------|---------------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 7 | presence_detect | RO | 0 | Presence Detect Flag Set to 1 when a presence pulse is detected from one or more slaves during the 1-Wire reset sequence. 0: No presence detect pulse during previous 1-Wire reset sequence. 1: Presence detect pulse on bus during previous 1-Wire reset sequence. | |
| 6:5 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 4 | od_spec_mode | RO | 0 | Overdrive Spec Mode Returns the version of the overdrive spec. | |
| 3 | ow_input | RO | - | OWM_IN State Returns the current logic level on the OWM_IO pin. 0: OWM_IO pin is low. 1: OWM_IO pin is high. | |
| 2 | bit_bang_oe | R/W | 0 | OWM Bit-Bang Output When bit-bang mode is enabled (<i>OWM_CFG.bit_bang_en</i> = 1), this bit sets the state of the OWM_IO pin. Setting this bit to 1 drives the OWM_IO pin low. Setting this bit to 0 releases the line, allowing the OWM_IO pin to be pulled high by the pullup resistor or held low by a slave device. 0: OWM_IO pin floating. 1: Drive OWM_IO pin to low state. | |
| 1 | sra_mode | R/W | 0 | Search ROM Accelerator Enable Enable Search ROM Accelerator mode. This mode is used to identify slaves and their addresses that are attached to the 1-Wire bus. 0: Search ROM accelerator mode disabled. 1: Search ROM accelerator mode enabled. | |
| 0 | start_ow_reset | R/W | 0 | Start 1-Wire Reset Pulse Write 1 to start a 1-Wire reset sequence. Automatically cleared by the OWM hardware when the reset sequence is complete. 0: 1-Wire reset sequence complete or inactive. 1: Start a 1-Wire reset sequence. | |

Table 16-8: OWM Data Buffer Register

| OWM Data | | OWM_DATA | | | [0x000C] |
|----------|-------|----------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 7:0 | tx_rx | R/W | 0 | OWM Data Field Writing to this field sets the transmit data and initiates a 1-Wire data transmit cycle. Reading from this field returns the data received by the master during the last 1-Wire data transmit cycle. | |

Table 16-9: OWM Interrupt Flag Register

| OWM Interrupt Flag | | OWM_INTFL | | | [0x0010] |
|--------------------|---------------|-----------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 4 | line_low | R/W1C | 0 | Line Low Flag If this flag is set, the OWM_IO pin was in a low state. Write 1 to clear this flag. | |
| 3 | line_short | R/W1C | 0 | Line Short Flag The OWM hardware detected a short on the OWM_IO pin. Write 1 to clear this flag. | |
| 2 | rx_data_ready | R/W1C | 0 | RX Data Ready Data received from the 1-Wire bus and is available in the <i>OWM_DATA.tx_rx</i> field. Write 1 to clear this flag. 0: RX data not available. 1: Data received and is available in the <i>OWM_DATA.tx_rx</i> field. | |
| 1 | tx_data_empty | R/W1C | 0 | TX Empty The OWM hardware automatically sets this interrupt flag when the data transmit is complete. Write 1 to clear this flag. 0: Either no data was sent or the data in the <i>OWM_DATA.tx_rx</i> field has not completed transmission. 1: Data in the <i>OWM_DATA.tx_rx</i> field was transmitted. | |
| 0 | ow_reset_done | R/W1C | 0 | Reset Complete This flag is set when a 1-Wire reset sequence completes. To start a 1-Wire reset sequence, see <i>OWM_CTRL_STAT.start_ow_reset</i> . Write 1 to clear this flag. 0: 1-Wire reset sequence not complete or bus idle. 1: 1-Wire reset sequence complete. | |

Table 16-10: OWM Interrupt Enable Register

| OWM Interrupt Enable | | OWM_INTEN | | | [0x0014] |
|----------------------|---------------|-----------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | RO | 0 | Reserved for Future Use Do not modify this field. | |
| 4 | line_low | R/W | 0 | Line Low Interrupt Enable I/O pin low detected interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled. | |
| 3 | line_short | R/W | 0 | Line Short Interrupt Enable I/O pin short detected interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled. | |
| 2 | rx_data_ready | R/W | 0 | Receive Data Ready Interrupt Enable RX data ready interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled. | |
| 1 | tx_data_empty | R/W | 0 | Transmit Data Empty Interrupt Enable TX data empty interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled. | |

| OWM Interrupt Enable | | OWM_INTEN | | | [0x0014] |
|----------------------|---------------|-----------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 0 | ow_reset_done | R/W | 0 | 1-Wire Reset Sequence Complete Interrupt Enable 1-Wire reset sequence completed. 0: Interrupt disabled. 1: Interrupt enabled. | |

Preliminary Draft 08/31/2020

17. Real-Time Clock (RTC)

17.1 Overview

The Real-Time Clock (RTC) is a 32-bit binary timer that keeps the time of day up to 136 years. It provides time-of-day and sub-second alarm functionality in the form of system interrupts.

The RTC operates on an external 32.768kHz time base. It can be generated from the internal crystal oscillator driving an external 32.768kHz crystal between the 32KIN and 32KOUT pins, or a 32.768kHz square wave driven directly into the 32KIN pin. Refer to the datasheet for the required electrical characteristics of the external crystal.

A user-configurable, digital frequency trim is provided for applications requiring higher accuracy.

The 32-bit seconds register *RTC_SEC* is incremented every time there is a rollover of the *RTC_SSEC.ssec* sub-seconds field.

Two alarm functions are provided:

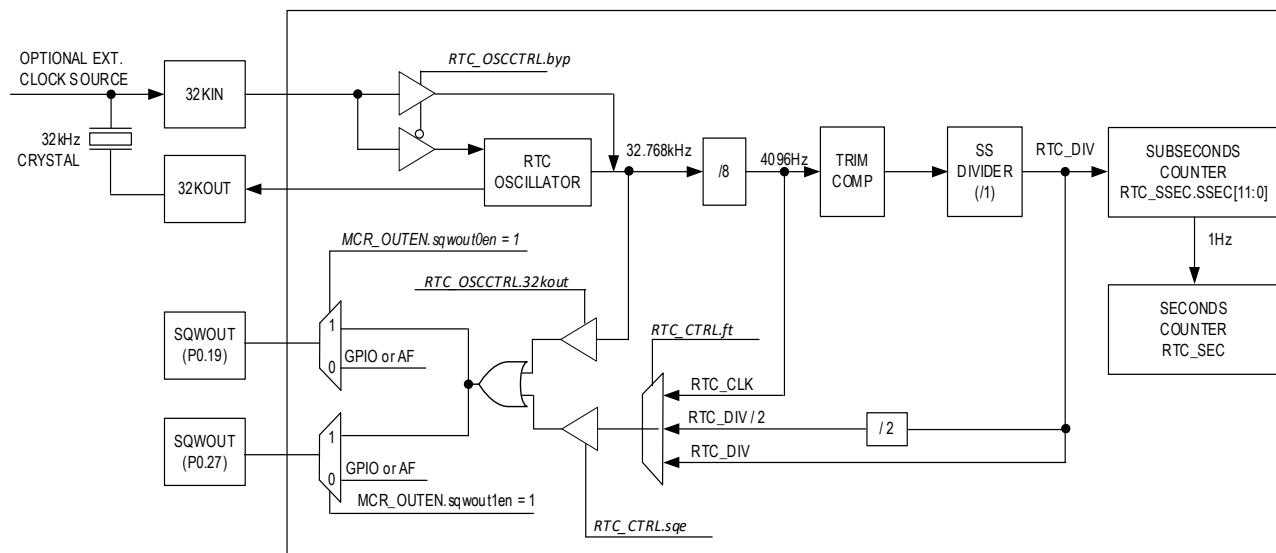
1. A programmable time-of-day alarm provides a single event, alarm timer using the *RTC_TODA* alarm register, *RTC_SEC* register, and *RTC_CTRL.ade* field.
2. A programmable sub-second provides a recurring alarm using the *RTC_SSECA* and *RTC_CTRL.ase* field

The RTC is powered in the always-on domain or if applicable, while there is a valid voltage on the V_{COREA} device pin.

Disabling the RTC stops incrementing *RTC_SSEC*, *RTC_SEC*, and the internal RTC sub-second counter, but preserves their current value. The 32kHz oscillator is not affected by the *RTC_CTRL.rtce* field.

The RTC increments the *RTC_TRIM.vrtc_tmr* field every 32 seconds while the RTC is enabled.

Figure 17-1: MAX78000 RTC Block Diagram (12-bit Sub-Second Counter)



17.2 Instances

One instance of the RTC peripheral is provided. The RTC counter and alarm registers are shown in [Table 17-1: MAX78000 RTC Counter and Alarm Registers](#).

Table 17-1: MAX78000 RTC Counter and Alarm Registers

| Field | Length | Counter Increment | Minimum | Maximum | Description |
|---------------------------|--------|-------------------|---------|---------|------------------------------|
| RTC_SEC | 32 | 1s | 1s | 136 yrs | Seconds Counter Register |
| RTC_SSEC | 12 | 244µs (1/4kHz) | 244µs | 1s | Sub-Seconds Counter Register |
| RTC_TODA | 20 | 1s | 1s | 12 days | Time-of-Day Alarm Register |
| RTC_SSECA | 32 | 244 µs (1/4kHz) | 244 µs | 12 days | Sub-Second Alarm Register |

17.3

Register Access Control

Access protection mechanisms prevent software from accessing critical registers and fields while RTC while hardware is updating them. Monitoring the [RTC_CTRL.busy](#) and [RTC_CTRL.rdy](#) fields allows software to determine when it is safe to write to registers and when registers will return valid results.

Table 17-2: RTC Register Access

| Register | Field | Read Access | Write Access | Busy = 1 during write | Description |
|-----------------------------|----------------|--|--|-----------------------|------------------------------|
| RTC_SEC | All | RTC_CTRL.busy = 0 RTC_CTRL.rdy = 1 [†] | RTC_CTRL.busy = 0 RTC_CTRL.rdy = 1 [†] | Y | Seconds Counter Register |
| RTC_SSEC | ssec | RTC_CTRL.busy = 0 RTC_CTRL.rdy = 1 [†] | RTC_CTRL.busy = 0 RTC_CTRL.rdy = 1 [†] | Y | Sub-Seconds Counter Register |
| RTC_TODA | All | Always | RTC_CTRL.busy = 0 RTC_CTRL.adc = 0 RTC_CTRL.rtce = 0 | Y | Time-of-Day Alarm Register |
| RTC_SSECA | All | Always | RTC_CTRL.busy = 0 RTC_CTRL.adc = 0 RTC_CTRL.rtce = 0 | Y | Sub-Second Alarm Register |
| RTC_TRIM | All | Always | RTC_CTRL.busy = 0 RTC_CTRL.we = 1 | Y | Trim Register |
| RTC_OSCCTRL | All | Always | RTC_CTRL.we = 1 | N | Oscillator Control Register |
| RTC_CTRL | rtce | Always | RTC_CTRL.busy = 0 RTC_CTRL.we = 1 | Y | RTC Enable field |
| | All other bits | Always | ++ | Y | |

[†] See [RTC_SEC and RTC_SSEC Read Access Control](#) for details.

⁺⁺ See [RTC_CTRL.busy](#) for limitations on specific bits.

17.3.1 [RTC_SEC and RTC_SSEC Read Access Control](#)

Software reads of the [RTC_SEC](#) and [RTC_SSEC](#) registers will return invalid results if the read operation occurs on the same cycle that the register is being updated by hardware. To avoid this, hardware sets [RTC_CTRL.rdy](#) to 1 for 120 µs when the [RTC_SEC](#) and [RTC_SSEC](#) registers are valid and can be read.

Alternatively, the software can set [RTC_CTRL.acre](#) to 1 to allow asynchronous reads of both the [RTC_SEC](#) and [RTC_SSEC](#) registers.

Software can use three methods to ensure valid results when reading *RTC_SEC* and *RTC_SSEC*:

1. Software clears *RTC_CTRL.rdy* to 0. Software polls *RTC_CTRL.rdy* until it reads 1 before reading the registers. The software has approximately 120µs to read the *RTC_SEC* and *RTC_SSEC* registers to ensure accuracy.
2. Set the *RTC_CTRL.rdye* field to 1 to generate an RTC interrupt when the next update cycle is complete and hardware sets *RTC_CTRL.rdy* to 1. RTC interrupt must be serviced and *RTC_SEC* and/or *RTC_SSEC* registers must be read while *RTC_CTRL.rdy* = 1 to ensure accuracy. This avoids the software overhead associated with polling to observe the state of *RTC_CTRL.rdy*.
3. Set *RTC_CTRL.acre* to 1 to allow asynchronous reads of both the *RTC_SEC* and *RTC_SSEC* registers. Multiple consecutive reads of *RTC_SEC* and *RTC_SSEC* must be executed until two consecutive reads are identical to ensure data accuracy.

17.3.2 RTC Write Access Control

The read-only status field *RTC_CTRL.busy* is set to 1 by hardware following a software instruction that writes to specific registers. The bit remains 1 while the software updates are being synchronized into the RTC. Software should not write to any of the registers until hardware indicates the synchronization is complete by clearing *RTC_CTRL.busy* to 0.

17.4 RTC Alarm Functions

The RTC provides time-of-day and sub-second interval alarm functions. The time-of-day alarm is implemented by matching the count values in the counter register with the value stored in the alarm register. The sub-second interval alarm provides an auto-reload timer that is driven by the trimmed RTC clock source.

17.4.1 Time-of-Day Alarm

Program the RTC Time-of-Day Alarm register (*RTC_TODA*) to configure the time-of-day-alarm. The alarm triggers when the value stored in *RTC_TODA.tod_alarm* matches the lower 20 bits of the *RTC_SEC* seconds count register. This allows programming the time-of-day-alarm to any future value between 1 second and 12 days relative to the current time with a resolution of 1 second. You must disable the time-of-day alarm before changing the *RTC_TODA.tod* field.

When the alarm occurs, a single event sets the Time-of-Day Alarm Interrupt Flag (*RTC_CTRL.aldf*) to 1.

Setting the *RTC_CTRL.aldf* bit to 1 in software results in an interrupt request to the processor if the Alarm Time-of-Day Interrupt Enable (*RTC_CTRL.ade*) bit is set to 1, and the RTC's system interrupt enable is set.

17.4.2 Sub-Second Alarm

The *RTC_SSECA* and *RTC_CTRL.ase* field control the sub-second alarm. Writing *RTC_SSECA* sets the starting value for the sub-second alarm counter. Writing the Sub-Second Alarm Enable (*RTC_CTRL.ase*) bit to 1 enables the sub-second alarm. Once enabled, an internal alarm counter begins incrementing from the *RTC_SSECA* value. When the counter rolls over from 0xFFFF FFFF to 0x0000 0000, hardware sets the *RTC_CTRL.alsf* bit triggering the alarm. At the same time, hardware also reloads the counter with the value previously written to *RTC_SSECA.rssa*.

You must disable the sub-second interval alarm, *RTC_CTRL.ase*, prior to changing the interval alarm value, *RTC_SSECA*.

The delay (uncertainty) associated with enabling the sub-second alarm is up to one period of the sub-second clock. This uncertainty is propagated to the first interval alarm. Thereafter, if the interval alarm remains enabled, the alarm triggers after each sub-second interval as defined without the first alarm uncertainty because the sub-second alarm is an auto-reload timer. Enabling the sub-second alarm with the sub-second alarm register set to 0 (*RTC_SSECA* = 0) results in the maximum sub-second alarm interval.

17.4.3 RTC Interrupt and Wakeup Configuration

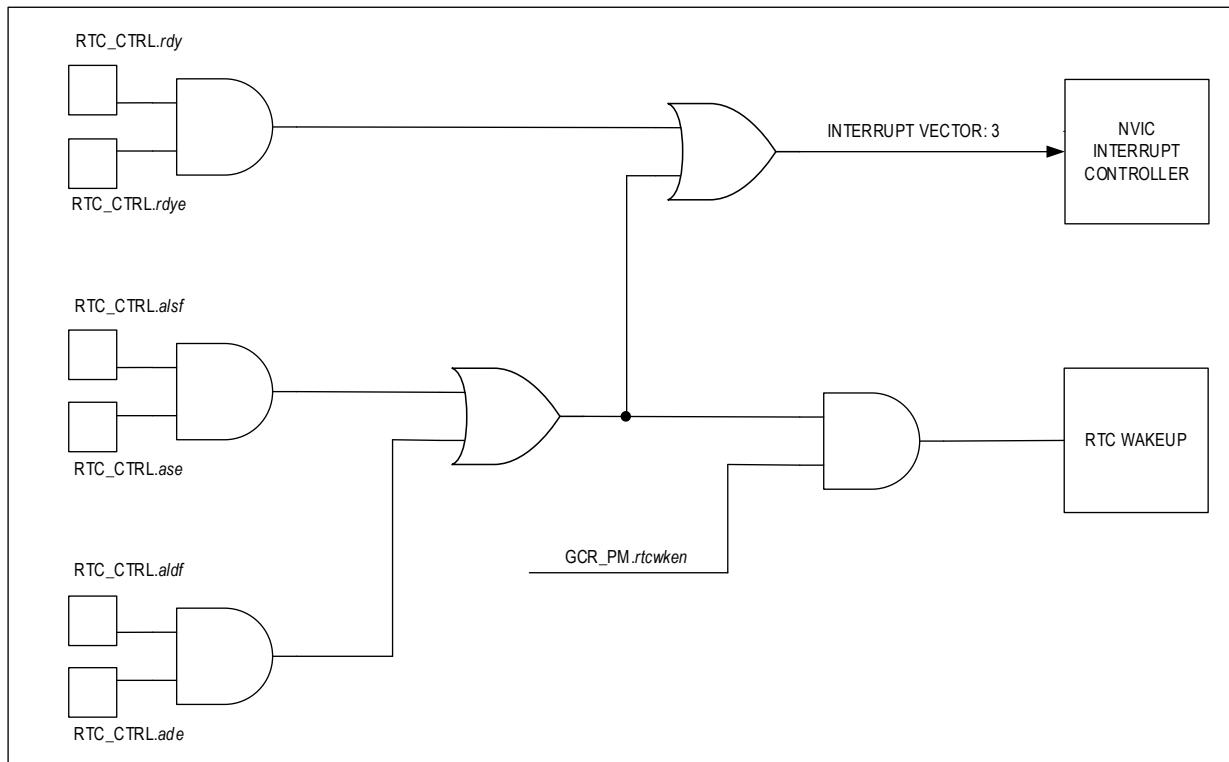
The following are a list of conditions that, when enabled, can generate an RTC interrupt.

1. Time-of-Day Alarm
2. Sub-second Alarm
3. *RTC_CTRL.rdy* field asserted high, signaling read access permitted

The RTC can be configured so the Time-of-day and sub-second alarms are a wakeup source for exiting the following low power modes:

1. BACKUP
2. DEEP SLEEP
3. UPM
4. SLEEP

Figure 17-2: RTC Interrupt/Wakeup Diagram Wakeup Function



Use this procedure to enable the RTC as a wakeup source:

1. Software configures the RTC interrupt enable bits so one or more interrupt conditions will generate an RTC interrupt.
2. Create a RTC IRQ handler function and register the address of the RTC IRQ handler using the NVIC.
3. Software sets *GCR_PM.rtcwken* to 1 to enable the system wakeup for by the RTC.
4. Software places the device into the desired low power mode. Refer to section *Operating Modes*

17.4.4 Square Wave Output

The RTC can output a 50% duty cycle square wave signal derived from the 32kHz oscillator on a selected device pin. Refer to [Table 17-3](#): for the device pins, frequency options, and control fields specific to this device. Frequencies noted as compensated are used during the RTC frequency calibration procedure because they incorporate the frequency adjustments provided by the digital trim function.

Table 17-3: MAX78000 RTC Square Wave Output Configuration

| Function | Option | Control Field |
|-------------------------|---------------------|--|
| Output Pin [†] | P3.1: SQWOUTL | <i>MCR_OUTEN.sqwout_en = 1</i> |
| Frequency Select | 1Hz (Compensated) | <i>RTC_OSCCTRL.freq_sel = 0b00</i> |
| | 512Hz (Compensated) | <i>RTC_OSCCTRL.freq_sel = 0b01</i> |
| | 4kHz | <i>RTC_OSCCTRL.freq_sel = 0b10</i> |
| | 32kHz | <i>RTC_OSCCTRL.freq_sel = 0bxx</i> <i>RTC_OSCCTRL.32k_out = 1</i> |
| Output Enable | 1Hz (Compensated) | <i>RTC_CTRL.sqe = 1</i> <i>RTC_OSCCTRL.32k_out = 0</i> |
| | 512Hz (Compensated) | <i>RTC_CTRL.sqe = 1</i> <i>RTC_OSCCTRL.32k_out = 0</i> |
| | 4kHz | <i>RTC_CTRL.sqe = 1</i> <i>RTC_OSCCTRL.32k_out = 0</i> |
| | 32kHz | <i>RTC_OSCCTRL.32k_out = 1</i> |

[†]Note: The square wave output can be enabled on more than one pin if desired.

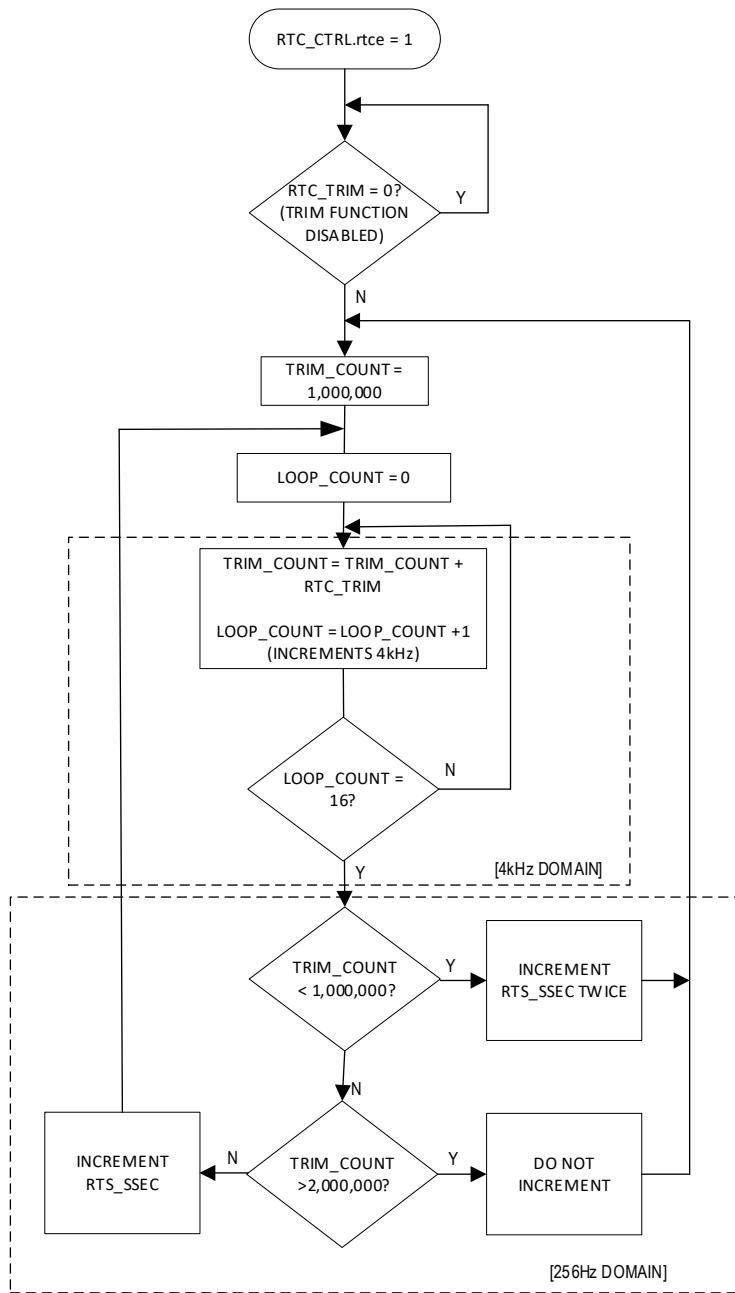
Use the following procedure to generate the square wave:

1. Software configures the fields shown in [Table 17-3](#): to select the desired frequency.
2. If more than one output pin is available, software configures the fields shown in [Table 17-3: MAX78000 RTC Square Wave Output Configuration](#) to select the output pin.
3. Software configures the fields shown in [Table 17-3](#): to enable the square wave output.

17.5 RTC Calibration

A digital trim facility provides the ability to compensate for RTC inaccuracies of up to $\pm 127\text{ppm}$ when compared against an external reference clock. The trimming function utilizes an independent, dedicated timer which increments the sub-second register based on a user-supplied, 2's compliment value in the *RTC_TRIM* register as shown in [Figure 17-3: Internal Implementation of Digital Trim, 4kHz](#).

Figure 17-3: Internal Implementation of Digital Trim, 4kHz



Preliminary Draft 08/31/2020

Complete the following steps to perform an RTC calibration:

1. If not already, software configures and enables one of the compensated calibration frequencies as described in section [Square Wave Output](#).
2. Measure the frequency on the square wave output pin and compute the deviation from an accurate reference clock.
3. Software clears [*RTC_CTRL.rdy*](#) to 0.
4. Wait for [*RTC_CTRL.rdy* = 1](#) by:
 - a. Software sets [*RTC_CTRL.rdye*](#) to 1 to generate an interrupt when [*RTC_CTRL.rdy* = 1](#), or
 - b. Software polls [*RTC_CTRL.rdy*](#) until the field = 1.
5. Software polls until [*RTC_CTRL.busy* = 0](#) to make sure any previous operations are complete
6. Software sets [*RTC_CTRL.we*](#) to 1 to allow access to [*RTC_TRIM*](#).
7. Software writes to [*RTC_TRIM*](#) register as desired to correct for measured inaccuracy.
8. Hardware clears [*RTC_CTRL.busy* = 0](#).
9. Software clears [*RTC_CTRL.we*](#) to 0.
10. Repeat the process as needed until the desired accuracy is achieved

17.6 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 17-4: RTC Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 17-4: RTC Register Summary

| Offset | Register | Description |
|----------|------------------------------------|--|
| [0x0000] | <i>RTC_SEC</i> | RTC Seconds Counter Register |
| [0x0004] | <i>RTC_SSEC</i> | RTC Sub-Second Counter Register |
| [0x0008] | <i>RTC_TODA</i> | RTC Time-of-Day Alarm Register |
| [0x000C] | <i>RTC_SSECA</i> | RTC Sub-Second Alarm Register |
| [0x0010] | <i>RTC_CTRL</i> | RTC Control Register |
| [0x0014] | <i>RTC_TRIM</i> | RTC 32KHz Oscillator Digital Trim Register |
| [0x0018] | <i>RTC_OSCCTRL</i> | RTC 32KHz Oscillator Control Register |

17.6.1 Register Details

Table 17-5: RTC Seconds Counter Register

| RTC Seconds Counter | | | RTC_SEC | [0x0000] |
|---------------------|-------|--------|---------|---|
| Bits | Field | Access | Reset | Description |
| 31:0 | sec | R/W | 0 | Seconds Counter This register is a binary count of seconds. |

Table 17-6: RTC Sub-Second Counter Register (12-bit)

| RTC Sub-Seconds Counter | | | RTC_SSEC | | [0x0004] |
|-------------------------|-------|--------|----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:12 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 11:0 | ssec | R/W | 0 | Sub-Seconds Counter (12-bit) <i>RTC_SEC</i> increments when this field rolls from 0xFFFF to 0x0000 | |

Table 17-7: RTC Time-of-Day Alarm Register

| RTC Time-of-Day Alarm | | | RTC_TODA | | [0x0008] |
|-----------------------|-----------|--------|----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:20 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 19:0 | tod_alarm | R/W | 0 | Time-of-Day Alarm Sets the time-of-day alarm from 1 second up to 12-days. When this field matches <i>RTC_SEC</i> [19:0], an RTC system interrupt is generated. | |

Table 17-8: RTC Sub-Second Alarm Register

| RTC Sub-Second Alarm | | | RTC_SSECA | | [0x000C] |
|----------------------|------------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | ssec_alarm | R/W | 0 | Sub-second Alarm (4KHz) Sets the starting and reload value of the internal sub-second alarm counter. The internal counter increments and generates an alarm when the internal counter rolls from 0xFFFF FFFF to 0x0000 0000. | |

Table 17-9: RTC Control Register

| RTC Control Register | | | RTC_CTRL | | [0x0010] |
|----------------------|-------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 15 | we | R/W | 0* | Write Enable This field controls access to the <i>RTC_TRIM</i> register, the RTC enable (<i>RTC_CTRL.rtce</i>) fields. 1: Writes to the <i>RTC_TRIM</i> register and the <i>RTC_CTRL.rtce</i> field are allowed. 0: Writes to the <i>RTC_TRIM</i> register and the <i>RTC_CTRL.rtce</i> field are ignored. <i>*Note: Reset on System Reset, Soft Reset, and <i>GCR_RSTD.rtc</i> assertion.</i> | |
| 14 | acre | R/W | 0 | Asynchronous Counter Read Enable Set this field to 1 to allow direct read access of the <i>RTC_SEC</i> and <i>RTC_SSEC</i> registers without waiting for <i>RTC_CTRL.rdy</i> . Multiple consecutive reads of <i>RTC_SEC</i> and <i>RTC_SSEC</i> must be executed until two consecutive reads are identical to ensure data accuracy. 0: <i>RTC_SEC</i> and <i>RTC_SSEC</i> registers are synchronized and should only be accessed while <i>RTC_CTRL.rdy</i> = 1. 1: <i>RTC_SEC</i> and <i>RTC_SSEC</i> registers are asynchronous and will require software interaction to ensure data accuracy. | |
| 13 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |
| 12:11 | - | R/W | 0 | Reserved for Future Use Do not modify this field. | |

| RTC Control Register | | | RTC_CTRL | | [0x0010] |
|----------------------|-------|--------|----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 10:9 | ft | R/W | 0* | Frequency Output Select Selects the RTC-derived frequency to output on the square wave output pin. See Table 17-3: MAX78000 RTC Square Wave Output Configuration for configuration details. 0b00: 1Hz (Compensated) 0b01: 512Hz (Compensated) 0b1x: 4kHz <i>*Note: Reset on POR only.</i> | |
| 8 | sqe | R/W | 0* | Square Wave Output Enable Enables the square wave output. Table 17-3: MAX78000 RTC Square Wave Output Configuration 0: Disabled. 1: Enabled. <i>*Note: Reset on POR only.</i> | |
| 7 | alsf | R/W | 0* | Sub-second Alarm Interrupt Flag This interrupt flag is set when a sub-second alarm condition occurs. This flag is a wake-up source for the processor. 0: No sub-second alarm pending. 1: Sub-second interrupt pending. <i>*Note: Reset on POR only.</i> | |
| 6 | aldf | R/W | 0* | Time-of-Day Alarm Interrupt Flag This interrupt flag is set by hardware when a time-of-day alarm occurs. 0: No Time-of-Day alarm interrupt pending. 1: Time-of-day interrupt pending. <i>*Note: Reset on POR only.</i> | |
| 5 | rdye | R/W | 0* | RTC Ready Interrupt Enable 0: Disabled. 1: Enabled. <i>*Note: Reset on System Reset, Soft Reset, and GCR_RST0 rtc assertion.</i> | |
| 4 | rdy | R/W0 | 0* | RTC Ready This bit is set to 1 for 120µs by hardware once a hardware update of the RTC_SEC and RTC_SSEC registers has occurred. Software should read RTC_SEC and RTC_SSEC while this hardware bit is set to 1. Software can clear this bit at any time. An RTC interrupt will be generated if RTC_CTRL.rdye = 1. 0: Software reads of RTC_SEC and RTC_SSEC are invalid. 1: Software reads of RTC_SEC and RTC_SSEC are valid. <i>*Note: Reset on System Reset, Soft Reset, and GCR_RST0 rtc assertion.</i> | |

| RTC Control Register | | | RTC_CTRL | | [0x0010] |
|----------------------|-------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 3 | busy | RO | 0* | RTC Busy Flag This field is set to 1 by hardware to indicate a register update is in progress when software writes to: <ul style="list-style-type: none"> • RTC_SEC register • RTC_SSEC register • RTC_TRIM register The following bits cannot be written if this field is set: <ul style="list-style-type: none"> • RTC_CTRL.rtce • RTC_CTRL.adc • RTC_CTRL.ase • RTC_CTRL.rdye • RTC_CTRL.aldf • RTC_CTRL.alsf • RTC_CTRL.sqe • RTC_CTRL.ft • RTC_CTRL.acre This field is automatically cleared by hardware when the update is complete. Software should poll this field for 0 after changing a RTC register, prior to making any other RTC register modifications. <p>0: RTC not busy. 1: RTC busy.</p> <p>*Note: Reset on POR only.</p> | |
| 2 | ase | R/W | 0* | Sub-Second Alarm Interrupt Enable Check the RTC_CTRL.busy flag after writing to this field to determine when the RTC synchronization is complete. <p>0: Disable. 1: Enable.</p> <p>*Note: Reset on POR only.</p> | |
| 1 | ade | R/W | 0* | Time-of-Day Alarm Interrupt Enable Check the RTC_CTRL.busy flag after writing to this field to determine when the RTC synchronization is complete. <p>0: Disable. 1: Enable.</p> <p>*Note: Reset on POR only.</p> | |
| 0 | rtce | R/W | 0* | Real-Time Clock Enable The RTC write enable (RTC_CTRL.we) bit must be set and RTC Busy (RTC_CTRL.busy) must read 0 before writing to this field. After writing to this bit, check the RTC_CTRL.busy flag for 0 to determine when the RTC synchronization is complete. <p>0: Disabled. 1: Enabled.</p> <p>*Note: Reset on POR only.</p> | |

Table 17-10: RTC 32KHz Oscillator Digital Trim Register

| RTC 32KHz Oscillator Digital Trim | | | RTC_TRIM | | [0x0014] |
|-----------------------------------|----------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:8 | vrtc_tmr | R/W | 0* | VRTC Time Counter Hardware increments this field every 32s while the RTC is enabled. <p>*Note: Reset on POR only.</p> | |

| RTC 32KHz Oscillator Digital Trim | | | RTC_TRIM | [0x0014] |
|-----------------------------------|-------|--------|----------|---|
| Bits | Field | Access | Reset | Description |
| 7:0 | trim | R/W | 0* | RTC Trim This field specifies the 2s complement value of the trim resolution. Each increment or decrement of the field adds or subtracts 1ppm at each 4kHz clock value with a maximum correction of ± 127ppm. <i>*Note: Reset on POR only.</i> |

Table 17-11: RTC 32KHz Oscillator Control Register

| RTC Oscillator Control | | | RTC_OSCCTRL | [0x0018] |
|------------------------|--------|--------|-------------|---|
| Bits | Field | Access | Reset | Description |
| 31:6 | - | R/W | 0 | Reserved for Future Use Do not modify this field. |
| 5 | 32kout | R/W | 0 | RTC Square Wave Output 0: Disabled. 1: Enables the 32kHz oscillator output or the external clock source is output on square wave output pin. See Table 17-3 : for configuration details. <i>*Note: Reset on POR only.</i> |
| 4 | bypass | R/W | 0 | RTC Crystal Bypass This field disables the RTC oscillator and allows an external clock source to be driven on the 32KIN pin. 0: Disable bypass. RTC time base is external 32kHz crystal. 1: Enable bypass. RTC time base is external square wave driven on 32KIN. <i>*Note: Reset on POR only.</i> |
| 3:0 | - | R/W | 0b1001 | Reserved for Future Use Software must not modify this field from its current value. |

18. Timers (TMR/LPTMR)

Multiple 32-bit and dual 16-bit, reloadable timers are provided.

The features include:

- Operation as single 32-bit counter or single/dual 16-bit counters
- Programmable prescaler with values from 1 to 4096
- Non-overlapping PWM output generation with configurable off-time
- Capture, compare, and capture/compare capability
- Timer input and output signals available, mapped as alternate function
- Configurable input pin for event triggering, clock gating, or capture signal
- Timer output pin for event output and PWM signal generation
- Independent interrupts

Certain instances (denoted LPTMR) seen in *Table 18-1: MAX78000 TMR/LPTMR Instances* can be configured to operate in low power modes and wake the device from the low-power back to ACTIVE mode

Each timer provides multiple operating modes. See *Table 18-2: MAX78000 I/O Signal Configuration* for which operating modes are available for each configuration:

- One-Shot: Timer counts up to terminal value then halts.
- Continuous: Timer counts up to terminal value then repeats.
- Counter: Timer counts input edges received on timer input pin.
- Pulse Width Modulated (PWM) / PWM Differential.
- Capture: Captures a snapshot of the current timer count when timer input edge transitions.
- Compare: Timer pin toggles when timer exceeds terminal count.
- Gated: Timer increments only when timer input pin is asserted.
- Capture/Compare: Timer counts when timer input is asserted, captures timer count when input is deasserted.

For detailed information on I²C bus operation, refer to Maxim Application Note 4024 “SPI/I²C Bus Lines Control Multiple Peripherals” at <https://www.maximintegrated.com/en/app-notes/index.mvp/id/4024>

18.1 Instances

Instances of the peripheral are listed in *Table 18-1: MAX78000 TMR/LPTMR Instances*. Both the TMR and LPTMR are functionally very similar, so for convenience they are referred to as just TMR. The LPTMR instances can function while the device is in certain low power modes.

Refer to the relevant data sheet for frequency limitations for external clock sources.

Table 18-1: MAX78000 TMR/LPTMR Instances

| Instance | Register Access Name | Cascade 32-Bit Mode | 16-Bit Mode | Operating Modes | CLK0 | CLK1 | CLK2 | CLK3 |
|----------|----------------------|---------------------|-------------|------------------------|------|----------|-------|------------------------------|
| TMRO | TMRO | Yes | Dual | ACTIVE SLEEP LPM | PCLK | ISO | IBRO | ERTCO |
| TMR1 | TMR1 | | | | | | | |
| TMR2 | TMR2 | | | | | | | |
| TMR3 | TMR3 | | | | | | | |
| LPTMRO | TMR4 | No | Single | ACTIVE SLEEP LPM | IBRO | ERTCO | INRO | LPTMRO_CLK* GPIO2.6 (AF1) |
| | | | | UPM | N/A | N/A | ERTCO | INRO |
| LPTMR1 | TMR5 | No | Single | ACTIVE SLEEP LPM | IBRO | IBRO / 8 | INRO | LPTMR1_CLK* GPIO2.7 (AF1) |
| | | | | μPM | N/A | N/A | ERTCO | INRO |

*Refer to data sheet for maximum frequency on external timer clock input pins.

Table 18-2: MAX78000 I/O Signal Configuration

| Instance | Register Access Name | TMROn_IOA | TMROn_IOB | TMROn_IOAN | TMROn_IOBN |
|----------|----------------------|--------------------------------------|--------------------------------------|--------------------|--------------------|
| TMRO | TMRO | TMROA_IOA (P0.2) TMROB_IOA (P0.8) | TMROA_IOB (P0.3) TMROB_IOB (P0.9) | TMROB_IOAN (P0.4) | TMROB_IOBN (P0.5) |
| TMR1 | TMR1 | TMR1A_IOA (P0.14) | TMR1A_IOB (P0.15) | TMR1B_IOAN (P0.12) | TMR1B_IOBN (P0.13) |
| TMR2 | TMR2 | TMR2A_IOA (P0.26) | TMR2A_IOB (P0.27) | - | - |
| TMR3 | TMR3 | TMR3B_IOA (P1.4) TMR3B_IOA (P1.6) | TMR3A_IOB (P1.5) TMR3A_IOB (P1.7) | - | - |
| LPTMRO | TMR4 | LPTMROB_IOA (P2.4) | - | - | - |
| LPTMR1 | TMR5 | LPTMR1_IOA (P2.5) | - | - | - |

18.2 Basic Timer Operation

The timer modes operate by incrementing the *TMROn_CNT.count* register, driven by either the timer clock, an external stimulus on the timer pin, or a combination of both. The *TMROn_CNT.count* register is always readable, even while the timer is enabled and counting.

Each timer mode has a user-configurable timer period, which terminates on the timer clock cycle following the end of timer period condition. Each timer mode has a different response at the end of a timer period, which can include changing the state of the timer pin, capturing a timer value, reloading *TMROn_CNT.count* with a new starting value, or disabling the counter. The end of a timer period will always set the corresponding interrupt bit and can generate an interrupt, if enabled.

In most modes the timer peripheral automatically sets *TMROn_CNT.count* to 0x0000 0001 at the end of a timer period, but *TMROn_CNT.count* is set to 0x0000_0000 following a system reset. This means the first timer period following a system reset will be one timer clock longer than subsequent timer periods if *TMROn_CNT.count* is not initialized to 0x0000 0001 during the timer configuration step.

18.3 32-Bit Single / 32-Bit Cascade / Dual 16-Bit

Most instances contain two 16-bit timers, which may support combinations of single or cascaded 32-bit modes, and single or dual 16-bit modes as shown in [Table 18-1: MAX78000 TMR/LPTMR Instances](#). In most cases the two 16-bit timers have the same functionality.

The terminology TimerA and TimerB are used to differentiate the organization of the 32-bit registers shown in [Table 18-3: TimerA/TimerB 32-Bit Field Allocations](#). Most of the other registers have the same fields duplicated in the upper and lower 16-bits are differentiated with the _a and _b suffixes. When the same functionality is supported by both 16-bit timers, the examples are shown more generically with the _a/_b omitted from the field names.

In the 32-bit modes, the fields and controls associated with TimerA are used to control the 32-bit functions.

When in dual 16-bit mode it is allowed to use TimerB without using TimerA.

In single 16-bit mode any fields associated with TimerB are ignored.

Table 18-3: TimerA/TimerB 32-Bit Field Allocations

| Register | Cascade 32-Bit Mode | DUAL 16-BIT MODE | | SINGLE 16-BIT MODE |
|---------------|---|---|--|---|
| Timer Counter | TimerA Count = <i>TMRn_CNT.count[31:0]</i> | TimerA Compare = <i>TMRn_CNT.compare[15:0]</i> | TimerB Count = <i>TMRn_CNT.count[31:16]</i> | TimerA Compare = <i>TMRn_CNT.compare[15:0]</i> |
| Timer Compare | TimerA Compare = <i>TMRn_CMP.compare[31:0]</i> | TimerA Compare = <i>TMRn_CMP.compare[15:0]</i> | TimerB Compare = <i>TMRn_CMP.compare[31:16]</i> | TimerA Compare = <i>TMRn_CMP.compare[15:0]</i> |
| Timer PWM | TimerA Count = <i>TMRn_PWM.pwm[31:0]</i> | TimerA Count = <i>TMRn_PWM.pwm[15:0]</i> | TimerB Count = <i>TMRn_PWM.pwm[31:16]</i> | TimerA Count = <i>TMRn_PWM.pwm[15:0]</i> |

18.4 Timer Clock Sources

Clocking of timer functions is driven by the timer clock frequency, f_{CNT_CLK} , which is a function of the selected clock source shown in [Table 18-1: MAX78000 TMR/LPTMR Instances](#). Most modes support multiple clock sources and prescaler values, which can be chosen independently for TimerA and TimerB when the peripheral is operating in dual 16-bit mode. The prescaler can be set from 1 to 4096 using the *TMRn_CNT.pres* field.

Equation 18-1: Timer Peripheral Clock Equation

$$f_{CNT_CLK} = \frac{f_{CLK_SOURCE}}{\text{prescaler}}$$

Software writes to the timer registers, and external events on timer pins will be asynchronous events to the slower timer clock frequency. These events are latched on the next rising edge of the timer clock. Since it is not possible to observe the timer clock directly, input events may have up to 0.5 timer clock delay before being recognized.

Use this procedure to change the timer peripheral clock source.

1. Disable the timer peripheral
 - a. Clear *TMRn_CTRL0.en* = 0 to initiate the disable.
 - b. Wait until hardware clears *TMRn_CTRL1.clken* = 0 to confirm the timer peripheral is disabled.
2. Disable the timer clock source
 - a. Clear *TMRn_CTRL0.clken* = 0 to initiate the disable.
 - b. Wait until hardware clears *TMRn_CTRL1.clkrdy* to 0 to confirm the timer clock source is disabled.
3. Set *TMRn_CTRL1.clksel* to the new desired clock source.
4. Enable the timer clock source
 - a. Set *TMRn_CTRL0.clken* = 1 to initiate the enable.
 - b. Wait until hardware sets *TMRn_CTRL1.clkrdy* to 1 to confirm the timer clock source is enabled.
5. Enable the timer peripheral
 - a. Set *TMRn_CTRL0.en* = 1 to initiate the enable.
 - b. Wait until hardware sets *TMRn_CNT.clken* = 1 to confirm the timer peripheral is enabled.

The timer peripheral should be disabled while changing any of the registers in the peripheral. If changing the clock source is part of the configuration process, it may be preferable to execute steps 1-4, configure the timer peripheral, and lastly enable the timer peripheral again as described in step 5.

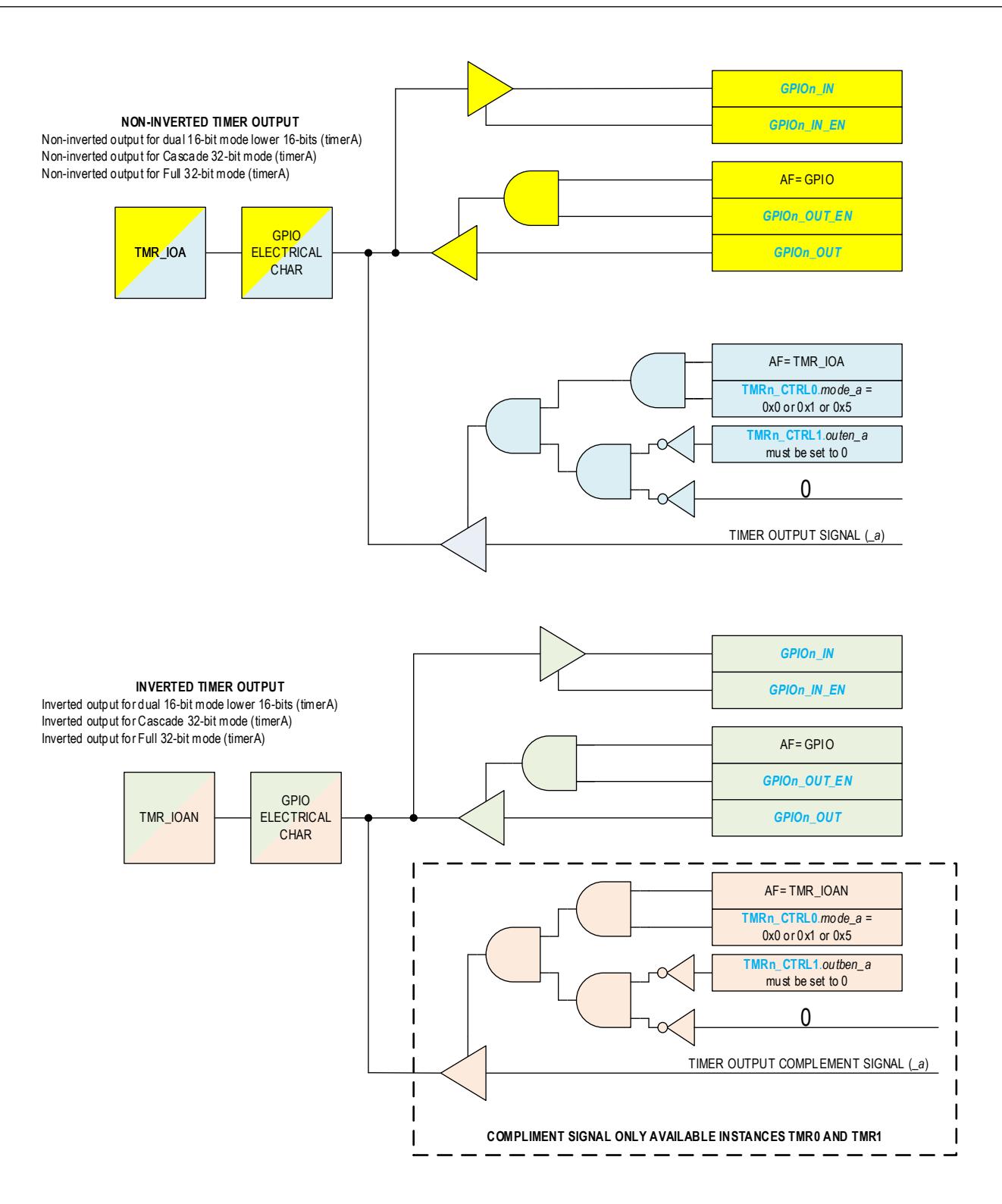
18.5 Timer Pin Functionality

Each timer instance can have an input signal and/or output signal depending on the operating mode. Depending on the specific device and package, the input and output signals may be on different device pins or may be multiplexed on a single bi-directional pin. Timer pin assignments are detailed in the data sheet for the specific device.

The timer pin functionality is mapped as an alternate function that is shared with a GPIO. When the timer pin alternate function is enabled, the timer pin will have the same electrical characteristics, such as pullup/pulldown strength, drive strength, etc. as the GPIO mode settings for that pin. The pin characteristics must be configured before enabling the timer. When configured as an output, the corresponding bit in the GPIO_OUT register should be configured to match the inactive state of the timer pin for that mode. Consult the GPIO section for details on how to configure the electrical characteristics for the pin.

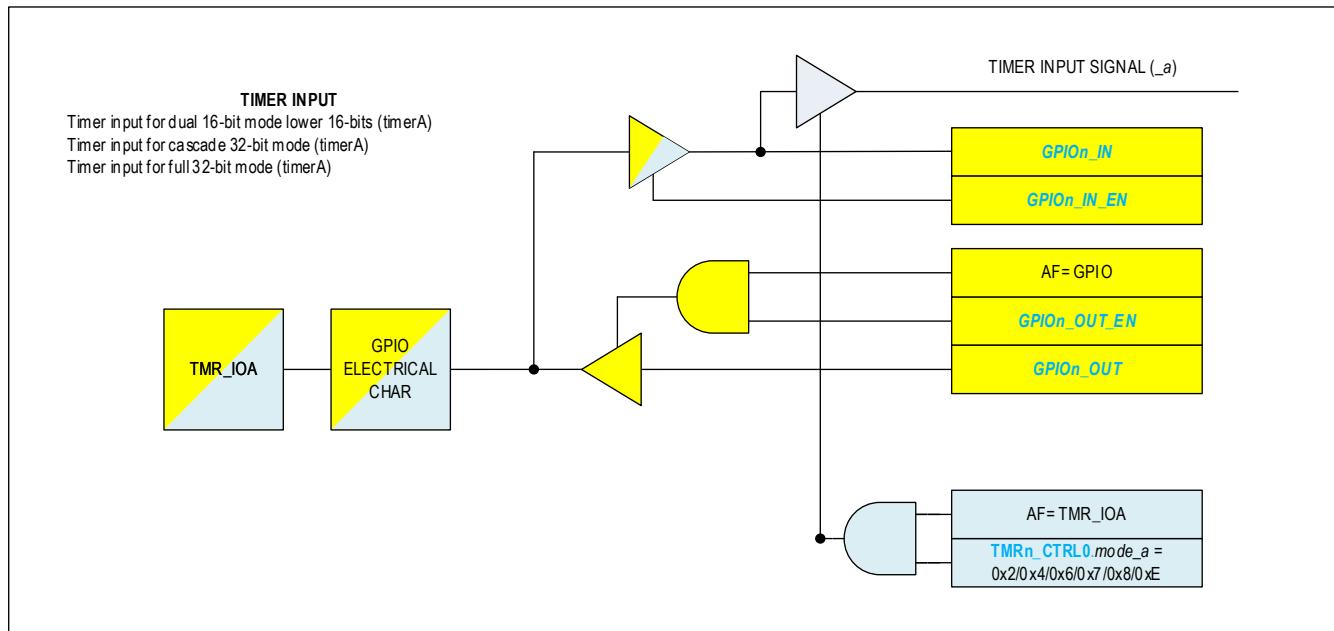
The I/O control for timer output signals is shown in [Figure 18-1: MAX78000 TimerA Output Functionality, Modes 0/1/3/5](#). The I/O control for timer input signals is shown in [Figure 18-2: MAX78000 TimerA Input Functionality, Modes 2/4/6/7/8](#).

Figure 18-1: MAX78000 TimerA Output Functionality, Modes 0/1/3/5



Preliminary Draft 08/31/2020

Figure 18-2: MAX78000 TimerA Input Functionality, Modes 2/4/6/7/8/14



18.6 Wakeup Events

In low-power modes, the system clock may be turned off to conserve power. LPTMR instances can continue to run from the clock sources shown [Table 18-1: MAX78000 TMR/LPTMR Instances](#). In this case, a wake-up event can be configured to wake up the clock control logic and re-enable the system clock.

Table 18-4: MAX78000 Wakeup Events

| Condition | Local Wakeup Flag <i>TMRe_INTFL</i> | Local Wakeup Enable | Low Power Peripheral Wakeup Flag | Low Power Peripheral Wakeup Enable | Power Management Wakeup Enable |
|----------------------|--|---------------------|----------------------------------|------------------------------------|--------------------------------|
| Any event for LPTMR0 | <i>irq_a</i> | N/A | <i>PWRSEQ_LPPWST.tmr4</i> | <i>PWRSEQ_.tmr4</i> | N/A |
| Any event for LPTMR1 | <i>irq_a</i> | N/A | <i>PWRSEQ_LPPWST.tmr5</i> | <i>PWRSEQ_.tmr5</i> | N/A |

18.7 PWM Operation

Synchronous and differential PWM modes are supported, described below respectively.

18.7.1 PWM Synchronous Mode

The peripheral is in PWM synchronous mode if *TMRe_CTRL0.pwmsync* = 1 and *TMRe_CTRL0.tmode* is configured for PWM. In the dual timer mode, we can configure the timers to either 2 master, 2 slaves or “1 master and 1 slave”.

The reload signals of the timer counter and PWM counter are coming from the pins which can be connected to the other timer circuitry which are synchronous to each other. The periods of the two timers are the same, and the duty cycle can be configured by the *TMRe_NOLCMP* register.

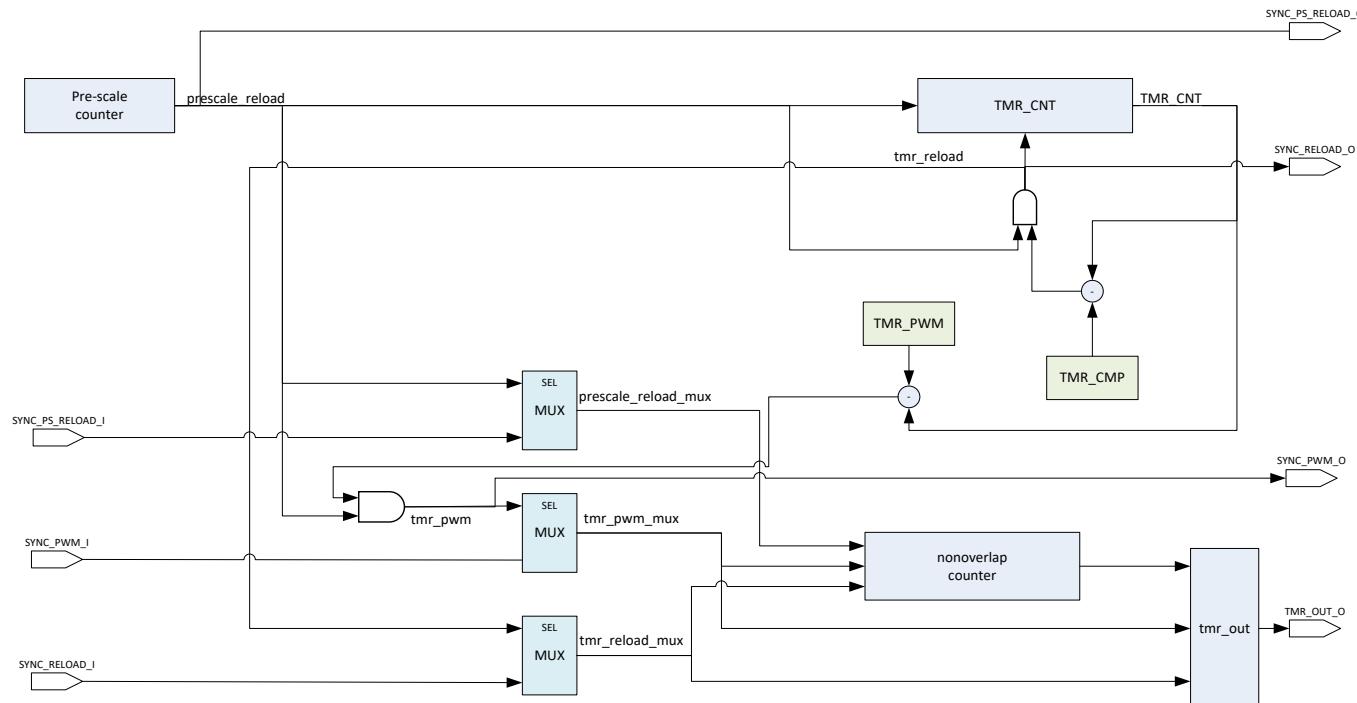
The timer is in master mode when *TMRe_CTRL0.pwmsync* = 0. The PWM operations are based on the setting from register. Three control signals (*prescale_reload_mux*, *tmr_pwm_mux*, *tmr_reload_mux*) control the behavior of nonoverlap counters and PWM output. The SYNC input signals are don’t care in master mode.

The timer is in slave mode when *TMRn_CTRL0.pwmsync* =1. The PWM operations are based on the SYNC input pins. Three control signals (*prescale_reload_mux*, *tmr_pwm_mux*, *tmr_reload_mux*) are coming from SYNC input pins, so that the behavior of nonoverlap counters and PWM output are controlled by another timer which can be in master mode or slave mode. Pre-scale counter, *TMRn_CNT.count* counter and SYNC output signals are controlled by the configuration of the registers, so are the behavior of interrupt and wake up functions.

The SYNC related signals are active for one CLK_TMR period and no asynchronous interfaces between them, that mean the master and slave should work on the same clock domain.

In low-power modes, the system clock may be turned off to conserve power. In this case, a wake-up event can be configured to wake up the clock control logic and re-enable the system clock. The Wakeup conditions are the same as the interrupts.

Figure 18-3: PWM Synchronous Mode Circuit Diagram



18.7.2 PWM Differential Mode

In Differential Mode, an additional PWM output $\emptyset A'$ (phase A prime) is available. $\emptyset A'$ functions as a differential or complementary output to the primary PWM output ($\emptyset A$). By default, $\emptyset A'$ will behave as an exact inverse. For nonoverlapping control of power transistors and other complementary switching applications, an 8-bit Non-Overlapping High and Low counter is featured to configure the differential output phases. This counter adjusts the two “dead time” phases from PWM output $\emptyset A'$ falling edge to $\emptyset A$ rising edge using the Non-Overlapping High Compare register (NOLHCM) and vice-versa for the Non-Overlapping Low Compare register (NOLLCM). This scheme provides independent control of these two dead phases. In PWM mode, each output has an additional independent polarity control bit *TMRn_CTRL0.nolhpol* and *TMRn_CTRL0.nollhpol* for additional waveform configuration which can result in some redundancies with the primary polarity bit *TMRn_CTRL0.pol*. *Figure 18-4: Timer Output Complementary Waveforms* shows the effect these three fields on the output waveforms.

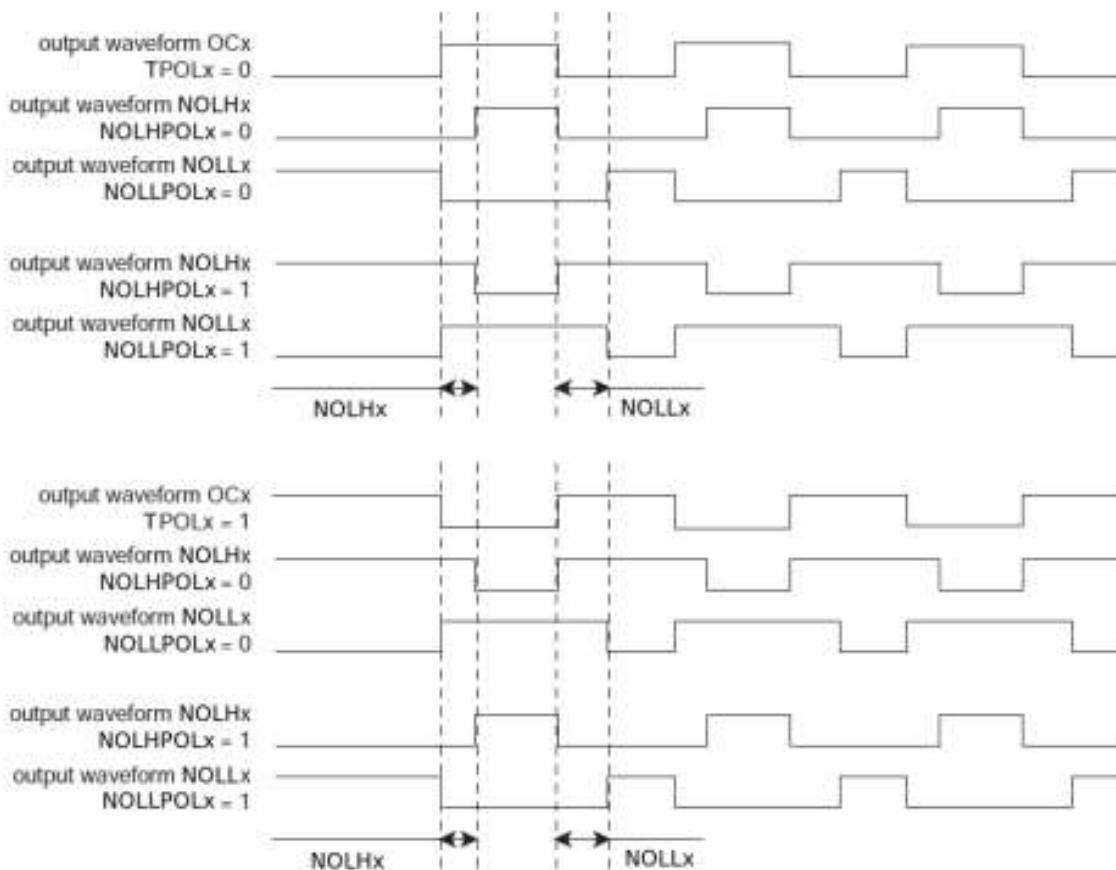
Finally, a timer PWM Synchronization Mode bit is available *TMRn_CTRL0.pwmsync* that multiplexes a primary timer’s main counter to other secondary (or slave) timers. This attribute will synchronize the timers’ periods, primary polarity polarities and duty-cycles to fix a base PWM waveform depicted as OCx in *Figure 18-4: Timer Output Complementary Waveforms* below. However, the Non-Overlapping registers are independent and configurable for each of the synchronized timers,

regardless of the `TMRn_CTRL0.pwmsync` bit. This provides independent adjustments to several sets of complementary switches.

The device is set up to have two sets of complementary outputs.

- For TimerA configured in PWM Differential Mode with `TMRn_CTRL0.pwmsync =1`, $\phi A = \text{TCLK}0$, $\phi A' = \text{TCLK}1$.
- For TimerB configured in PWM Differential Mode with `TMRn_CTRL0.pwmsync =1`, $\phi A = \text{TCLK}2$, $\phi A' = \text{TCLK}3$.

Figure 18-4: Timer Output Complementary Waveforms



18.8 Operating Modes

Multiple operating modes are supported. The availability of some operating modes is dependent on the device and package-specific implementation of the external input and output signals. The settings for *TMRn_CTRL1.outen* and *TMRn_CTRL1.outben* indirectly affect both input and output signals. The values of those two fields must be set exactly as shown in the Operating Mode Signals tables.

Table 18-5: MAX78000 Operating Mode Signals, TMR0/TMR1

| Timer Mode | TMR0 | TMR1 | Signal | Required? |
|---------------------|---|---|-----------|-----------|
| One-Shot Mode (0) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Optional |
| | TimerA Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOAN | Optional |
| | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Optional |
| | TimerB Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOBN | Optional |
| Continuous Mode (1) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Optional |
| | TimerA Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOAN | Optional |
| | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Optional |
| | TimerB Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOBN | Optional |
| Counter Mode (2) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Yes |
| Capture Mode (4) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Yes |
| Compare Mode (5) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Optional |

| Timer Mode | TMR0 | TMR1 | Signal | Required? |
|-----------------------------------|---|---|------------------------|-----------|
| | TimerA Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOAN | Optional |
| | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOB | Optional |
| | TimerB Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Complementary Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOBN | Optional |
| <i>Gated Mode (6)</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOB | Yes |
| <i>Capture/Compare Mode (7)</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOB | Yes |
| <i>Dual Edge Capture Mode (8)</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOB | Yes |
| Reserved (0x9) | | | | |
| Reserved (0xA) | | | | |
| Reserved (0xB) | | | | |
| Reserved (0xC) | | | | |
| Reserved (0xD) | | | | |
| <i>Inactive Gated Mode (14)</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TM _{Rn} _IOB | Yes |
| Reserved (0xF) | | | | |

Table 18-6: MAX78000 Operating Mode Signals, TMR2/TMR3

| Timer Mode | TMR2 | TMR3 | Signal | Required? |
|----------------------------|--|--|----------|-----------|
| One-Shot Mode (0) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Optional |
| | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Optional |
| Continuous Mode (1) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Optional |
| | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Optional |
| Counter Mode (2) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Yes |
| Capture Mode (4) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0l</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0l</i> | TMRn_IOB | Yes |
| Compare Mode (5) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0l</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0l</i> | TMRn_IOA | Optional |
| | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Optional |
| Gated Mode (6) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Yes |
| Capture/Compare Mode (7) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Yes |
| Dual Edge Capture Mode (8) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Yes |
| Reserved (0x9) | | | | |
| Reserved (0xA) | | | | |

| Timer Mode | TMR2 | TMR3 | Signal | Required? |
|---------------------------------|--|--|----------|-----------|
| Reserved (0xB) | | | | |
| Reserved (0xC) | | | | |
| Reserved (0xD) | | | | |
| <i>Inactive Gated Mode (14)</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerB Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOB | Yes |
| Reserved (0xF) | | | | |

Table 18-7: MAX78000 Operating Mode Signals, LPTMR0/LPTMR1

| TIMER MODE | LPTMR0 | LPTMR1 | SIGNAL | REQUIRED? |
|----------------------------|---|---|----------|-----------|
| One-Shot Mode (0) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Optional |
| Continuous Mode (1) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Optional |
| Counter Mode (2) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| Capture Mode (4) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| Compare Mode (5) | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Output Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Optional |
| Gated Mode (6) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| Capture/Compare Mode (7) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| Dual Edge Capture Mode (8) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| Reserved (0x9) | | | | |
| Reserved (0xA) | | | | |
| Reserved (0xB) | | | | |
| Reserved (0xC) | | | | |
| Reserved (0xD) | | | | |
| Inactive Gated Mode (14) | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TimerA Input Signal <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i> | TMRn_IOA | Yes |
| Reserved (0xF) | | | | |

18.8.1 One-Shot Mode (0)

In One-shot mode the timer peripheral increments *TMRn_CNT.count* until it matches *TMRn_CMP.compare* and then stops incrementing and disables the timer. The timer optionally pulses the timer output signal to its active state for one clock cycle. In this mode, the timer must be re-enabled to start another one-shot mode event.

The timer period ends on the timer clock following *TMRn_CNT.count = TMRn_CMP.compare*.

The timer peripheral automatically performs the following actions at the end of the timer period:

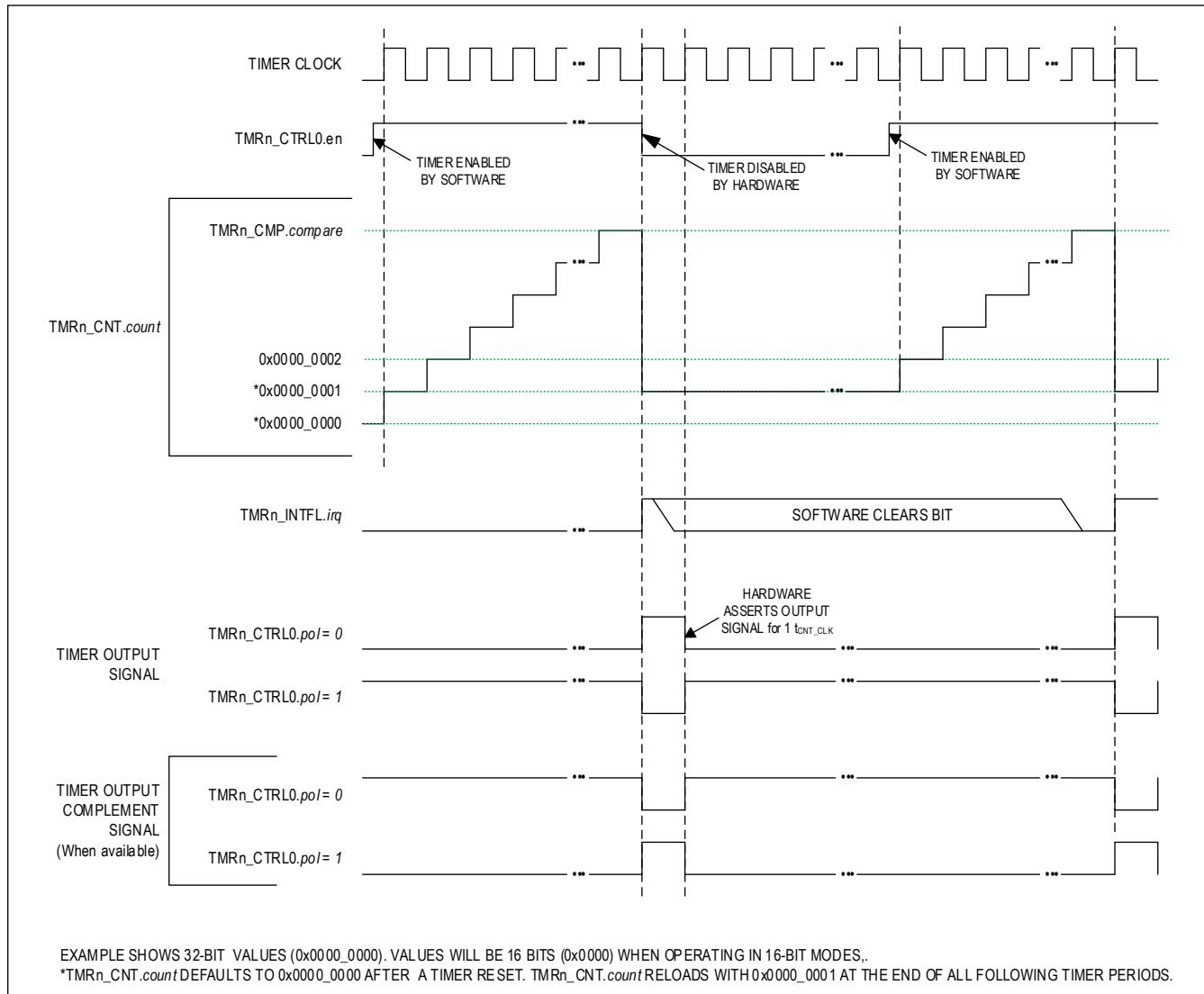
- Hardware resets *TMRn_CNT.count* to 0x0000 0001.
- Hardware disables the timer by clearing *TMRn_CTRL0.en = 0*.
- If the timer output is enabled, the timer pin is driven to its active state for one timer clock. It then returns to its inactive state.
- The corresponding *TMRn_INTFL.irq* field will be set to 1 to indicate a timer interrupt event occurred.

The timer period is calculated using *Equation 18-2: One-shot Mode Timer Period*.

Equation 18-2: One-shot Mode Timer Period

$$\text{One-shot mode timer period in seconds} = \frac{\text{TMR_CMP} - \text{TMRn_CNT}_{\text{INITIAL_VALUE}} + 1}{f_{\text{CNT_CLK}} (\text{Hz})}$$

Figure 18-5: One-Shot Mode Diagram



Configure the timer for One-Shot mode by doing the following:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set `TMRn_CTRL0.mode` to 0x0 to select One-shot mode.
4. Set `TMRn_CTRL0.pres` to set the prescaler that determines the timer frequency.
5. If using the timer output function:
 - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set `TMRn_CTRL1.outen` and `TMRn_CTRL1.outben` to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the timer output pin.
6. If using the inverted timer output function:
 - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set `TMRn_CTRL1.outen` and `TMRn_CTRL1.outben` to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the inverted timer output pin.
7. If using the timer interrupt, enable corresponding field in the `TMRn_CTRL1` register.
8. Write the compare value to `TMRn_CMP.compare`.
9. If desired, write an initial value to `TMRn_CNT.count`. This effects only the first period; subsequent timer periods always reset `TMRn_CNT.count` = 0x0000 0001.
10. Enable the timer peripheral as described in [Timer Clock Sources](#).

18.8.2 Continuous Mode (1)

In Continuous mode, the timer peripheral increments `TMRn_CNT.count` until it matches `TMRn_CMP.compare`, resets `TMRn_CNT.count` to 0x0000_0001, and continues incrementing. The timer optionally toggles the state of the timer output signal at the end of the timer period.

The timer period ends on the timer clock following `TMRn_CNT.count` = `TMRn_CMP.compare`.

The timer peripheral automatically performs the following actions at the end of the timer period:

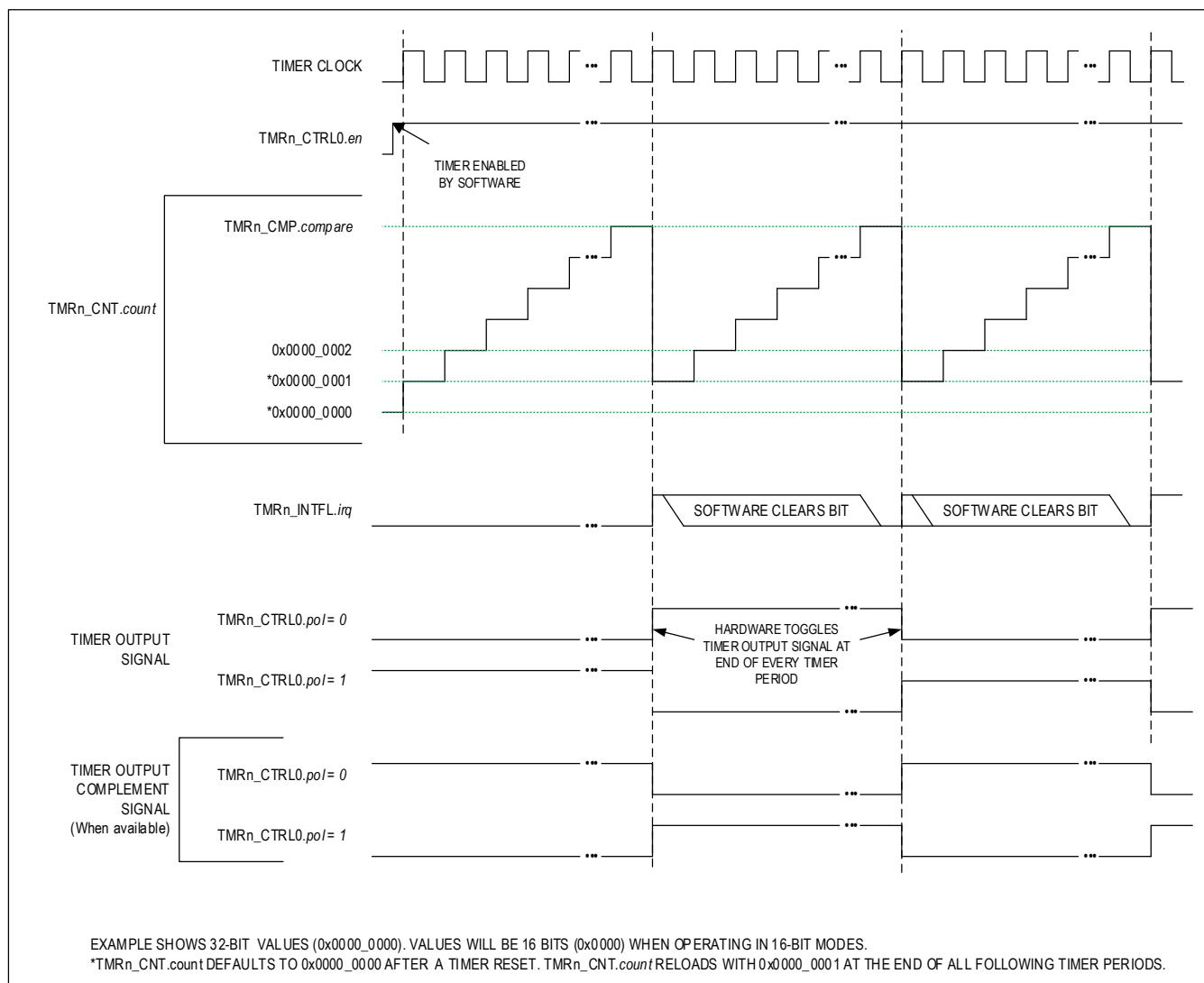
- Hardware resets `TMRn_CNT.count` to 0x0000_0001. The timer remains enabled and continues incrementing.
- Hardware disables the timer by clearing `TMRn_CTRL0.en` = 0.
- If the timer output signal is toggled.
- The corresponding `TMRn_INTFL.irq` field will be set to 1 to indicate a timer interrupt event occurred.

The Continuous Mode Timer Period is calculated using [Equation 18-3: Continuous Mode Timer Period](#).

Equation 18-3: Continuous Mode Timer Period

$$\text{Continuous mode timer period in seconds} = \frac{\text{TMRn_CMP} - \text{TMRn_CNT}_{\text{INITIAL_VALUE}} + 1}{f_{\text{CNT_CLK}} \text{ (Hz)}}$$

Figure 18-6: Continuous Mode Diagram



Configure the timer for Continuous mode by performing the steps following:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set `TMRn_CTRL0.mode` to 0x1 to select Continuous mode.
4. Set `TMRn_CTRL0.pres` to set the prescaler that determines the timer frequency.
5. If using the timer output function:
 - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set `TMRn_CTRL1.outen` and `TMRn_CTRL1.outben` to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the timer output pin.
6. If using the inverted timer output function:
 - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set `TMRn_CTRL1.outen` and `TMRn_CTRL1.outben` to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the inverted timer output pin.
7. If using the timer interrupt, enable corresponding field in the `TMRn_CTRL1` register.
8. Write the compare value to `TMRn_CMP.compare`.
9. If desired, write an initial value to `TMRn_CNT.count`. This effects only the first period; subsequent timer periods always reset `TMRn_CNT.count` = 0x0000 0001.
10. Enable the timer peripheral as described in [Timer Clock Sources](#).

18.8.3 Counter Mode (2)

In Counter mode, the timer peripheral increments `TMRn_CNT.count` each time when a transition occurs on the timer input signal. When `TMRn_CNT.count` = `TMRn_CMP.compare` the interrupt bit is set and `TMRn_CNT.count` is set to 0x0000 0001 and continues incrementing. The timer can be configured to increment on either the rising edge or the falling edge, but not both.

The timer prescaler setting has no effect in this mode. The frequency of the timer's input signal (f_{CTR_CLK}) must not exceed 25 percent of the PCLK frequency as shown [Equation 18-4](#).

Equation 18-4: Counter Mode Maximum Clock Frequency

$$f_{CTR_CLK} \leq \frac{f_{PCLK} (\text{Hz})}{4}$$

The timer period ends on the rising edge of PCLK following `TMRn_CNT.count` = `TMRn_CMP.compare`.

The timer peripheral automatically performs the following actions at the end of the timer period:

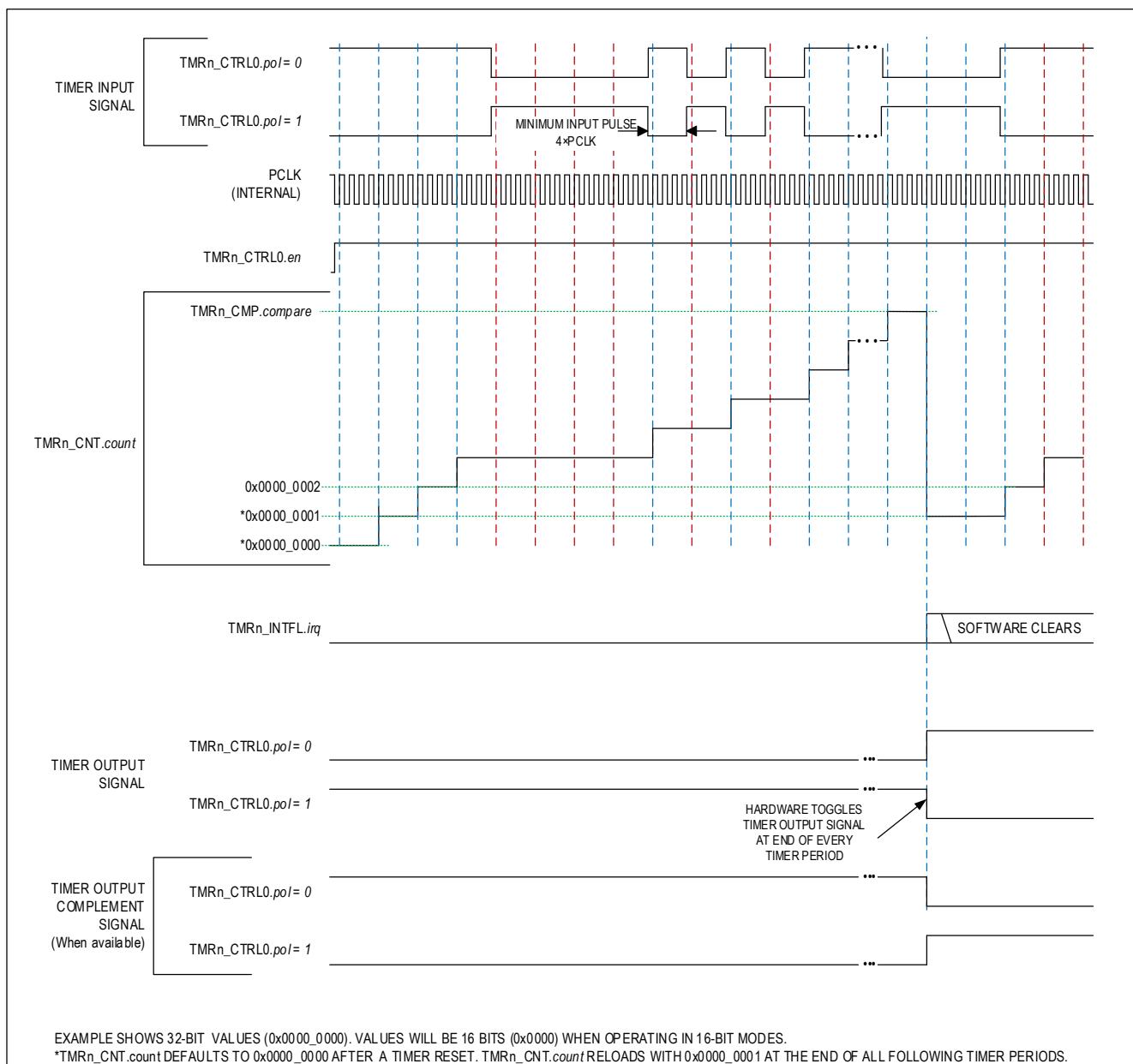
- Hardware resets `TMRn_CNT.count` to 0x0000_0001. The timer remains enabled and continues incrementing.
- Hardware toggles the state of the timer output signal. The timer output pin will change state if the timer output is enabled.
- Hardware sets the corresponding `TMRn_INFL.irq` field to 1 to indicate a timer interrupt event occurred.

In Counter mode, the number of timer input transitions since timer start is calculated using [Equation 18-5](#).

Equation 18-5: Counter Mode Timer Input Transitions

$$\text{Counter mode timer input transitions} = TMRn_CNT_{CURRENT_VALUE} - TMRn_CNT_{INITIAL_VALUE}$$

Figure 18-7: Counter Mode Diagram



Configure the timer for Counter mode by doing the following:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set `TMRn_CTRL0.mode` 0x2 to select Counter mode.
4. Configure the timer input function:
 - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set `TMRn_CTRL1.outen` and `TMRn_CTRL1.outben` to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the timer input pin.
5. Write the compare value to `TMRn_CMP.compare`.
6. If desired, write an initial value to `TMRn_CNT.count`. This effects only the first period; subsequent timer periods always reset `TMRn_CNT.count` = 0x0000 0001.
7. Enable the timer peripheral as described in [Timer Clock Sources](#).

18.8.4 PWM Mode (3)

In PWM mode, the timer sends a Pulse-Width Modulated (PWM) output using the timer's output signal. The timer first counts up to the match value stored in the `TMRn_PWM.pwm` register. At the end of the cycle where the `TMRn_CNT.count` value matches the `TMRn_PWM.pwm`, the timer output signal toggles state. The timer continues counting until it reaches the `TMRn_CMP.compare` value.

The timer period ends on the rising edge of f_{CNT_CLK} following `TMRn_CNT.count = TMRn_CMP.compare`.

The timer peripheral automatically performs the following actions at the end of the timer period:

- The `TMRn_CNT.count` is reset to 0x0000_0001, and the timer resumes counting.
- The timer output signal is toggled.
- The corresponding `TMRn_INTFL.irq` field will be set to 1 to indicate a timer interrupt event occurred.

When `TMRn_CTRL0.pol` = 0, the timer output signal starts low and then transitions to high when the `TMRn_CNT.count` value matches the `TMRn_PWM` value. The timer output signal remains high until the `TMRn_CNT.count` value reaches the `TMRn_CMP.compare`, resulting in the timer output signal transitioning low, and the `TMRn_CNT.count` value resetting to 0x0000 0001.

When `TMRn_CTRL0.pol` = 1, the Timer output signal starts high and transitions low when the `TMRn_CNT.count` value matches the `TMRn_PWM` value. The timer output signal remains low until the `TMRn_CNT.count` value reaches `TMRn_CMP.compare`, resulting in the timer output signal transitioning high, and the `TMRn_CNT.count` value resetting to 0x0000 0001.

Complete the following steps to configure a timer for PWM mode and initiate the PWM operation:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set *TMRn_CTRL0.mode* to 0x3 to select PWM mode.
4. Set *TMRn_CTRL0.pres* to set the prescaler that determines the timer frequency.
5. Configure the pin as a timer input and configure the electrical characteristics as needed.
6. Set *TMRn_CTRL0.pol* to match the desired initial (inactive) state.
7. Set *TMRn_CTRL0.pol* to select the initial logic level (high or low) and PWM transition state for the timer's output.
8. Set *TMRn_CNT.count* to the starting count, typically 0x0000_0001. The initial *TMRn_CNT.count* value only effects the initial period in PWM mode with subsequent periods always setting *TMRn_CNT.count* to 0x0000_0001.
9. Set the *TMRn_PWM* value to the transition period count.
10. Set the *TMRn_CMP.compare* value for the PWM second transition period. Note: *TMRn_CMP.compare* must be greater than the *TMRn_PWM* value.
11. If using the timer interrupt, set the interrupt priority and enable the interrupt.
12. Enable the timer peripheral as described in [Timer Clock Sources](#).

The PWM period is calculated using the following equation:

Equation 18-6: Timer PWM Period

$$\text{PWM period in seconds} = \frac{\text{TMRn_CNT}}{f_{\text{CNT_CLK}} (\text{Hz})}$$

If an initial starting value other than 0x0000_0001 is loaded into the *TMRn_CNT.count* register, use the One-Shot mode equation to determine the initial PWM period.

If *TMRn_CTRL0.pol* is 0, the ratio of the PWM output high time to the total period is calculated using [Equation 18-7, below](#).

Equation 18-7: Timer PWM Output High Time Ratio with Polarity 0

$$\text{PWM output high time ratio (\%)} = \frac{(\text{TMR_CMP} - \text{TMR_PWM})}{\text{TMR_CMP}} \times 100$$

If *TMRn_CTRL0.pol* is set to 1, the ratio of the PWM output high time to the total period is calculated using [Equation 18-8](#).

Equation 18-8: Timer PWM Output High Time Ratio with Polarity 1

$$\text{PWM output high time ratio (\%)} = \frac{\text{TMR_PWM}}{\text{TMR_CMP}} \times 100$$

18.8.5 Capture Mode (4)

The Capture mode is most often used to measure the time between user-determined events. The timer increments from an initial value until an event occurs. The event can be the external transition of the timer input signal or a software-generated pseudo-event that captures the current value of *TMRn_CNT.count* and writes it to *TMRn_PWM.pwm*.

A rollover event is a special timer period event which serves as a “timeout” feature for the timer. Write the maximum period for a valid capture event to *TMRn_CMP.compare*. If the selected capture event does not occur before *TMRn_CNT.count* = *TMRn_CMP.compare* a rollover timer period event will occur.

All timer period events will reset *TMRn_CNT.count* = 0x0000_0001 and set the corresponding timer interrupt event flag.

Software should read *TMRn_PWM.pwm* to determine the event that set timer interrupt event flag. *TMRn_PWM.pwm* = 0 indicates a rollover timer period event occurred. Any other value indicates a capture timer period event occurred.

Three events are possible in this mode

18.8.5.1 Capture Event (Timer Input Signal)

A Capture event occurs on the timer clock following the user-specified positive or negative transition of the external input signal. This triggers a ‘capture’ timer period event which copies *TMRn_CNT.count* to the *TMRn_PWM.pwm* register.

The timer peripheral automatically performs the following actions when a timer input capture event occurs:

- Hardware copies the value in *TMRn_CNT.count* to *TMRn_PWM.pwm*.
- The timer remains enabled and continues incrementing.
- The corresponding *TMRn_INTL.irq* field will be set to 1 to indicate a timer interrupt event occurred.

18.8.5.2 Capture Event (Software Capture)

A software capture event occurs when software generates a pseudo-transition on the timer input signal.

18.8.5.3 Rollover Event

The Rollover event occurs when *TMRn_CNT.count* = *TMRn_CMP.compare*, indicating that a capture event did not occur with the desired timer period. The timer peripheral automatically performs the following actions when a rollover event occurs:

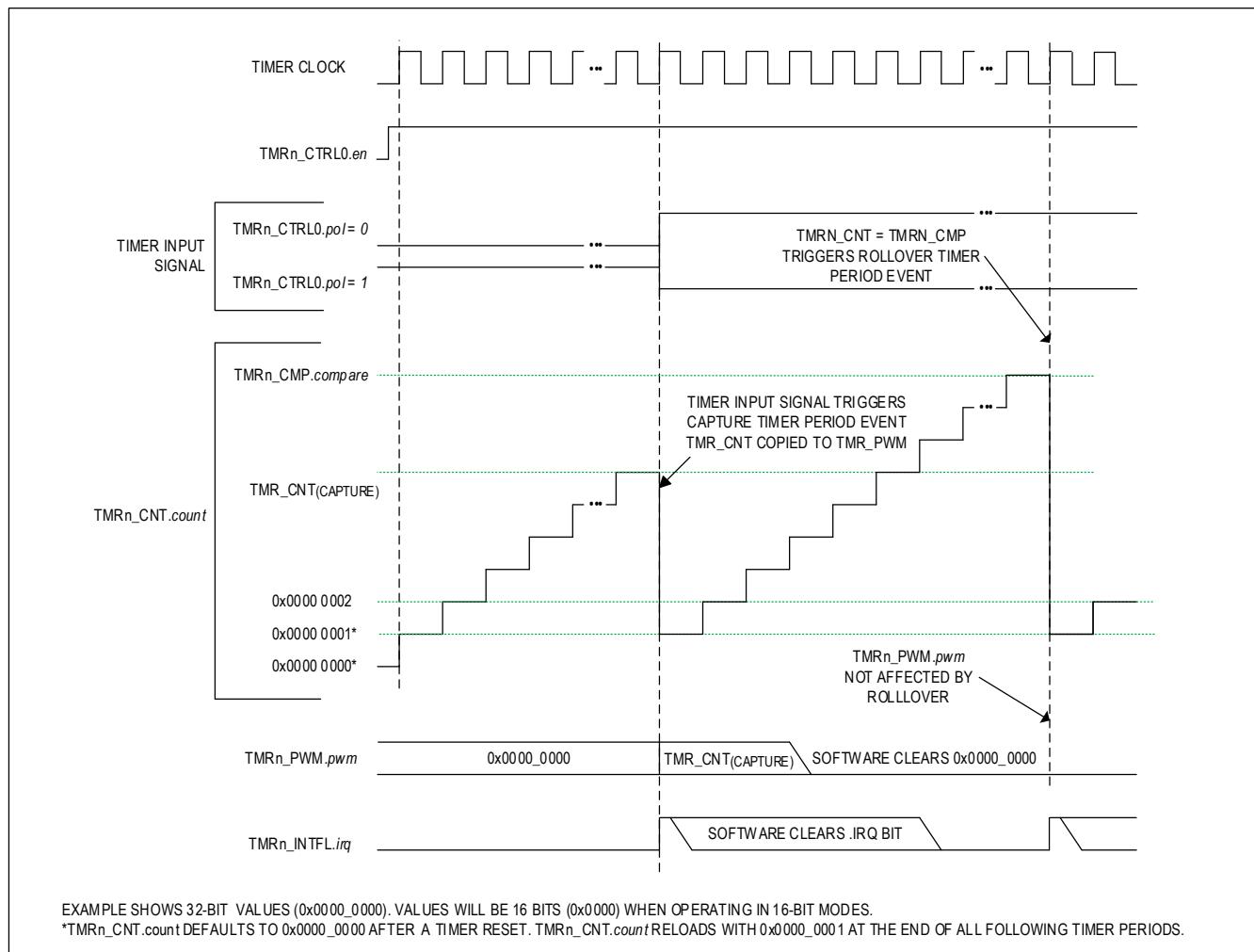
- Hardware resets *TMRn_CNT.count* to 0x0000 0001. The timer remains enabled and continues incrementing. The value in *TMRn_PWM.pwm* remains unchanged.
- Hardware sets the corresponding *TMRn_INTL.irq* field to 1 to indicate a timer interrupt event occurred.

Equation 18-9: Capture Mode Elapsed Time Calculation in Seconds

$$\text{Capture elapsed time in seconds} = \frac{\text{TMR_PWM} - \text{TMR_CNT}_{\text{INITIAL_VALUE}}}{f_{\text{CNT_CLK}}}$$

*Note: The capture elapsed time calculation is only valid after the capture event occurs, and the timer stores the captured count in the *TMRn_PWM* register.*

Figure 18-8: Capture Mode Diagram



Configure the timer for Capture mode by doing the following:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set [`TMRn_CTRL0.mode`](#) to 0x4 to select Capture mode.
4. Configure the timer input function:
 - a. Set [`TMRn_CTRL0.pol`](#) to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set [`TMRn_CTRL1.outen`](#) and [`TMRn_CTRL1.outben`](#) to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the timer input pin.
5. Write the initial value to [`TMRn_CNT.count`](#). This effects only the first period; subsequent periods always begin with 0x0000_0001.
6. Write the compare value to [`TMRn_CMP.compare`](#).
7. Select the capture event by setting [`TMRn_CTRL1.capecventselsel`](#).
8. Enable the timer peripheral as described in [Timer Clock Sources](#).

The timer period is calculated using the following equation:

Equation 18-10: Capture Mode Elapsed Time Calculation in Seconds

$$\text{Capture elapsed time in seconds} = \frac{\text{TMR_PWM} - \text{TMR_CNT}_{\text{INITIAL_VALUE}}}{f_{\text{CNT_CLK}}}$$

Note: The capture elapsed time calculation is only valid after the capture event occurs, and the timer stores the captured count in the *TMRn_PWM* register.

18.8.6 Compare Mode (5)

In Compare mode the timer peripheral increments continually from 0x0000_0000 (after the first timer period) to the maximum value of the 32- or 16-bit mode, then rolls over to 0x0000_0000 and continues incrementing. The end of timer period event occurs when the timer value matches the compare value, but the timer continues to increment until the count reaches 0xFFFF FFFF. The timer counter then rolls over and continues counting from 0x0000_0000.

The timer period ends on the timer clock following *TMRn_CNT.count* = *TMRn_CMP.compare*.

The timer peripheral automatically performs the following actions when a timer period event:

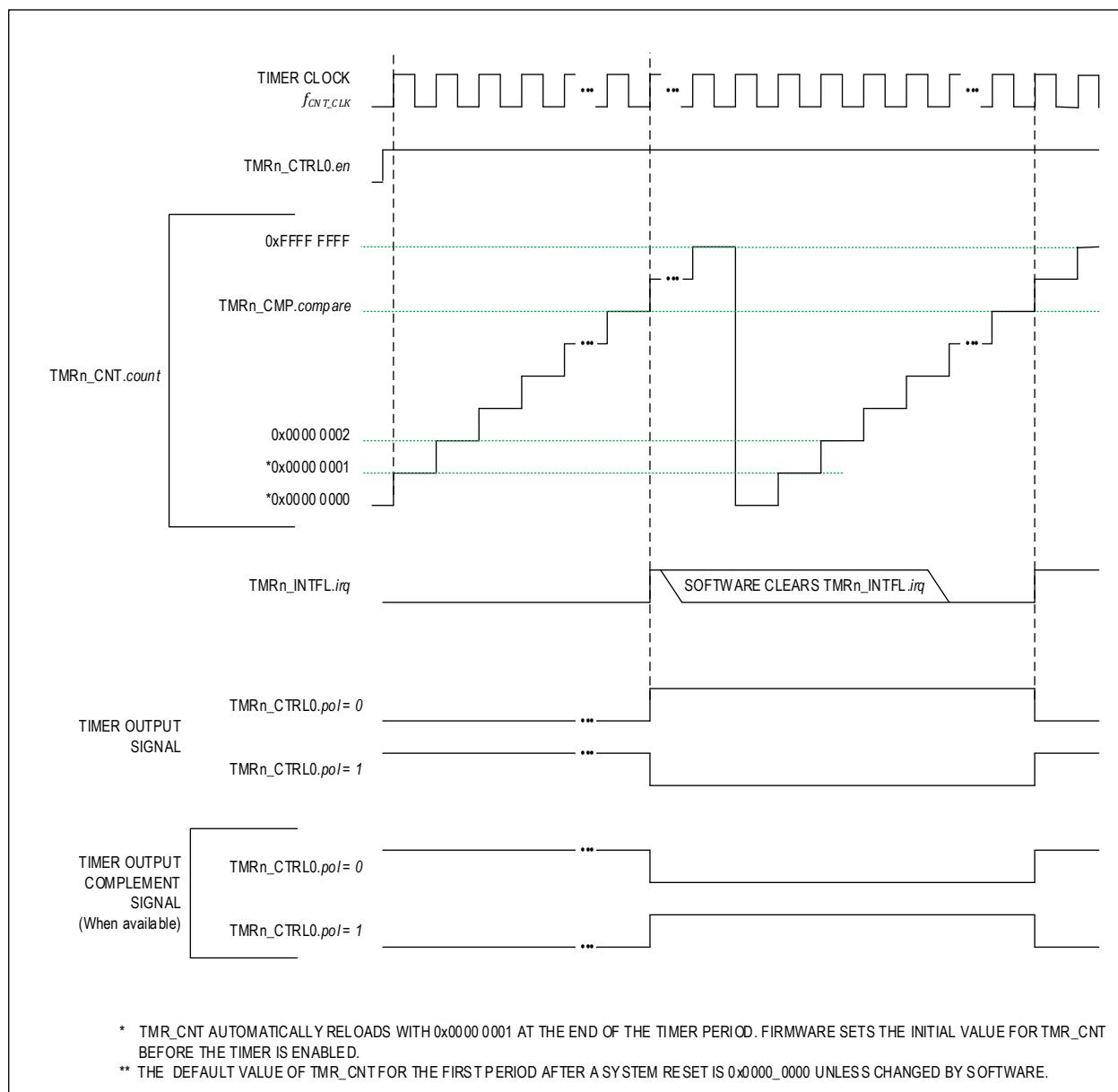
- Unlike other modes, *TMRn_CNT.count* is reset to 0x0000_0000 not 0x0000_0001 at the end of the timer period. The timer remains enabled and continues incrementing.
- The corresponding *TMRn_INTFL.irq* field will be set to 1 to indicate a timer interrupt event occurred.
- Hardware toggles the state of the timer output signal. The timer output pin will change state if the timer output is enabled.

The Compare Mode timer period is calculated using *Equation 18-12: Capture Mode Elapsed Time*.

Equation 18-11: Compare Mode Timer Period

$$\text{Compare mode timer period in second} = \frac{(\text{TMR_CMP} - \text{TMR_CNT}_{\text{INITIAL_VALUE}} + 1)}{f_{\text{CNT_CLK}}(\text{Hz})}$$

Figure 18-9: Compare Mode Diagram



Configure the timer for Compare mode by doing the following:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set `TMRn_CTRL0.mode` to 0x5 to select Compare mode.
4. Set `TMRn_CTRL0.pres` to set the prescaler that determines the timer frequency.
5. If using the timer output function:
 - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set `TMRn_CTRL1.outen` and `TMRn_CTRL1.outben` to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the timer output pin.
6. If using the inverted timer output function:
 - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set `TMRn_CTRL1.outen` and `TMRn_CTRL1.outben` to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the inverted timer output pin.
7. If using the timer interrupt, enable corresponding field in the `TMRn_CTRL1` register.
8. Write the compare value to `TMRn_CMP.compare`.
9. If desired, write an initial value to `TMRn_CNT.count`. This effects only the first period; subsequent timer periods always reset `TMRn_CNT.count` = 0x0000 0001.
10. Enable the timer peripheral as described in [Timer Clock Sources](#).

18.8.7 Gated Mode (6)

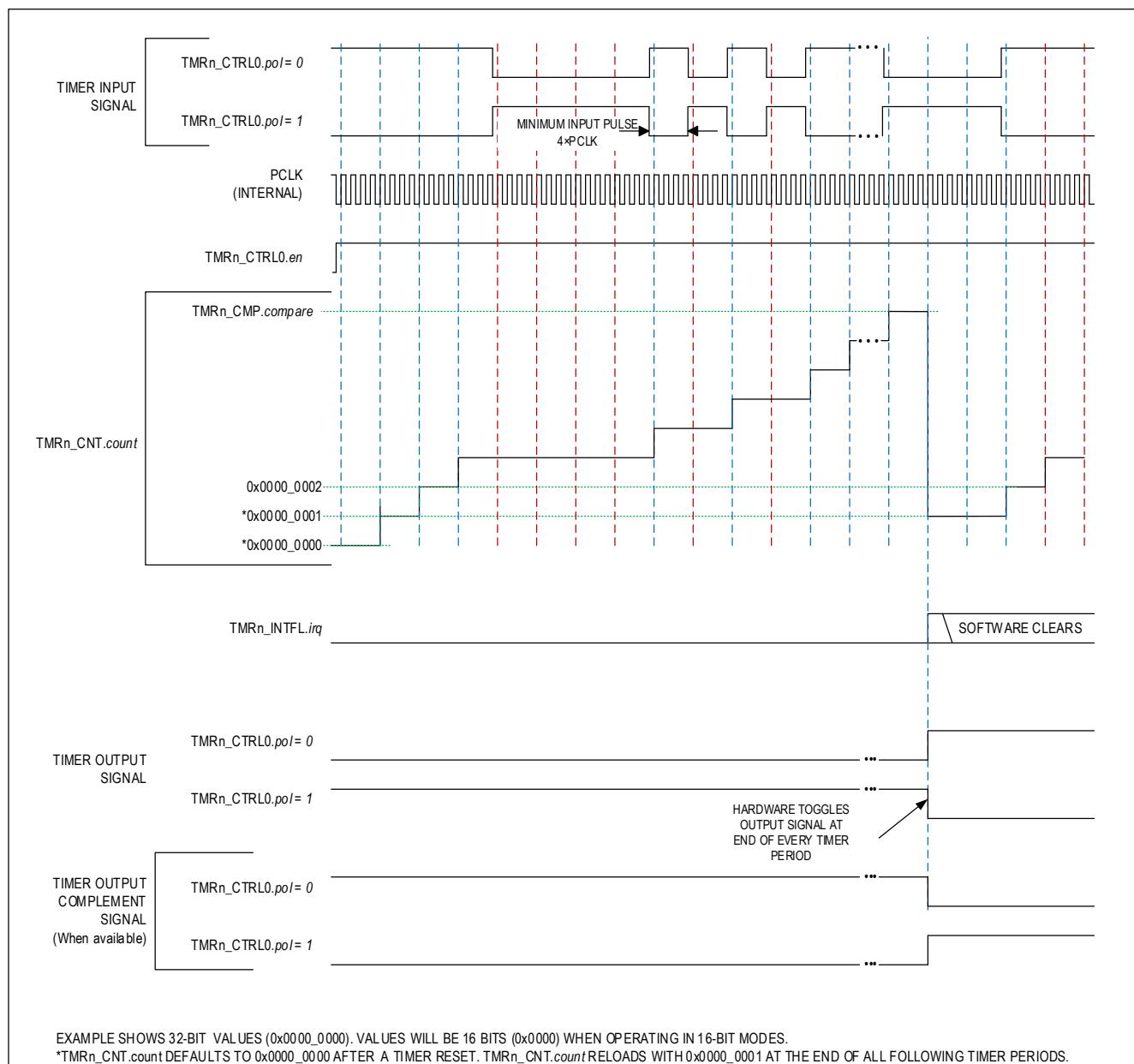
Gated mode is similar to continuous mode, except that `TMRn_CNT.count` only increments when the timer input signal is in its active state.

The timer period ends on the timer clock following `TMRn_CNT.count` = `TMRn_CMP.compare`.

The timer peripheral automatically performs the following actions at the end of the timer period:

- Hardware resets `TMRn_CNT.count` to 0x0000 0001. The timer remains enabled and continues incrementing.
- If the timer output signal toggles state. The timer output pin will change state if the timer output is enabled.
- The corresponding `TMRn_INTFL.irq` field will be set to 1 to indicate a timer interrupt event occurred.

Figure 18-10: Gated Mode Diagram



Configure the timer for Gated mode by doing the following:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set `TMRn_CTRL0.mode` to 0x6 to select Gated mode.
4. Configure the timer input function:
 - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set `TMRn_CTRL1.outen` and `TMRn_CTRL1.outben` to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the timer input pin.
5. Write the initial value to `TMRn_CNT.count`. This effects only the first period; subsequent periods always begin with 0x0000_0001.
6. Write the compare value to `TMRn_CMP.compare`.
7. Enable the timer peripheral as described in [Timer Clock Sources](#).

18.8.8 Capture/Compare Mode (7)

In Capture/Compare mode, the timer starts counting after the first external timer input transition occurs. The transition, a rising edge or falling edge on the timer's input signal, is set using the `TMRn_CTRL0.pol` bit.

Each subsequent transition, after the first transition of the timer input signal, captures the `TMRn_CNT.count` value, writing it to the `TMRn_PWM.pwm` register (capture event). When a capture event occurs, a timer interrupt is generated, the `TMRn_CNT.count` value is reset to 0x0000_0001, and the timer resumes counting.

If no capture event occurs, the timer counts up to `TMRn_CMP.compare`. At the end of the cycle where the `TMRn_CNT.count` equals the `TMRn_CMP.compare`, a timer interrupt is generated, the `TMRn_CNT.count` value is reset to 0x0000 0001, and the timer resumes counting.

The timer period ends when the selected transition occurs on the timer pin, or on the clock cycle following $TMRn_CNT.count = TMRn_CMP.compare$.

The actions performed at the end of the timer period are dependent on the event that ended the timer period:

If the end of the timer period was caused by a transition on the timer pin:

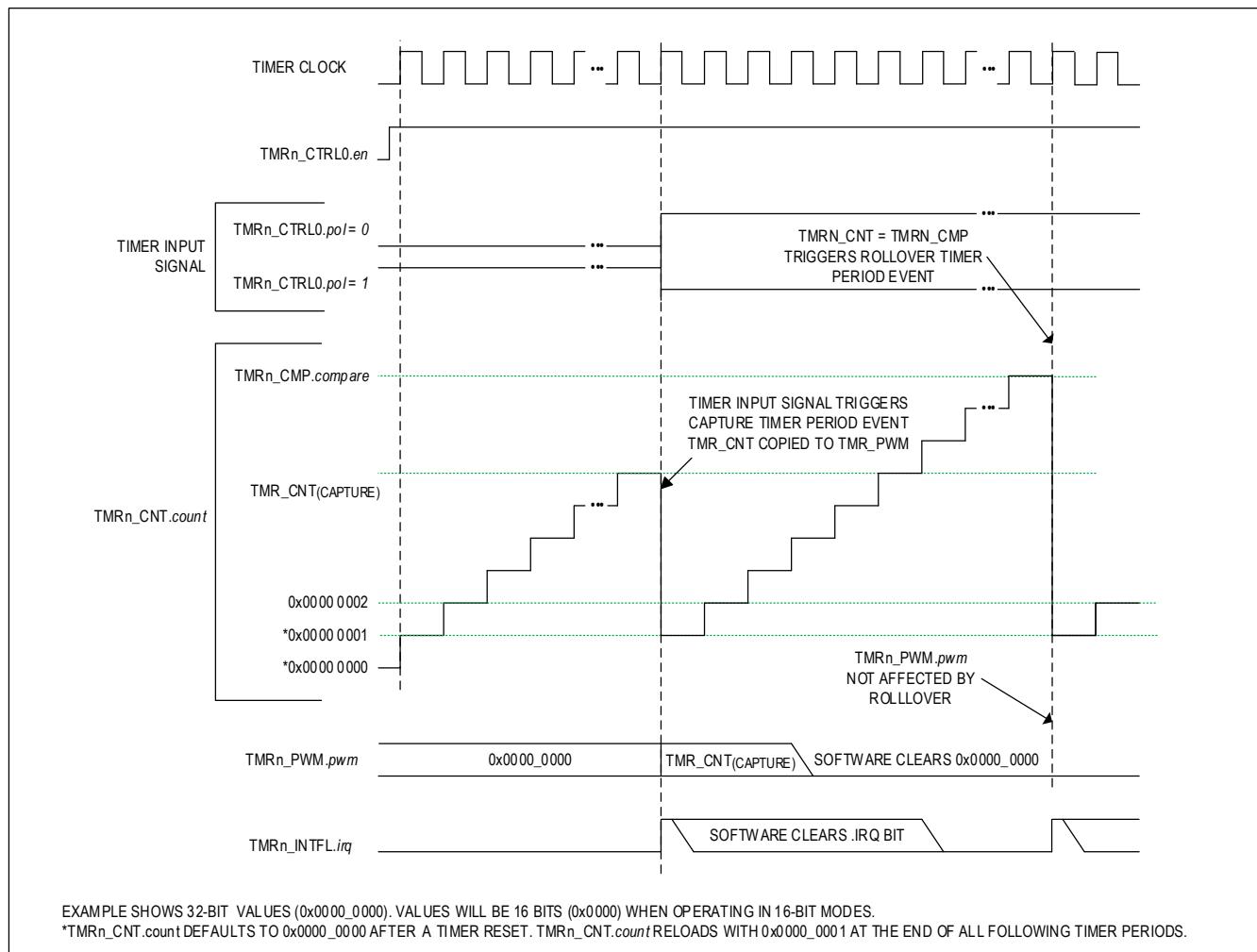
- Hardware copies the value in `TMRn_CNT.count` to `TMRn_PWM.pwm`.
- Hardware resets `TMRn_CNT.count` to 0x0000 0001. The timer remains enabled and continues incrementing.
- The corresponding `TMRn_INTFL.irq` field will be set to 1 to indicate a timer interrupt event occurred.

In Capture/Compare mode, the elapsed time from the timer start to the capture event is calculated using [Equation 18-12, below](#).

Equation 18-12: Capture Mode Elapsed Time

$$\text{Capture elapsed time in seconds} = \frac{\text{TMR_PWM} - \text{TMRn_CNT}_{\text{INITIAL_CNT_VALUE}}}{f_{\text{CNT_CLK}} \text{ (Hz)}}$$

Figure 18-11: Capture/Compare Mode Diagram



Configure the timer for Capture/Compare mode by doing the following:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set [`TMn_CTRL0.mode`](#) to 7 to select Capture/Compare mode.
4. Configure the timer input function:
 - a. Set [`TMn_CTRL0.pol`](#) to select the positive edge ([`TMn_CTRL0.pol = 1`](#)) or negative edge ([`TMn_CTRL0.pol = 0`](#)) transition causes the capture event..
 - b. Configure the GPIO electrical characteristics as desired.
 - c. Set [`TMn_CTRL1.outen`](#) and [`TMn_CTRL1.outben`](#) to the values shown in the [Operating Modes](#) section.
 - d. Select the correct alternate function mode for the timer input pin.
5. Write the initial value to [`TMn_CNT.count`](#). This effects only the first period; subsequent periods always begin with 0x0000 0001.
6. Write the compare value to [`TMn_CMP.compare`](#).
7. Enable the timer peripheral as described in [Timer Clock Sources](#).

Note: No interrupt is generated by the first transition of the input signal.

18.8.9 Dual Edge Capture Mode (8)

Dual edge capture mode is like capture mode except the counter can capture on both edges of the timer input pin.

18.8.10 Inactive Gated Mode (14)

Inactive gated mode is like gated mode except that the interrupt will be triggered when the timer input pin is in its inactive state.

18.9 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 18-8: Timer Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 18-8: Timer Register Summary

| Offset | Register | Description |
|----------|-----------------------------|--|
| [0x0000] | TMRn_CNT | Timer Counter Register |
| [0x0004] | TMRn_CMP | Timer Compare Register |
| [0x0008] | TMRn_PWM | Timer PWM Register |
| [0x000C] | TMRn_INTFL | Timer Interrupt Register |
| [0x0010] | TMRn_CTRLO | Timer Control Register |
| [0x0014] | TMRn_NOLCMP | Timer Non-Overlapping Compare Register |
| [0x0018] | TMRn_CTRL1 | Timer Configuration Register |
| [0x001C] | TMRn_WKFL | Timer Wakeup Status Register |

18.9.1 Register Details

Table 18-9: Timer Count Register

| Timer Count | | TMRn_CNT | | | [0x0000] |
|-------------|-------|--------------------------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | count | R/W | 0 | Timer Count This field increments at a rate dependent on the selected timer operating mode. The function of the bits in this field are dependent on the 32-bit/16-bit configuration. Reads of this register always return the current value. Writing to this field results in hardware clearing the TMRn_INTFL.wrdone to 0. Firmware must wait until the TMRn_INTFL.wrdone field reads 1 before proceeding. Disable the timer by setting TMRn_CTRL1.wrdis to 1 before writing to this field. | |

Table 18-10: Timer Compare Register

| Timer Compare | | TMRn_CMP | | | 0x0004 |
|---------------|---------|--------------------------|-------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | compare | R/W | 0 | Timer Compare Value The value in this register is used as the compare value for the timer's count value. The compare field meaning is determined by the specific mode of the timer. See the timer mode's detailed configuration section for compare usage and meaning. | |

Table 18-11: Timer PWM Register

| Timer PWM | | | | TMRn_PWM | 0x0004 |
|-----------|---------|--------|-------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | compare | R/W | 0 | Timer PWM Match In PWM mode, this field sets the count value for the first transition period of the PWM cycle. At the end of the cycle when <i>TMRn_CNT.count</i> = <i>TMRn_CMP.compare</i> , the PWM output transitions to the second period of the PWM cycle. The second PWM period count is stored in <i>TMRn_CMP.compare</i> . <i>TMRn_PWM.pwm</i> must be less than <i>TMRn_CMP.compare</i> for PWM mode operation. Timer Capture Value In Capture, Compare, and Capture/Compare modes, this field is used to store the <i>TMRn_CNT.count</i> value when a Capture, Compare, or Capture/Compare event occurs. Hardware will automatically clear the corresponding <i>TMRn_INTFL.wrdone</i> field to 0 when this field is written, and software must wait until <i>TMRn_INTFL.wrdone</i> = 1 before proceeding. Disable the timer by setting <i>TMRn_CTRL1.wrdis</i> to 1 before writing to this field. | |

Table 18-12: Timer Interrupt Register

| Timer Interrupt | | | | TMRn_INTFL | 0x000C |
|-----------------|----------|--------|-------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 31:26 | - | RO | 0 | Reserved Do not modify | |
| 24 | wr_dis_b | R/W | 0 | TimerB Write Protect in Dual Timer Mode Set this field to 0 to write protect the TimerB fields in the <i>TMRn_CNT.count[31:16]</i> and <i>TMRn_PWM.pwm[31:16]</i> . When this field is set to 0, 32-bit writes to the <i>TMRn_CNT</i> and <i>TMRn_PWM</i> registers only modify the lower 16-bits associated with TimerA. <i>Note: This field always reads 0 if the timer is configured as a 32-bit cascade timer.</i> 0: Enabled 1: Disabled | |
| 25 | wrdone_b | R | 0 | TimerB Write Done This field is cleared to 0 by hardware when software performs a write to <i>TMRn_CNT.count[31:16]</i> or <i>TMRn_PWM.pwm[31:16]</i> when in dual timer mode. Wait until the field is set to 1 again before proceeding. 0: Operation in progress. 1: Operation complete. | |
| 23:17 | - | RO | 0 | Reserved Do not modify | |
| 16 | irq_b | R/W1C | 0 | Interrupt Event TimerB This field is set when a TimerB interrupt event occurs. Write 1 to clear. 0: No event 1: Interrupt event occurred | |
| 15:10 | - | RO | 0 | Reserved Do not modify | |

| Timer Interrupt | | | | TMRn_INTFL | 0x000C |
|-----------------|----------|--------|-------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 9 | wr_dis_a | R/W | 0 | Dual Timer Mode Write Protect TimerB This field disables write access to <i>TMRn_CNT.count[31:16]</i> and <i>TMRn_PWM.pwm[31:16]</i> so that only the 16 bits associated with updating TimerA are modified during writes to the <i>TMRn_CNT</i> and <i>TMRn_PWM</i> registers. This bit will always read 0 if the timer is configured in a 32-bit mode. 0: Enabled 1: Disabled | |
| 8 | wrdone_a | R | 0 | Write Done TimerA This field is cleared to 0 by hardware when firmware performs a write to <i>TMRn_CNT.count[31:16]</i> or <i>TMRn_PWM.pwm[31:16]</i> when in dual timer mode. Wait until the field reads 1 before proceeding. 0: Operation in progress 1: Operation complete | |
| 7:1 | - | RO | 0 | Reserved Do not modify | |
| 0 | irq_a | W1C | 0 | Interrupt Event TimerA This field is set when a TimerA interrupt event occurs. Write 1 to clear. 0: No event 1: Interrupt event occurred | |

Table 18-13: Timer Control 0 Register

| Timer Control 0 | | | | TMRn_CTRL0 | 0x0010 |
|-----------------|---------|--------|-------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 31 | en_b | R/W | 0 | Enable TimerB 0: Disabled 1: Enabled | |
| 30 | clken_b | R/W | 0 | Timer Clock Enable TimerB 0: Disabled 1: Enabled | |
| 29 | rst_b | W1 | 0 | RST TimerB 0: No action 1: Reset TimerB | |
| 28:24 | - | RO | 0 | Reserved Do not modify | |

| Timer Control 0 | | | TMRn_CTRL0 | | 0x0010 |
|-----------------|-----------|--------|------------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 23:20 | clkdivb | R/W | 0 | Timer Prescaler Select TimerB The <i>clkdivb</i> field selects a prescaler that divides the Timer Peripheral's source clock to set the Timer's Count Clock as follows: $f_{CNT_CLK} = f_{CLK_SOURCE} / \text{prescaler}$ Refer to the operational mode details to determine which modes use the prescaler. 0: 1 1: 2 2: 4 3: 8 4: 16 5: 32 6: 64 7: 128 8: 256 9: 512 10: 1024 11: 2048 12: 4096 13-15: Reserved | |
| 19:16 | mode_b | R/W | 0 | Timer Mode Select TimerB Set this field to the desired timer mode for TimerB. 0: One-Shot 1: Continuous 2: Counter 3: PWM 4: Capture 5: Compare 6: Gated 7: Capture/Compare 8: Dual-Edge Capture 9 - 11: Reserved for Future Use 12: Internally Gated 13 - 15: Reserved | |
| 15 | en_a | R/W | 0 | Enable TimerA 0: Disabled 1: Enabled | |
| 14 | clken_a | R/W | 0 | Timer Clock Enable TimerA 0: Disabled 1: Enabled | |
| 13 | rst_a | W1 | 0 | RST TimerA 0: No action 1: Reset TimerA | |
| 12 | pwmckbd_a | R/W | 1 | PWM Output ϕ_A' Disable TimerA 1: Disable PWM Output 0: Enable PWM Output ϕ_A' | |

| Timer Control 0 | | | TMRn_CTRL0 | | 0x0010 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------------|-----------|--------|------------|--|--------|---|----|---|----|---|----|---|----|----|----|----|----|----|----|-----|----|-----|----|-----|-----|------|-----|------|-----|------|--------|----------|--|
| Bits | Field | Access | Reset | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | nollpol_a | R/W | 0 | PWM Output $\phi_{A'}$ Polarity Bit TimerA 1: Output $\phi_{A'}$ inverted 0: Output $\phi_{A'}$ non-inverted | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | nolhpol_a | R/W | 0 | PWM Output ϕ_A Polarity Bit TimerA 1: Output ϕ_A inverted 0: Output ϕ_A non-inverted | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | pwmsync_a | R/W | 0 | PWM Synchronization Mode TimerA 0: Disabled 1: Enabled | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | pol_a | R/W | 0 | Timer Polarity TimerA Selects the polarity of the timer's input and output signal. This setting is not used if the GPIO is not configured for the alternate function. The <i>pol</i> field meaning is determined by the specific mode of the timer. See the mode's detailed configuration section for <i>pol</i> usage. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7:4 | clkdiva | R/W | 0 | Timer Prescaler Select TimerA The <i>clkdiva</i> field selects a prescaler that divides the Timer Peripheral's source clock to set the Timer's Count Clock as follows: $f_{CNT_CLK} = f_{CLK_SOURCE} / \text{prescaler}$ Refer to the specific Timer Operational Mode to determine which modes use the Prescaler selected. <table style="margin-left: 20px;"> <tr><td>0:</td><td>1</td></tr> <tr><td>1:</td><td>2</td></tr> <tr><td>2:</td><td>4</td></tr> <tr><td>3:</td><td>8</td></tr> <tr><td>4:</td><td>16</td></tr> <tr><td>5:</td><td>32</td></tr> <tr><td>6:</td><td>64</td></tr> <tr><td>7:</td><td>128</td></tr> <tr><td>8:</td><td>256</td></tr> <tr><td>9:</td><td>512</td></tr> <tr><td>10:</td><td>1024</td></tr> <tr><td>11:</td><td>2048</td></tr> <tr><td>12:</td><td>4096</td></tr> <tr><td>13-15:</td><td>Reserved</td></tr> </table> | 0: | 1 | 1: | 2 | 2: | 4 | 3: | 8 | 4: | 16 | 5: | 32 | 6: | 64 | 7: | 128 | 8: | 256 | 9: | 512 | 10: | 1024 | 11: | 2048 | 12: | 4096 | 13-15: | Reserved | |
| 0: | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1: | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2: | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3: | 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4: | 16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5: | 32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6: | 64 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7: | 128 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8: | 256 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9: | 512 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10: | 1024 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11: | 2048 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12: | 4096 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13-15: | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Timer Control 0 | | | TMRn_CTRL0 | | 0x0010 |
|-----------------|--------|--------|------------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 3:0 | mode_a | R/W | 0 | Timer Mode Select TimerA Set this field to the desired timer mode for TimerA. 0: One-Shot 1: Continuous 2: Counter 3: PWM 4: Capture 5: Compare 6: Gated 7: Capture/Compare 8: Dual-Edge Capture 9-11: Reserved for Future Use 12: Internally Gated 13-15: Reserved | |

Table 18-14: Timer Non-Overlapping Compare Register

| Timer Non-Overlapping Compare | | | TMRn_NOLCMP | | [0x0010] |
|-------------------------------|-------|--------|-------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:24 | hi_b | R/W | 0 | Non-Overlapping High Compare 1 [15:8] TimerB The 8-bit timer count value of non-overlapping time between the falling edge of PWM output ϕ_A' and the next rising edge of PWM output ϕ_A . | |
| 23:16 | lo_b | R/W | 0 | Non-Overlapping Low Compare 1 [7:0] TimerB The 8-bit timer count value of non-overlapping time between the falling edge of PWM output ϕ_A and next rising edge of PWM output ϕ_A' . | |
| 15:8 | hi_a | R/W | 0 | Non-Overlapping High Compare 0 [15:8] TimerA The 8-bit timer count value of non-overlapping time between the falling edge of PWM output ϕ_A' and the next rising edge of PWM output ϕ_A . | |
| 7:0 | lo_a | R/W | 0 | Non-Overlapping Low Compare 0 [7:0] TimerA The 8-bit timer count value of non-overlapping time between the falling edge of PWM output ϕ_A and next rising edge of PWM output ϕ_A' . | |

Table 18-15: Timer Control 1 Register

| Timer Control 1 | | | TMRn_CTRL1 | | 0x0010 |
|-----------------|---------|--------|------------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 31 | cascade | R/W | 0 | 32-bit Cascade Timer Enable This field is only used by instances that support the 32-bit cascade configuration. 0: Dual 16-bit timers 1: Cascade TimerB:TimerA into one 32-bit timer TimerA | |
| 30:29 | - | RO | 0 | Reserved Do not modify | |
| 28 | we_b | R/W | 0 | Timer Wakeup Function TimerB 0: Disabled 1: Enabled. | |

| Timer Control 1 | | | | TMRn_CTRL1 | 0x0010 |
|-----------------|----------------|--------|-------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 27 | sw_capevent_b | R/W | 0 | Software Capture Event TimerB The bit is just like TMR_CAP_EVENT_Ix and TMR_GATE_I in the capture related mode(capture, capture/compare, dual-edge capture). It will create positive/negative edge to create the capture event. 0: Event=0 1: Event=1 | |
| 26:25 | capevent_sel_b | R/W | 0 | Capture Event Selection TimerB 0: TMR_GATE_I 1: Cap event 0 2: Cap event 1 3: SW_CAPEVENT | |
| 24 | ie_b | R/W | 0 | IRQ Interrupt Enable TimerB 0: Disabled 1: Enabled | |
| 23 | negtrig_b | R/W | 0 | Negative Edge Trigger for Event TimerB 0: Positive-edge trigger 1: Negative-edge trigger | |
| 22:20 | event_sel_b | R/W | 0 | Event Selection TimerB 0: Event disable 1: Start event 0 2: Start event 1 3: Start event 2 4: Start event 3 5: Reserved 6: Reserved 7: Reserved | |
| 19 | clkrdy_b | R/W | 0 | TimerB Clock Ready 0: Timer not enabled or synchronization in progress 1: Timer is enabled | |
| 18 | clken_b | R/W | 0 | TimerB Clock Enable 0: Timer not enabled or synchronization in progress 1: Timer is enabled | |
| 17:16 | clksel_b | R/W | 0 | Clock Source TimerB See Table 18-1: MAX78000 TMR/LPTMR Instances for the clock sources supported by each instance. 0: Peripheral clock 1: Timer Clock 1 2: Timer Clock 2 3: Timer Clock 3 | |
| 15 | - | RO | 0 | Reserved Do not modify | |
| 14 | outben_a | R/W | 0 | OUTBEN The value of this field should be set as described in the Operating Mode Signals tables in the Operating Modes section. | |
| 13 | outen_a | R/W | 0 | OUTEN The value of this field is should be set as described in the Operating Mode Signals tables in the Operating Modes section. | |
| 12 | we_a | R/W | 0 | Timer Wakeup Function TimerA 0: Disabled 1: Enabled. | |

| Timer Control 1 | | | | TMRn_CTRL1 | 0x0010 |
|-----------------|---------------|--------|-------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 11 | sw_capevent_a | R/W | 0 | Software Capture Event TimerA The bit is just like TMR_CAP_EVENT_Ix and TMR_GATE_I in the capture related mode (capture, capture/compare, dual-edge capture). It will create positive/negative edge to create the capture event. 0: Event = 0 1: Event = 1 | |
| 10:9 | capeventsel_a | R/W | 0 | Capture Event Selection TimerA 0: TMR_GATE_I 1: Cap event 0 2: Cap event 1 3: SW_CAPEVENT | |
| 8 | ie_a | R/W | 0 | IRQ Interrupt Enable TimerA 0: Disabled 1: Enabled | |
| 7 | negtrig_a | R/W | 0 | Negative Edge Trigger for Event TimerA 0: Positive-edge trigger 1: Negative-edge trigger | |
| 6:4 | event_sel_a | R/W | 0 | Event Selection TimerA 0: Event disable 1: Start event 0 2: Start event 1 3: Start event 2 4: Start event 3 5-7: Reserved | |
| 3 | clkrdy_a | R/W | | TimerA Clock Ready 0: Timer not enabled or synchronization in progress 1: Timer is enabled | |
| 2 | clken_a | R/W | | TimerA Clock Enable 0: Timer not enabled or synchronization in progress 1: Timer is enabled | |
| 1:0 | clksel_a | R/W | | Clock Source TimerA See Table 18-1: MAX78000 TMR/LPTMR Instances for the clock sources supported by each instance. 0: Peripheral clock 1: Timer Alternate Clock 1 2: Timer Alternate Clock 2 3: Timer Alternate Clock 3 | |

Table 18-16: Timer Wakeup Status Register

| Timer Wakeup Status | | | | TMRn_WKFL | 0x001C |
|---------------------|-------|--------|-------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | RO | 0 | Reserved Do not modify | |
| 16 | b | R/W1C | 1 | Wakeup Event TimerB This flag is set when a wakeup event occurs for TimerB. Write 1 to clear. 0: No event 1: Wakeup event occurred | |
| 15:1 | - | RO | 0 | Reserved Do not modify | |

| Timer Wakeup Status | | | | TMRn_WKFL | 0x001C |
|---------------------|-------|--------|-------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 0 | a | R/W1C | 1 | Wakeup Event TimerA This flag is set when a wakeup event occurs for TimerA. Write 1 to clear. 0: No event 1: Wakeup event occurred | |

Preliminary Draft 08/31/2020

19. Wakeup Timer (WUT)

The Wakeup Timer (WUT) is a unique instance of a 32-bit timer.

1. The wakeup timer uses the is 32.768kHz clock source
2. Programmable prescaler with values from 1 to 4096
3. Supports standard 32-bit timer modes
 - a. One-Shot: Timer counts up to the terminal value then halts.
 - b. Continuous: Timer counts up to the terminal value then repeats.
4. Independent interrupt

19.1 Basic Operation

The timer modes operate by incrementing the *WUTn_CNT.count* register. The *WUTn_CNT.count* register is always readable, even while the timer is enabled and counting.

Each timer mode has a user-configurable timer period, which terminates on the timer clock cycle following the end of timer period condition. The end of a timer period will always set the corresponding interrupt bit and can generate an interrupt, if enabled.

The timer peripheral automatically sets *WUTn_CNT.count* to 0x0000 0001 at the end of a timer period, but *WUTn_CNT.count* is set to 0x0000 0000 following a system reset. This means the first timer period following a system reset will be one timer clock longer than subsequent timer periods if *WUTn_CNT.count* is not initialized to 0x0000 0001 during the timer configuration step.

The timer clock frequency, f_{CNT_CLK} , is a divided version of the 32.768kHz RTC clock. The divisor/prescaler can be set from 1 to 4096 using the *WUTn_CTRL.pres3:WUTn_CTRL.pres* as shown in *Table 19-1: MAX78000 WUT Clock Period*:

Table 19-1: MAX78000 WUT Clock Period

| pres3 | pres | Prescaler | f_{CNT_CLK} (kHz) |
|--------------|-------------|------------------|--|
| 0 | 0b000 | 1 | 32.768 |
| 0 | 0b001 | 2 | 16.384 |
| 0 | 0b010 | 4 | 8.192 |
| 0 | 0b011 | 8 | 4.096 |
| 0 | 0b100 | 16 | 2.048 |
| 0 | 0b101 | 32 | 1.024 |
| 0 | 0b110 | 64 | 0.512 |
| 0 | 0b111 | 128 | 0.256 |
| 1 | 0b000 | 256 | 0.128 |
| 1 | 0b010 | 512 | 0.064 |
| 1 | 0b011 | 1024 | 0.032 |
| 1 | 0b100 | 2048 | 0.016 |
| 1 | 0b101 | 4096 | 0.008 |
| 1 | 0b110 | Reserved | Reserved |

| pres3 | pres | Prescaler | f_{CNT_CLK} (kHz) |
|-------|-------|-----------|----------------------|
| 1 | 0b111 | Reserved | Reserved |

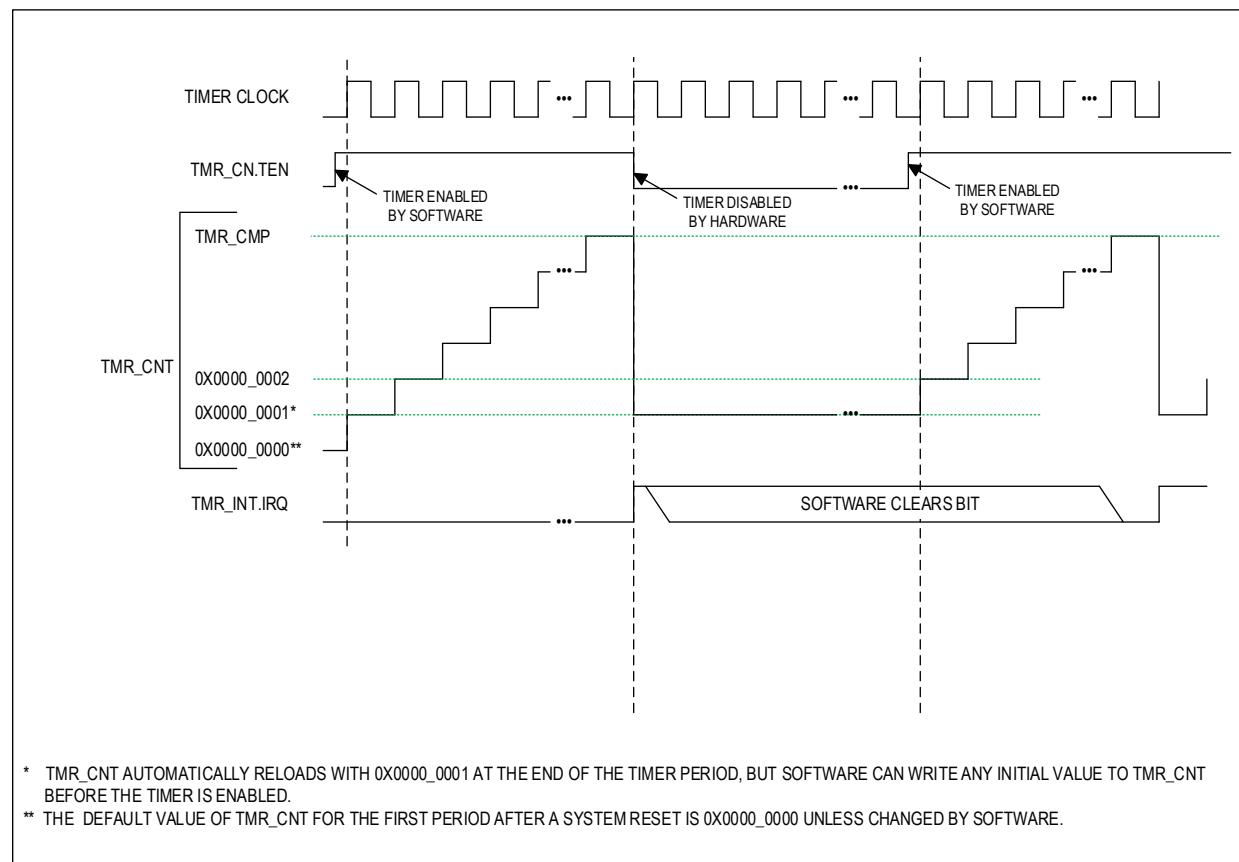
19.2 Timer Pin Functionality

The WUT does not have any timer pin functionality. Timer modes that require external inputs or outputs are not implemented on this device.

19.3 One-Shot Mode (000b)

In One-shot mode the timer peripheral increments $WUTn_CNT.count$ until it matches $WUTn_CMP$ and then stops incrementing and disables the timer. In this mode, the timer must be re-enabled to start another one-shot mode event.

Figure 19-1: One-Shot Mode Diagram



19.3.1 One-Shot Mode Timer Period

The timer period ends on the timer clock following $WUTn_CNT.count = WUTn_CMP$.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. $WUTn_CNT.count$ is reset to 0x0000 0001.
2. The timer is disabled by setting $WUTn_CTRL.ten = 0$.
3. The timer interrupt bit $WUTn_INTRirq_clr$ will be set. An interrupt will be generated if enabled.

19.3.2 One-Shot Mode Configuration

Configure the timer for One-Shot mode by doing the following:

1. Set $WUTn_CTRL.ten = 0$ to disable the timer.
2. Set $WUTn_CTRL.tmode$ to 000b to select One-shot mode.
3. Set $WUTn_CTRL.pres3:WUTn_CTRL.pres$ to determine the timer period.
4. Enable the interrupt and set the interrupt priority.
5. Write an initial value to $WUTn_CNT.count$, if desired. This effects only the first period; subsequent timer periods always reset $WUTn_CNT.count = 0x0000 0001$.
6. Write the compare value to $WUTn_CMP$.
7. Set $WUTn_CTRL.ten = 1$ to enable the timer.

The timer period is calculated using the following equation:

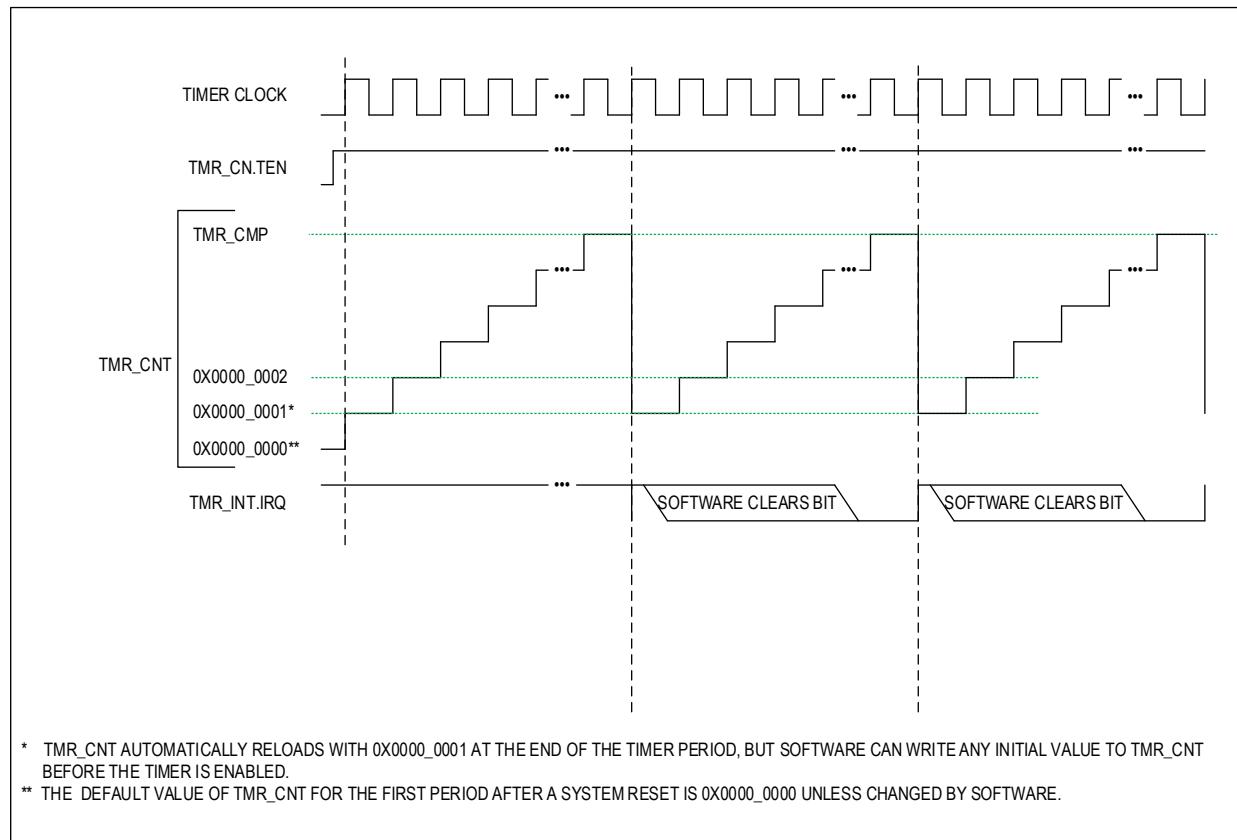
Equation 19-1: One-shot Mode Timer Period

$$\text{One-shot mode timer period in seconds} = \frac{WDTn_CMP - WDTn_CNT_{INITIAL_VALUE} + 1}{f_{CNT_CLK} (\text{Hz})}$$

19.4 Continuous Mode (001b)

In Continuous mode, the timer peripheral increments `WUTn_CNT.count` until it matches `WUTn_CMP`, resets `WUTn_CNT.count` to 0x0000 0001, and continues incrementing.

Figure 19-2: Continuous Mode Diagram



19.4.1 Continuous Mode Timer Period

The timer period ends on the timer clock following $WUTn_CNT.count = WUTn_CMP$.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. $WUTn_CNT.count$ is reset to 0x0000 0001. The timer remains enabled and continues incrementing.
2. The timer interrupt bit $WUTn_INTR.irq_clr$ will be set. An interrupt will be generated if enabled.

19.4.2 Continuous Mode Configuration

Configure the timer for Continuous mode by performing the steps following:

1. Set $WUTn_CTRL.ten = 0$ to disable the timer.
2. Set $WUTn_CTRL.tmode$ to 001b to select Continuous mode.
3. Set $WUTn_CTRL.pres3:WUTn_CTRL.pres$ to determines the timer period.
4. Enable the interrupt and set the interrupt priority.
5. Write an initial value to $WUTn_CNT.count$, if desired. The initial value is only used for the first period; subsequent timer periods always reset the $WUTn_CNT.count$ register to 1.
6. Write the compare value to $WUTn_CMP$.
7. Set $WUTn_CTRL.ten$ to 1 to enable the timer.

The Continuous Mode Timer Period is calculated using [Equation 19-2](#).

Equation 19-2: Continuous Mode Timer Period

$$\text{Continuous mode timer period in seconds} = \frac{WUTn_CMP - WUTn_CNT_{INITIAL\ VALUE} + 1}{f_{CNT_CLK} (\text{Hz})}$$

19.5 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 19-2: Wakeup Timer Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 19-2: Wakeup Timer Register Summary

| Offset | Register Name | Description |
|----------|----------------------------|---------------------------------|
| [0x0000] | WUTn_CNT | Wakeup Timer Counter Register |
| [0x0004] | WUTn_CMP | Wakeup Timer Compare Register |
| [0x000C] | WUTn_INTFL | Wakeup Timer Interrupt Register |
| [0x0010] | WUTn_CTRL | Wakeup Timer Control Register |

19.5.1 Register Details

Table 19-3: Wakeup Timer Count Register

| Wakeup Timer Count | | | WUTn_CNT | [0x0000] |
|--------------------|-------|--------|--------------------------|---|
| Bits | Name | Access | Reset | Description |
| 31:0 | count | R/W | 0 | Timer Count Value The current count value for the timer. This field increments as the timer counts. Reads of this register are always valid. Prior to writing this field, disable the timer by clearing bit WUTn_CTRL.ten . |

Table 19-4: Wakeup Timer Compare Register

| Wakeup Timer Compare | | | WUTn_CMP | [0x0004] |
|----------------------|---------|--------|--------------------------|--|
| Bits | Name | Access | Reset | Description |
| 31:0 | compare | R/W | 0 | Timer Compare Value The value in this register is used as the compare value for the timer's count value. The compare field meaning is determined by the specific mode of the timer. See the timer mode's detailed configuration section for compare usage and meaning. |

Table 19-5: Wakeup Timer Interrupt Register

| Wakeup Timer Interrupt | | | WUTn_INTFL | [0x000C] |
|------------------------|---------|--------|----------------------------|--|
| Bits | Name | Access | Reset | Description |
| 31:1 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. |
| 0 | irq_clr | RW | 0 | Timer Interrupt If set, this field indicates a timer interrupt condition occurred. Writing any value to this bit clears the timer's interrupt. 0: Timer interrupt is not active. 1: Timer interrupt occurred. |

Table 19-6: Wakeup Timer Control Register

| Wakeup Timer Control | | | WUTn_CTRL | | [0x0010] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------|-------|-----------|-----------|--|----------|------|-----------|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|----|---|-------|----|---|-------|----|---|-------|-----|---|-------|-----|---|-------|-----|---|-------|------|---|-------|------|---|-------|------|---|-------|----------|---|-------|----------|--|
| Bits | Name | Access | Reset | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31:13 | - | RO | 0 | Reserved Do not modify | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12:9 | - | DNM | 0 | Reserved Do Not Modify DNM | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | pres3 | R/W | 0 | Timer Prescaler Select MSB See <i>WUTn_CTRL.pres</i> for details on this field's usage. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | ten | R/W | 0 | Timer Enable 0: Timer disable 1: Timer enabled | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | - | DNM | 0 | Reserved Do Not Modify DNM | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5:3 | pres | R/W | 0 | Timer Prescaler Select Sets the timer's prescaler value. The prescaler divides the PCLK input to the timer and sets the timer's count clock, $f_{CNT_CLK} = \frac{f_{CLK_SOURCE}}{\text{prescaler}}$ The timer's prescaler setting is a 4-bit value with <i>pres3</i> as the most significant bit and <i>pres</i> as the three least significant bits. The table below shows the prescaler values based on <i>pres3:pres</i> . | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | <table border="1"> <thead> <tr> <th>pres3</th> <th>pres</th> <th>Prescaler</th> </tr> </thead> <tbody> <tr><td>0</td><td>0b000</td><td>1</td></tr> <tr><td>0</td><td>0b001</td><td>2</td></tr> <tr><td>0</td><td>0b010</td><td>4</td></tr> <tr><td>0</td><td>0b011</td><td>8</td></tr> <tr><td>0</td><td>0b100</td><td>16</td></tr> <tr><td>0</td><td>0b101</td><td>32</td></tr> <tr><td>0</td><td>0b110</td><td>64</td></tr> <tr><td>0</td><td>0b111</td><td>128</td></tr> <tr><td>1</td><td>0b000</td><td>256</td></tr> <tr><td>1</td><td>0b010</td><td>512</td></tr> <tr><td>1</td><td>0b011</td><td>1024</td></tr> <tr><td>1</td><td>0b100</td><td>2048</td></tr> <tr><td>1</td><td>0b101</td><td>4096</td></tr> <tr><td>1</td><td>0b110</td><td>Reserved</td></tr> <tr><td>1</td><td>0b111</td><td>Reserved</td></tr> </tbody> </table> | pres3 | pres | Prescaler | 0 | 0b000 | 1 | 0 | 0b001 | 2 | 0 | 0b010 | 4 | 0 | 0b011 | 8 | 0 | 0b100 | 16 | 0 | 0b101 | 32 | 0 | 0b110 | 64 | 0 | 0b111 | 128 | 1 | 0b000 | 256 | 1 | 0b010 | 512 | 1 | 0b011 | 1024 | 1 | 0b100 | 2048 | 1 | 0b101 | 4096 | 1 | 0b110 | Reserved | 1 | 0b111 | Reserved | |
| pres3 | pres | Prescaler | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0b000 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0b001 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0b010 | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0b011 | 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0b100 | 16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0b101 | 32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0b110 | 64 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0b111 | 128 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0b000 | 256 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0b010 | 512 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0b011 | 1024 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0b100 | 2048 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0b101 | 4096 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0b110 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0b111 | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Wakeup Timer Control | | | WUTn_CTRL | | [0x0010] |
|----------------------|-------|--------|-----------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 2:0 | tmode | R/W | 0 | Timer Mode Select Sets the timer's operating mode. 0: One-Shot 1: Continuous 2 - 7: Reserved for Future Use | |

Preliminary Draft 08/31/2020

20. Watchdog Timer (WDT)

The watchdog timer (WDT) protects against corrupt or unreliable software, power faults, and other system-level problems which may place the IC into an improper operating state. Application software must periodically write a special sequence to a dedicated register to confirm the application is operating correctly. Failure to reset the watchdog timer within a user-specified time frame can first generate an interrupt allowing the application the opportunity to identify and correct the problem. In the event the application cannot regain normal operation, as a last resort the watchdog timer can generate a system reset.

Some instances provide a windowed timer function. These instances support an additional feature that can detect watchdog timer resets that occur too early as well as too late (or never). This could happen if program execution is corrupted and is accidentally forced into a tight loop of code that contains a watchdog sequence. This would not be detected with a traditional WDT, because the end of the timeout periods would never be reached. A new set of "watchdog timer early" fields are available to support the lower limits required for windowing. Traditional watchdog timers can only detect a loss of program control that fails to reset the watchdog timer.

Each time the application performs a reset, as early as possible in the application software, the peripheral control register should be examined to determine if the reset was cause by at WDT late reset event (or WDT early reset event if the window function is supported). If so, software should take the desired action as part of its restart sequence.

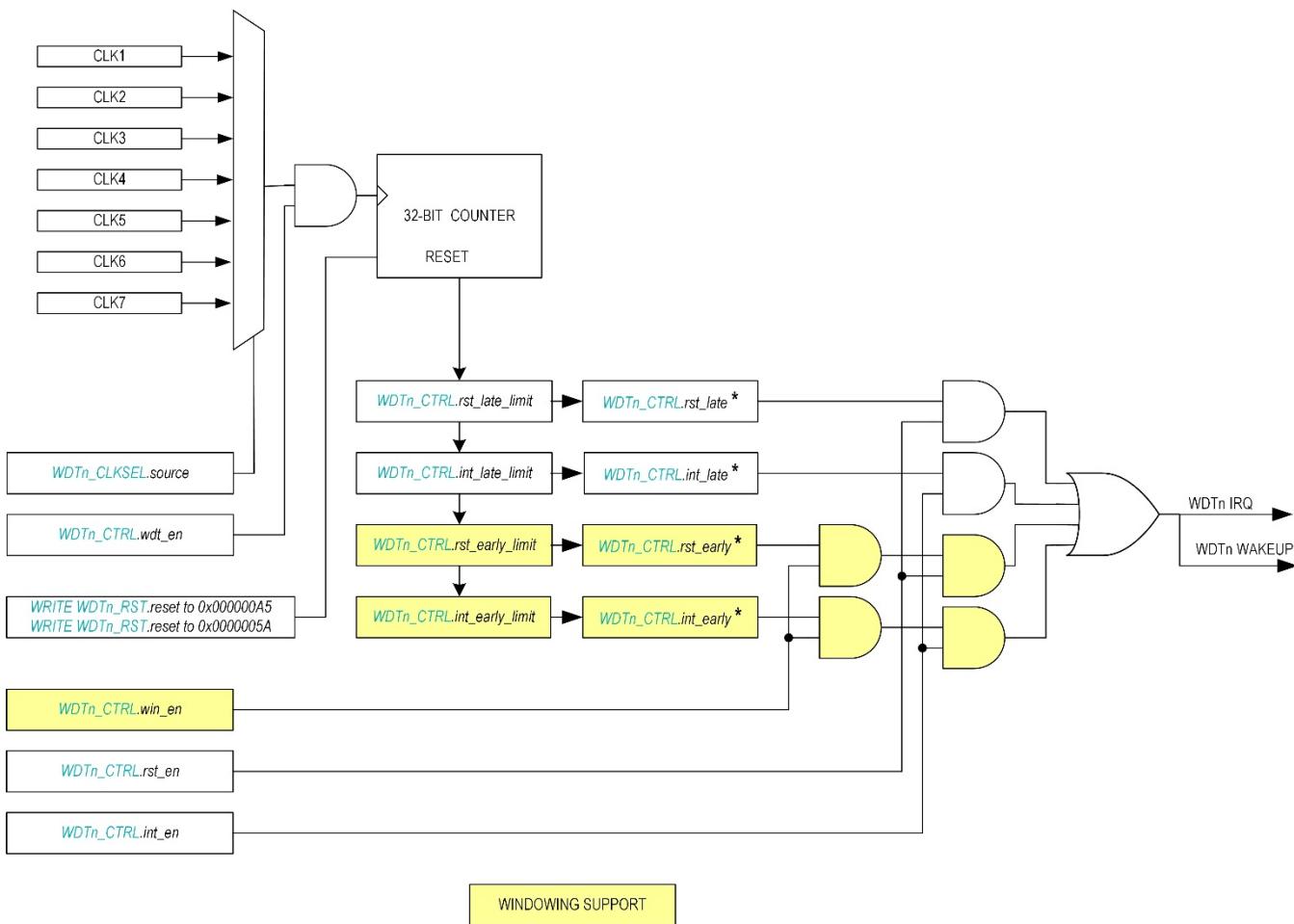
The WDT is a critical safety feature, and most fields are reset on POR or system reset events only.

Features:

- Single ended (legacy) watchdog timeout
- Windowed mode adds lower-limit timeout settings to detect loss of control in tight code loops.
- Configurable clock source
- Configurable time-base
- Programmable upper and lower limits for reset and interrupts from 2^{16} to 2^{1327} time-base ticks
- New register to read the WDT counter register, simplifying code development

Figure 20-1: Windowed Watchdog Timer Block Diagram shows the block diagram of the WDT.

Figure 20-1: Windowed Watchdog Timer Block Diagram



* INTERRUPT FLAGS ARE SET REGARDLESS OF *.win_en*, AND *.int_en* (FOR INTERRUPTS) and *.rst_en* (FOR WDT RESETS).

20.1 Instances

Table 20-1: MAX78000 WDT Instances Summary shows the peripheral instance, the available clock sources, and indicates which instances support the windowed watchdog functionality.

Table 20-1: MAX78000 WDT Instances Summary

| Instance | Register Access Name | Window Support | External Clock | CLK1 | CLK2 | CLK3 |
|----------|----------------------|----------------|----------------|------|-------|------|
| WDTO | WDTO | Yes | N/A | PCLK | IBRO | - |
| LPWDTO | WDT1 | Yes | N/A | IBRO | ERTCO | INRO |

20.2 Usage

When enabled, *WDTn_CNT.count* increments once every t_{WDTCLK} period. During correct operation, the WDT will periodically execute the feed sequence and reset *WDTn_CNT.count* to 0x0000 0000 within the target window.

The upper and lower limits of the target window are user-configurable to accommodate different applications and non-deterministic execution times within an application.

The WDT can generate interrupt and/or reset events in response to the WDT activity. Interrupts are typically configured to respond first to an event outside the target window. The approach is that a minor system event may have temporarily delayed the execution of the feed sequence, so the event can be diagnosed in an interrupt routine and control returned to the system. When the WDT feed sequence occurs much earlier than expected or not at all, a reset event can be generated that will force the system to a known good state before continuing.

Traditional WDTs only detect execution errors that fail to perform the WDT feed sequence. If the counter reaches the WDT late interrupt threshold, the device will attempt to regain program control by vectoring to the dedicated WDT ISR. The ISR should reset the WDT counter, perform the desired recovery activity, and then return execution to a known good address.

If the execution error prevents the successful execution of the ISR, the WDT continues to increment until the count reaches the WDT late reset threshold. The WDT will generate a late reset event which sets the WDT late reset flag and generates a system interrupt.

Instances which support the window feature (`WDTn_CTRL.win_en = 1`) can generate a WDT early interrupt event if the WDT feed sequence occurs earlier than expected. In an analogous manner the device will attempt to regain program control by vectoring to the dedicated WDT ISR. The WDT ISR should reset the WDT counter, perform the desired recovery activity ,and then return execution to a known good address.

A WDT feed sequence that occurs earlier than the WDT early reset threshold indicates the execution error is significant enough to initiate a reset the device to correct the problem. The WDT will generate an early reset event which sets the WDT late reset flag and generates a system interrupt.

The event flags are set regardless of the corresponding interrupt or reset enable. This includes the early interrupt and early event flags, even if the WDT is disabled (`WDTn_CTRL.win_en = 0`).

20.3 WDT Feed Sequence

The WDT feed sequence protects the system against unintentional altering of the WDT count and unintentional enabling or disabling of the timer itself.

Two consecutive write instructions to the `WDTn_RST.reset` field are required to reset the `WDTn_CNT.count = 0`. Global interrupts should be disabled immediately before, and reenabled after the writes, to ensure both writes to the `WDTn_RST.reset` field complete without interruption.

The feed sequence must also be performed immediately before enabling the WDT, to prevent an accidental triggering of the reset or interrupt as soon as the timer is enabled. There is no timed access window for these write operations; the operations can be separated by any length of time as long as they occur in the required sequence

1. Disable interrupts.
2. In consecutive write operations:
 - a. Write `WDTn_RST.reset`: 0x000000A5
 - b. Write `WDTn_RST.reset`: 0x0000005A
3. If desired, enable or disable the timer
4. Re-enable interrupts

20.4 WDT Events

Multiple events are supported as shown in *Table 20-2: MAX78000 WDT Event Summary*. The corresponding event flag is set when the event occurs.

Typically the system is configured such that the late interrupt events will occur prior to the late reset events, and early interrupts will occur when the feed sequence has the least error from the target time, prior to the early reset events.

The event flags will be set even if the corresponding interrupt or reset enable flags regardless of the state of the corresponding enable fields. This includes the early interrupt and early event flags, even if `WDTn_CTRL.win_en = 0`.

Software must clear the all event flags before enabling the timers.

Table 20-2: MAX78000 WDT Event Summary

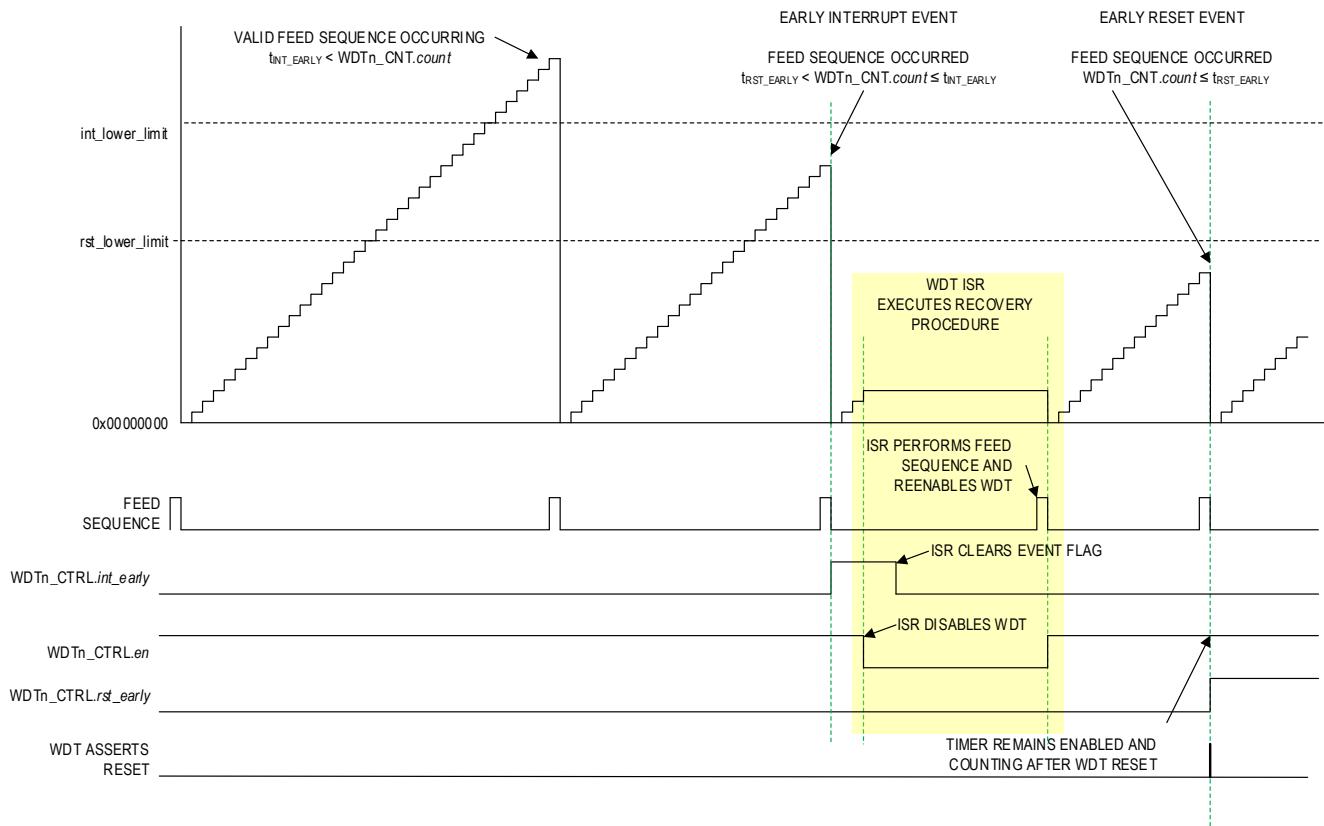
| Event | Condition | Local Interrupt Event Flag | Local Interrupt Event Enable |
|-----------------|---|---|-----------------------------------|
| Early Interrupt | Feed sequence occurs while <code>WDTn_CTRL.rst_early_val ≤ WDTn_CNT.count < WDTn_CTRL.int_early_val</code> <code>WDTn_CTRL.win_en = 1</code> | <code>WDTn_CTRL.int_early</code> | <code>WDTn_CTRL.wdt_int_en</code> |
| Early Reset | Feed sequence occurs while <code>WDTn_CNT.count < WDTn_CTRL.rst_early_val</code> <code>WDTn_CTRL.win_en = 1</code> | <code>WDTn_CTRL.rst_early</code> | <code>WDTn_CTRL.wdt_RST_en</code> |
| Interrupt Late | <code>WDTn_CNT.count = WDTn_CTRL.int_late_val</code> | <code>WDTn_CTRL.int_late</code> | <code>WDTn_CTRL.wdt_int_en</code> |
| Reset Late | <code>WDTn_CNT.count = WDTn_CTRL.rst_late_val</code> | <code>WDTn_CTRL.rst_late</code> | <code>WDTn_CTRL.wdt_RST_en</code> |
| Timer Enabled | <code>WDTn_CTRL.clkrdy</code> 0 → 1 | No event flags are set by a timer enabled event | |

20.4.1 WDT Early Reset

The early reset event occurs if software performs the WDT feed sequence while `WDTn_CNT.count < WDTn_CTRL.rst_late_val` threshold as shown in [Table 20-2: MAX78000 WDT Event Summary](#).

[Figure 20-2: WDT Early Interrupt and Reset Event Sequencing Details](#) shows the sequencing details associated with an early reset event.

Figure 20-2: WDT Early Interrupt and Reset Event Sequencing Details



The following occurs when a WDT early reset event occurs:

1. Hardware sets `WDTn_CTRL.rst_early` to 1.
2. Hardware initiates a system reset:
3. Hardware resets `WDTn_CNT.count` to 0x0000_0000 during the reset event.
4. The `WDTn_CTRL.en` field is unaffected by a system reset. The WDT continues incrementing.
5. The `WDTn_CTRL.rst_early` field is unaffected by a system reset.

20.4.2 WDT Early Interrupt

The early interrupt event occurs if software performs the WDT feed sequence while `WDTn_CTRL.rst_early_val ≤ WDTn_CNT.count < WDTn_CTRL.int_early_val` as shown in [Table 20-2: MAX78000 WDT Event Summary](#). [Figure 20-2: WDT Early Interrupt and Reset Event Sequencing Details](#) shows the sequencing details associated with an early reset event. shows the sequencing details associated with an early interrupt event, including the required functions performed by the WDT ISR.

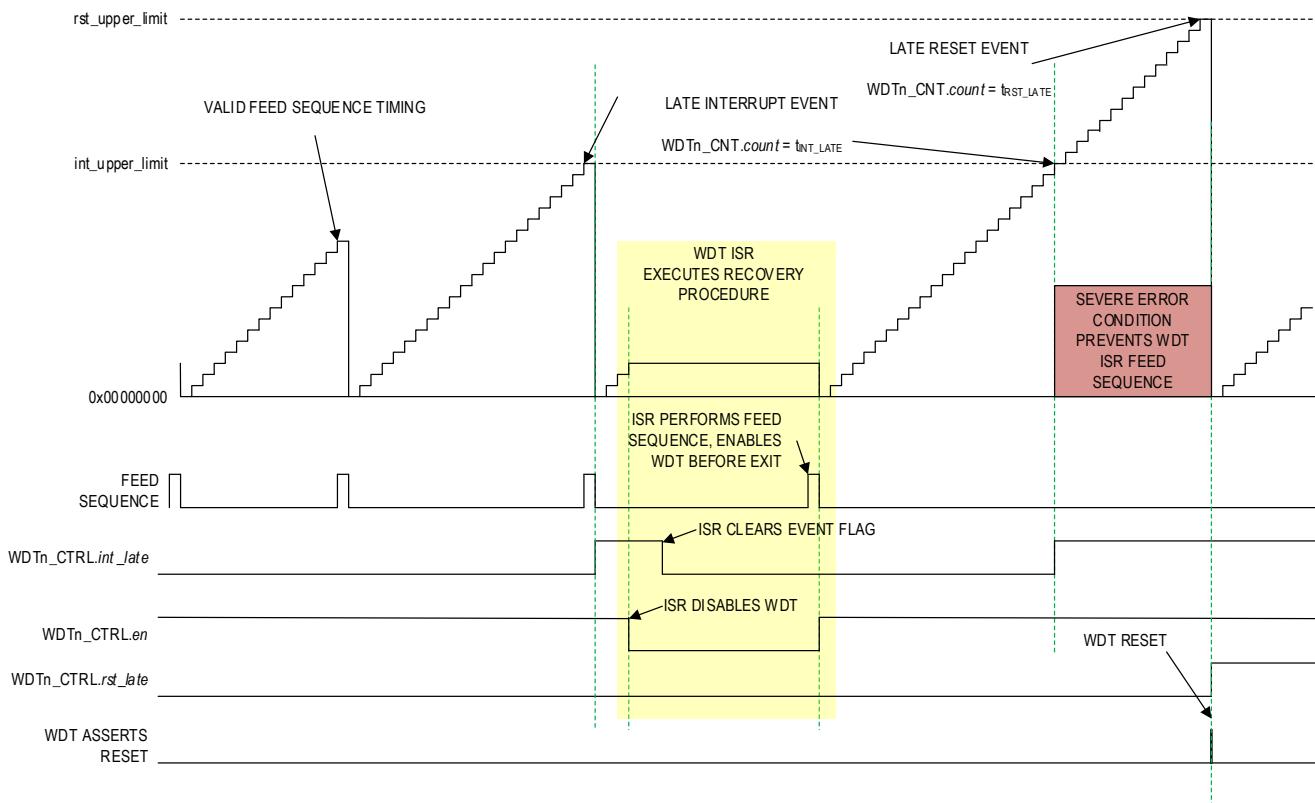
The following occurs when WDT late interrupt event occurs:

1. Hardware sets `WDTn_CTRL.int_late` to 1.
2. Hardware initiates an interrupt if enabled.

20.4.3 WDT Late Reset

The late reset event occurs if the counter increments to the point where $WDTn_CNT.count = WDTn_CTRL.rst_late$ threshold as shown in [Table 20-2: MAX78000 WDT Event Summary](#). [Figure 20-3: WDT Late Interrupt and Reset Event Sequencing Details](#) shows the sequencing details associated with a late reset event.

Figure 20-3: WDT Late Interrupt and Reset Event Sequencing Details



The following occurs when a WDT late reset event occurs:

1. Hardware sets $WDTn_CTRL.rst_late$ to 1.
2. Hardware initiates a system reset:
3. Hardware resets $WDTn_CNT.count$ to 0x0000_0000 during the reset event.
4. The $WDTn_CTRL.en$ field is unaffected by a system reset. The WDT continues incrementing.
5. The $WDTn_CTRL.rst_late$ field is unaffected by a system reset.

20.4.4 WDT Late Interrupt

The late reset event occurs if the counter increments to the point where $WDTn_CNT.count = WDTn_CTRL.rst_late$ threshold as shown in [Table 20-2: MAX78000 WDT Event Summary](#). [Figure 20-3: WDT Late Interrupt and Reset Event Sequencing Details](#) shows the sequencing details associated with a late interrupt event, including the required functions performed by the WDT ISR.

The following occurs when WDT late interrupt event occurs:

1. Hardware sets $WDTn_CTRL.int_late$ to 1.
2. Hardware initiates an interrupt if enabled.

20.5 Initializing the WDT

The full procedure for configuring the WDT is shown below:

1. Execute the WDT feed sequence and disable the WDT:
2. Disable global interrupts
3. Write `WDTn_RST.reset` to 0x0000 00A5.
4. Write `WDTn_RST.reset` to 0x0000 005A. Hardware will reset `WDTn_CNT.count` = 0x0000 0000.
5. Set `WDTn_CTRL.en` to 0 to disable the WDT.
6. Verify the peripheral is disabled before proceeding:
7. Poll `WDTn_CTRL.clkrdy` until it reads 1, or
8. Set `WDTn_CTRL.clk_rdy_ie` = 1 to generate a WDT enabled interrupt event.
9. Re-enable global interrupts.
10. Configure `WDTn_CLKSEL.source` to select the clock source.
11. Configure the traditional/legacy thresholds:
 - a. Configure `WDTn_CTRL.int_late_thd` to the desired threshold for the WDT late interrupt event.
 - b. Configure `WDTn_CTRL.rst_late_val` to the desired threshold for the WDT late reset event.
14. If using the optional windowed WDT feature:
 - a. Set `WDTn_CTRL.win_en` = 1 to enable the windowed WDT feature.
 - b. Configure `WDTn_CTRL.int_early_val` to the desired threshold for the WDT early interrupt event.
 - c. Configure `WDTn_CTRL.rst_early_val` to the desired threshold for the WDT early reset event.
15. Set `WDTn_CTRL.wdt_int_en` to generate an interrupt when a WDT late interrupt event occurs. If `WDTn_CTRL.win_en` = 1 an interrupt will be generated by both a WDT late interrupt event and also a WDT early interrupt event.
16. Set `WDTn_CTRL.wdt_rst_en` to generate an interrupt when a WDT late reset event occurs. If `WDTn_CTRL.win_en` = 1 an interrupt will be generated by a WDT late reset event and also a WDT early reset event.
17. Execute the WDT feed sequence and enable the WDT:
18. Disable global interrupts
19. Write `WDTn_RST.reset` to 0x0000 00A5.
20. Write `WDTn_RST.reset` to 0x0000 005A. Hardware will reset `WDTn_CNT.count` = 0x0000 0000.
21. Set `WDTn_CTRL.en` to 1 to enable the WDT.
22. Verify the peripheral is enabled before proceeding:
23. Poll `WDTn_CTRL.clkrdy` until it reads 1, or
24. Set `WDTn_CTRL.clkrdy_ie` = 1 to generate a WDT enabled event interrupt.
25. Re-enable global interrupts.

20.6 Resets

The WDT is a critical safety feature, and most of the fields are set on POR or system reset events only.

20.7 Using the WDT as a Long-Interval Timer

One application of the WDT is as a very long interval timer in ACTIVE mode. The timer can be configured to generate a WDT late interrupt event for as long as 2^{32} periods of the selected watchdog clock source. The WDT should not be enabled to generate WDT reset events in this application.

20.8 Using the WDT as a Long-Interval Wakeup Timer

Another application of the WDT is as a very long interval wakeup source from SLEEP mode. The timer can be configured to generate a WDT wakeup event for as long as 2^{32} periods of the selected watchdog clock source. Perform the following procedure to initialize the WDT for this purpose:

20.8.1 Registers

See [Table 2-4: APP Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 20-3: WDT Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 20-3: WDT Register Summary

| Offset | Register | Name |
|--------|-----------------------------|---------------------------|
| 0x0000 | WDTn_CTRL | WDT Control Register |
| 0x0004 | WDTn_RST | WDT Reset Register |
| 0x0008 | WDTn_CLKSEL | WDT Clock Select Register |
| 0x000C | WDTn_CNT | WDT Count Register |

20.8.2 Register Details

Table 20-4: WDT Control Register

| WDT Control | | | | WDTn_CTRL | [0x0000] |
|-------------|-----------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31 | rst_late | R/W | 0 | Reset Late Event A watchdog reset event occurred after the time specified in WDTn_CTRL.rst_late_val . This flag will be set even if WDTn_CTRL.win_en = 0 or WDTn_CTRL.wdt_RST_en = 0. The user will have to clear the flag appropriately in case of carried over flags from prior operations. 0: Watchdog did not cause reset event. 1: Watchdog reset occurred after WDTn_CTRL.rst_early_val . | |
| 30 | rst_early | R/W | 0 | Reset Early Event A watchdog reset event occurred before the time specified in WDTn_CTRL.rst_early_val . This flag will be set even if WDTn_CTRL.win_en = 0 or WDTn_CTRL.wdt_RST_en = 0. The user will have to clear the flag appropriately in case of carried over flags from prior operations. 0: Watchdog did not cause reset event. 1: Watchdog reset occurred before WDTn_CTRL.rst_early_val . | |
| 29 | win_en | R/W | 0 | Window Function Enable 0: Disabled. WDT recognizes interrupt late and reset late events, supporting legacy implementations. 1: Enabled | |
| 28 | clkrdy | R | 0 | Clock Status This field is cleared by hardware when software changes the state of WDTn_CTRL.en . Hardware sets the field to 1 when the change to the requested enable/disable is complete. 0: WDT clock is off 1: WDT clock is on | |

| WDT Control | | | | WDTn_CTRL | [0x0000] |
|-------------|---------------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 27 | clkrdy_ie | R/W | 0 | Clock Switch Ready Interrupt Enable This interrupt prevents the user from needing to poll the WDTn_CTRL.clkrdy all the time. When WDTn_CTRL.clkrdy goes from high to low, the interrupt signals the transition is complete. 0: Disabled 1: Enabled | |
| 26:24 | - | RO | 0 | Reserved | |
| 23:20 | rst_early_val | R/W | 0 | Reset Early Event Threshold 0x0: $2^{31} \times t_{WDTCLOCK}$ 0x1: $2^{30} \times t_{WDTCLOCK}$ 0x2: $2^{29} \times t_{WDTCLOCK}$ 0x3: $2^{28} \times t_{WDTCLOCK}$ 0x4: $2^{27} \times t_{WDTCLOCK}$ 0x5: $2^{26} \times t_{WDTCLOCK}$ 0x6: $2^{25} \times t_{WDTCLOCK}$ 0x7: $2^{24} \times t_{WDTCLOCK}$ 0x8: $2^{23} \times t_{WDTCLOCK}$ 0x9: $2^{22} \times t_{WDTCLOCK}$ 0xA: $2^{21} \times t_{WDTCLOCK}$ 0xB: $2^{20} \times t_{WDTCLOCK}$ 0xC: $2^{19} \times t_{WDTCLOCK}$ 0xD: $2^{18} \times t_{WDTCLOCK}$ 0xE: $2^{17} \times t_{WDTCLOCK}$ 0xF: $2^{16} \times t_{WDTCLOCK}$ | |
| 19:16 | int_early_val | R/W | 0 | Interrupt Early Event Threshold 0x0: $2^{31} \times t_{WDTCLOCK}$ 0x1: $2^{30} \times t_{WDTCLOCK}$ 0x2: $2^{29} \times t_{WDTCLOCK}$ 0x3: $2^{28} \times t_{WDTCLOCK}$ 0x4: $2^{27} \times t_{WDTCLOCK}$ 0x5: $2^{26} \times t_{WDTCLOCK}$ 0x6: $2^{25} \times t_{WDTCLOCK}$ 0x7: $2^{24} \times t_{WDTCLOCK}$ 0x8: $2^{23} \times t_{WDTCLOCK}$ 0x9: $2^{22} \times t_{WDTCLOCK}$ 0xA: $2^{21} \times t_{WDTCLOCK}$ 0xB: $2^{20} \times t_{WDTCLOCK}$ 0xC: $2^{19} \times t_{WDTCLOCK}$ 0xD: $2^{18} \times t_{WDTCLOCK}$ 0xE: $2^{17} \times t_{WDTCLOCK}$ 0xF: $2^{16} \times t_{WDTCLOCK}$ | |
| 15:13 | - | RO | 0 | Reserved | |
| 12 | int_early | R/W | 0 | Interrupt Early Flag A feed sequence was performed earlier than the time determined by the WDTn_CTRL.int_early_thd field. This flag will be set even if WDTn_CTRL.win_en = 0. 0: No interrupt event 1: Interrupt event occurred. Generates a WDT IRQ if WDTn_CTRL.wdt_int_en = 1. | |

| WDT Control | | | | WDTn_CTRL | [0x0000] |
|-------------|--------------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 11 | wdt_RST_en | R/W | 0 | WDT Reset Enable 0: Disabled 1: Enabled | |
| 10 | wdt_INT_en | R/W | 0 | WDT Interrupt Enable 0: Disabled 1: Enabled | |
| 9 | int_late | R/W | 0 | Interrupt Late Flag A watchdog feed sequence did not occur before the time determined by the <i>WDTn_CTRL.int_late_val</i> field. 0: No interrupt event 1: Interrupt event occurred. Generates a WDT IRQ if <i>WDTn_CTRL.wdt_int_en</i> = 1. | |
| 8 | en | R/W | 0 | WDT Enable This field enables/disables the WDTCLK into the peripheral. <i>WDTn_CTRL.count</i> holds its value while the WDT is disabled. The WDT feed sequence must be performed immediately before any change to this field. 0: Disabled 1: Enabled | |
| 7:4 | rst_late_val | R/W | 0 | Reset Late Event Threshold 0x0: $2^{31} \times t_{WDTCLK}$ 0x1: $2^{30} \times t_{WDTCLK}$ 0x2: $2^{29} \times t_{WDTCLK}$ 0x3: $2^{28} \times t_{WDTCLK}$ 0x4: $2^{27} \times t_{WDTCLK}$ 0x5: $2^{26} \times t_{WDTCLK}$ 0x6: $2^{25} \times t_{WDTCLK}$ 0x7: $2^{24} \times t_{WDTCLK}$ 0x8: $2^{23} \times t_{WDTCLK}$ 0x9: $2^{22} \times t_{WDTCLK}$ 0xA: $2^{21} \times t_{WDTCLK}$ 0xB: $2^{20} \times t_{WDTCLK}$ 0xC: $2^{19} \times t_{WDTCLK}$ 0xD: $2^{18} \times t_{WDTCLK}$ 0xE: $2^{17} \times t_{WDTCLK}$ 0xF: $2^{16} \times t_{WDTCLK}$ | |

| WDT Control | | | | WDTn_CTRL | [0x0000] |
|-------------|--------------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 3:0 | int_late_val | R/W | 0 | Interrupt Late Event Threshold 0x0: $2^{31} \times t_{WDTCLK}$ 0x1: $2^{30} \times t_{WDTCLK}$ 0x2: $2^{29} \times t_{WDTCLK}$ 0x3: $2^{28} \times t_{WDTCLK}$ 0x4: $2^{27} \times t_{WDTCLK}$ 0x5: $2^{26} \times t_{WDTCLK}$ 0x6: $2^{25} \times t_{WDTCLK}$ 0x7: $2^{24} \times t_{WDTCLK}$ 0x8: $2^{23} \times t_{WDTCLK}$ 0x9: $2^{22} \times t_{WDTCLK}$ 0xA: $2^{21} \times t_{WDTCLK}$ 0xB: $2^{20} \times t_{WDTCLK}$ 0xC: $2^{19} \times t_{WDTCLK}$ 0xD: $2^{18} \times t_{WDTCLK}$ 0xE: $2^{17} \times t_{WDTCLK}$ 0xF: $2^{16} \times t_{WDTCLK}$ | |

Table 20-5: WDT Reset Register

| WDT Reset | | | | WDTn_RST | [0x0004] |
|-----------|-------|--------|----------------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:8 | - | RO | 0 | Reserved Do not modify this field. | |
| 7:0 | reset | R/W | 0 [†] | Reset Watchdog Timer Count Writing the WDT feed sequence, in two consecutive write instructions, to this register resets the internal counter to 0x0000 0000. 1. Write <code>WDTn_RST.reset</code> : 0x0000 00A5 2. Write <code>WDTn_RST.reset</code> : 0x0000 005A Writes to the <code>WDTn_CTRL.en</code> field, which enable or disable the WDT, must be the next instruction following the WDT feed sequence. [†] Note: This field is set to 0 on a POR and is not affected by other resets. | |

Table 20-6: WDT Clock Source Select Register

| WDT Clock Source Select | | | | WDTn_CLKSEL | [0x0008] |
|-------------------------|--------|--------|----------------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:3 | - | RO | 0 | Reserved | |
| 2:0 | source | R/W | 0 [†] | Clock Source Select Refer to the Table 20-1: MAX78000 WDT Instances Summary for available clock options. 0x0: PCLK 0x1: CLK1 0x2: CLK2 0x3: CLK3 0x4: CLK4 0x5: CLK5 0x6: CLK6 0x7: CLK7 [†] Note: This field is only reset on a POR and unaffected by other resets. | |

Table 20-7: WDT Count Register

| WDT Count | | | | WDTn_CNT | [0x000C] |
|-----------|-------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | count | R | 0 | WDT Counter The counter value for debug. This register is reset by system reset, as well as the watchdog feeding sequence. When the WDT clock is off, the feeding sequence generates an asynchronous reset of 1 PCLK width. When the WDT clock is on, the feeding sequence generates a synchronous reset that is handshake to the WDT clock domain. | [0x000C] |

21. Pulse Train Engine (PT)

Each independent pulse train engine operates either in Square Wave mode which generates a continuous 50% duty-cycle square wave, or pulse train mode which generates a continuous programmed bit pattern from 2- to 32-bits in length. Pulse train engines are used independently or may be synchronized together to generate signals in unison. The frequency of each generated output can be set separately based on a divisor of the Peripheral Clock.

21.1 Instances

The device provides 16 instances of the pulse train engine peripheral.

- PTO to PT15

All peripheral registers share a common register set.

21.2 Pulse Train Engine Features

The pulse train outputs with individually programmable modes, patterns and output enables. The pulse train engine uses the Peripheral Clock (PCLK), $f_{PTE_CLK} = f_{PCLK}$, ensuring all pulse train outputs use the same clock source.

- Independent or synchronous pulse train output operation
- Atomic Enable and Atomic Disable
- Synchronous enable or disable of pulse train output(s) without modification to non-intended pulse train outputs
- Multiple Output Modes:
 - Square Wave Output mode generates a repeating square wave (50% duty cycle)
 - Pattern Output mode for generating a customizable output wave based on a programmable bit pattern from 2 to 32 output cycles
- Global clock for all generated outputs
- Individual rate configuration for each pulse train output
- Configuration registers are modifiable while the pulse train engine is running
- Pulse train outputs can be halted and resumed at the same point

21.3 Engine

The pulse train engine uses the Peripheral Clock as the peripheral input clock. Each pulse train output is individually configurable and independently controlled.

The following sections describe the available configuration options for each individual pulse train output.

21.3.1 Pulse Train Output Modes

Each pulse train output supports the following modes:

- Pulse Train Mode
- Bit Patter Length
- Square Wave Mode

21.3.1.1 Pulse Train Mode

When pulse train n (PTn) is configured in pulse train mode, the configuration also includes the bit length (up to 32-bits) of the custom pulse train. This is configured using the 5-bit field $PTn_RATE_LENGTH.mode$ as follows:

$PTn_RATE_LENGTH.mode = 1$ (PTn configured in Square Wave mode)

$PTn_RATE_LENGTH.mode > 1$ (PTn configured in pulse train mode. The value of mode is the pattern bit length.)

$PTn_RATE_LENGTH.mode = 0$ (PTn bit length configured for pulse train mode, 32-bit pattern)

21.3.1.2 In Pulse Train Mode, Set the Bit Pattern

If an output is set to pulse train mode, then configure a custom bit pattern from 2-bits to 32-bits in length in the 32-bit register [PTn_TRAIN](#). The pattern is shifted out least significant bit (LSB) first. If the output is configured in Square Wave mode, then the [PTn_TRAIN](#) register is ignored.

Equation 21-1: Pulse Train Mode Output Function

$$PTn_TRAIN = [\text{Bit pattern for } PTn]$$

Synchronize Two or More Outputs, if Needed

The write-only register [PTG_RESET](#) “PT Global Resync” allows two or more outputs to be reset and synchronized. Write to any bit in [PTG_RESET](#) to simultaneously reset any outputs in pulse train mode to the beginning of the pattern (the LSB) set in the [PTn_TRAIN](#) bit-pattern register, and reset the output to 0 for outputs in Square Wave mode.

21.3.1.3 Pulse Train Loop Mode

By default, a pulse train engine runs indefinitely until it is disabled by firmware.

A pulse train engine can be configured to repeat its pattern a specified number of times, called Loop mode. To select Loop mode, write a non-zero value to the 16-bit field [PTn_LOOP.count](#). When the pulse train engine is enabled, this field decrements by 1 each time a complete pattern is shifted through the output pin. When the count reaches 0, the output is halted, and the corresponding flag in the [PTG_INTFL](#) register is set.

21.3.1.4 Pulse Train Loop Delay

If the pulse train is configured in Loop mode, a delay can be inserted after each repeated output pattern. To enable a delay, write the 12-bit field [PTn_LOOP.delay](#) with the number of Peripheral Clock cycles to delay between the most significant bit (MSB) of the last pattern to the least-significant bit (LSB) of the next pattern. During this delay, the output is held at the MSB of the last pattern. If the loop counter has not reached 0, then it is decremented when the next pattern starts.

21.3.1.5 Pulse Train Automatic Restart Mode

When an engine in pulse train mode is in Loop mode and stops when the loop count reaches 0, this is called a Stop Event. A Stop Event can optionally trigger one or more pulse trains to restart from the beginning. This is called Automatic Restart mode. While only pulse train engines operating in pulse train mode can operate in Loop mode and can optionally restart a pulse train engine, Automatic Restart mode can trigger pulse train engines operating in pulse train mode or in Square Wave mode.

If a running pulse train engine is triggered by another pulse train’s Stop Event, Automatic Restart restarts the running pulse train engine from the beginning of its pattern. If a pulse train engine is triggered by another pulse train’s Stop Event, and it is not running, Automatic Restart sets the enable bit to 1, and starts the pulse train engine.

The settings for this mode are contained in the [PTn_RESTART](#) register for each pulse train engine. Note that the configuration for automatic restart is set using the pulse engine(s) triggered by the automatic restart, not the pulse train engine(s) that trigger the automatic restart. For example, the [PT8_RESTART](#) register configures which pulse train engine triggers PT8 to restart.

Each pulse train engine can be configured to perform an Automatic Restart when it detects a Stop Event from one or two pulse trains.

If [PTn_RESTART.on_pt_x_loop_exit](#) = 1, then pulse train engine n automatically restarts when it detects a Stop Event from pulse train x, where x is the value in the 5-bit field [PTn_RESTART.pt_x_select](#).

If [PTn_RESTART.on_pt_y_loop_exit](#) = 1, then pulse train engine n automatically restarts when it detects a Stop Event from pulse train y, where y is the value in 5-bit field [PTn_RESTART.pt_y_select](#).

A pulse train engine can be configured to restart on its own Stop Event, allowing the pulse train to run indefinitely.

Each individual pulse train can be configured for:

- No Automatic Restart
- Automatic Restart triggered by a Stop Event from Pulse Train x only
- Automatic Restart triggered by a Stop Event from Pulse Train y only
- Automatic Restart triggered by a Stop Event from both Pulse Train x and Pulse Train y

21.4 Enabling and Disabling a Pulse Train Output

The [PTG_ENABLE](#) register is used to enable and disable each of the individual pulse train outputs. Enable a given pulse train output by setting the respective bit in the [PTG_ENABLE](#) register. Halt a pulse train output by clearing the respective bit in the [PTG_ENABLE](#) register.

Note: Prior to changing a pulse train output's configuration the corresponding pulse train output should be halted to prevent unexpected behavior.

21.5 Atomic Pulse Train Output Enable and Disable

Deterministic enable and disable operations are critical for pulse train outputs that must be synchronized in an application. The [PTG_ENABLE](#) register does not perform atomic access directly. Atomic operations are supported using the registers [PTG_SAFE_EN](#), [PTG_SAFE_DIS](#).

For most pulse train peripherals, enabling and disabling individual pulse trains is performed by setting and clearing bits in the global enable/disable register, which for this peripheral is [PTG_ENABLE](#). For most Arm Cortex-M microcontrollers, this is usually done by bit banding. Because bit banding performs a read, modify, write (RMW), some pulse trains could start and end during the RMW operation, often with unpredictable results.

To ensure safe and predictable operation, two additional registers are used to enable and disable the outputs.

21.5.1 Pulse Train Atomic Enable

[PTG_SAFE_EN](#) “Global Safe Enable” is a write-only register. To safely enable outputs without a RMW, write a 32-bit value to this register with a 1 in the bit positions corresponding to the pulse train engines to be enabled. This immediately sets to 1 the corresponding bits in the [PTG_ENABLE](#) register to 1, which enables the corresponding pulse train engine. Writing a 0 to any bit position in the [PTG_SAFE_EN](#) register has no effect on the state of the corresponding pulse train enable bit. If the corresponding pulse train engine is already enabled and running, writing a 1 to that bit position in the [PTG_SAFE_EN](#) register has no effect.

21.5.2 Pulse Train Atomic Disable

[PTG_SAFE_DIS](#) “Global Safe Disable” is a write-only register for disabling a pulse train engine without performing a RMW. To safely disable pulse train engines, write a 32-bit value to this register with a 1 in the bit positions corresponding to the pulse train engines to be disabled. This immediately clears to 0 the corresponding bits in [PTG_ENABLE](#) which disables the corresponding pulse train engines. Writing a 0 to any bit position in the [PTG_SAFE_DIS](#) register has no effect on the state of the corresponding pulse train enable bit.

Bit banding is not supported for the [PTG_ENABLE](#), [PTG_SAFE_EN](#), and [PTG_SAFE_DIS](#) registers and can have unpredictable results.

21.6 Pulse Train Halt and Disable

Once a pulse train engine is enabled and running, it continues to run until one of the following events stops the output:

- The corresponding enable bit in the [PTG_ENABLE](#) register is cleared to 0 to halt the output.
- A 1 is written to the corresponding disable bit in the [PTG_SAFE_DIS](#) register to halt the output.
- The corresponding resync bit in the [PTG_RESYNC](#) register is cleared to 0 to halt and reset the output.

- *PTn_LOOP* was initialized to a non-zero value, and the loop count has reached 0 (this has no effect in Square Wave mode; it only applies to pulse train mode).

When a pulse train is halted, the corresponding enable bit in *PTG_ENABLE* is automatically cleared to 0.

21.7 Pulse Train Interrupts

Each pulse train can generate an interrupt only if it is configured in pulse train mode, and the loop counter *PTG_SAFE_DIS* was initialized to a non-zero number. When *PTG_SAFE_DIS* counts down to 0, the corresponding status flag in the *PTG_INTFL* register is set. If the corresponding interrupt enable bit in the *PTG_INTEN* register is set, the event also generates an interrupt.

21.8 Registers

See *Table 2-4: APB Peripheral Base Address Map* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in *Table 21-1: Pulse Train Engine Register Summary*. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register *PERIPHERALn_CTRL* resolves to *PERIPHERAL0_CTRL* and *PERIPHERAL1_CTRL* for instances 0 and 1, respectively.

See *Table 2-1: Field Access Definitions* for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 21-1: Pulse Train Engine Register Summary

| Offset | Register | Description |
|----------|------------------------|----------------------------------|
| [0x0000] | <i>PTG_ENABLE</i> | PT Global Enable/Disable Control |
| [0x0004] | <i>PTG_RESYNC</i> | PT Global Resync |
| [0x0008] | <i>PTG_INTFL</i> | PT Stopped Global Status Flags |
| [0x000C] | <i>PTG_INTEN</i> | PT Global Interrupt Enable |
| [0x0010] | <i>PTG_SAFE_EN</i> | PT Global Safe Enable |
| [0x0014] | <i>PTG_SAFE_DIS</i> | PT Global Safe Disable |
| [0x0020] | <i>PTn_RATE_LENGTH</i> | PT0 Configuration |
| [0x0024] | <i>PTn_TRAIN</i> | PT0 Pulse Train Mode Bit Pattern |
| [0x0028] | <i>PTn_LOOP</i> | PT0 Loop Control |
| [0x002C] | <i>PTn_RESTART</i> | PT0 Automatic Restart |
| [0x0030] | <i>PTn_RATE_LENGTH</i> | PT1 Configuration |
| [0x0034] | <i>PTn_TRAIN</i> | PT1 Pulse Train Mode Bit Pattern |
| [0x0038] | <i>PTn_LOOP</i> | PT1 Loop Control |
| [0x003C] | <i>PTn_RESTART</i> | PT1 Automatic Restart |
| [0x0040] | <i>PTn_RATE_LENGTH</i> | PT2 Configuration |
| [0x0044] | <i>PTn_TRAIN</i> | PT2 Pulse Train Mode Bit Pattern |
| [0x0048] | <i>PTn_LOOP</i> | PT2 Loop Control |
| [0x004C] | <i>PTn_RESTART</i> | PT2 Automatic Restart |
| [0x0050] | <i>PTn_RATE_LENGTH</i> | PT3 Configuration |
| [0x0054] | <i>PTn_TRAIN</i> | PT3 Pulse Train Mode Bit Pattern |
| [0x0058] | <i>PTn_LOOP</i> | PT3 Loop Control |
| [0x005C] | <i>PTn_RESTART</i> | PT3 Automatic Restart |
| [0x0060] | <i>PTn_RATE_LENGTH</i> | PT4 Configuration |
| [0x0064] | <i>PTn_TRAIN</i> | PT4 Pulse Train Mode Bit Pattern |
| [0x0068] | <i>PTn_LOOP</i> | PT4 Loop Control |

| Offset | Register | Description |
|----------|------------------------|-----------------------------------|
| [0x006C] | <i>PTn_RESTART</i> | PT4 Automatic Restart |
| [0x0070] | <i>PTn_RATE_LENGTH</i> | PT5 Configuration |
| [0x0074] | <i>PTn_TRAIN</i> | PT5 Pulse Train Mode Bit Pattern |
| [0x0078] | <i>PTn_LOOP</i> | PT5 Loop Control |
| [0x007C] | <i>PTn_RESTART</i> | PT5 Automatic Restart |
| [0x0080] | <i>PTn_RATE_LENGTH</i> | PT6 Configuration |
| [0x0084] | <i>PTn_TRAIN</i> | PT6 Pulse Train Mode Bit Pattern |
| [0x0088] | <i>PTn_LOOP</i> | PT6 Loop Control |
| [0x008C] | <i>PTn_RESTART</i> | PT6 Automatic Restart |
| [0x0090] | <i>PTn_RATE_LENGTH</i> | PT7 Configuration |
| [0x0094] | <i>PTn_TRAIN</i> | PT7 Pulse Train Mode Bit Pattern |
| [0x0098] | <i>PTn_LOOP</i> | PT7 Loop Control |
| [0x009C] | <i>PTn_RESTART</i> | PT7 Automatic Restart |
| [0x00A0] | <i>PTn_RATE_LENGTH</i> | PT8 Configuration |
| [0x00A4] | <i>PTn_TRAIN</i> | PT8 Pulse Train Mode Bit Pattern |
| [0x00A8] | <i>PTn_LOOP</i> | PT8 Loop Control |
| [0x00AC] | <i>PTn_RESTART</i> | PT8 Automatic Restart |
| [0x00B0] | <i>PTn_RATE_LENGTH</i> | PT9 Configuration |
| [0x00B4] | <i>PTn_TRAIN</i> | PT9 Pulse Train Mode Bit Pattern |
| [0x00B8] | <i>PTn_LOOP</i> | PT9 Loop Control |
| [0x00BC] | <i>PTn_RESTART</i> | PT9 Automatic Restart |
| [0x00C0] | <i>PTn_RATE_LENGTH</i> | PT10 Configuration |
| [0x00C4] | <i>PTn_TRAIN</i> | PT10 Pulse Train Mode Bit Pattern |
| [0x00C8] | <i>PTn_LOOP</i> | PT10 Loop Control |
| [0x00CC] | <i>PTn_RESTART</i> | PT10 Automatic Restart |
| [0x00D0] | <i>PTn_RATE_LENGTH</i> | PT11 Configuration |
| [0x00D4] | <i>PTn_TRAIN</i> | PT11 Pulse Train Mode Bit Pattern |
| [0x00D8] | <i>PTn_LOOP</i> | PT11 Loop Control |
| [0x00DC] | <i>PTn_RESTART</i> | PT11 Automatic Restart |
| [0x00E0] | <i>PTn_RATE_LENGTH</i> | PT12 Configuration |
| [0x00E4] | <i>PTn_TRAIN</i> | PT12 Pulse Train Mode Bit Pattern |
| [0x00E8] | <i>PTn_LOOP</i> | PT12 Loop Control |
| [0x00EC] | <i>PTn_RESTART</i> | PT12 Automatic Restart |
| [0x00F0] | <i>PTn_RATE_LENGTH</i> | PT13 Configuration |
| [0x00F4] | <i>PTn_TRAIN</i> | PT13 Pulse Train Mode Bit Pattern |
| [0x00F8] | <i>PTn_LOOP</i> | PT13 Loop Control |
| [0x00FC] | <i>PTn_RESTART</i> | PT13 Automatic Restart |
| [0x0100] | <i>PTn_RATE_LENGTH</i> | PT14 Configuration |
| [0x0104] | <i>PTn_TRAIN</i> | PT14 Pulse Train Mode Bit Pattern |
| [0x0108] | <i>PTn_LOOP</i> | PT14 Loop Control |
| [0x010C] | <i>PTn_RESTART</i> | PT14 Automatic Restart |
| [0x0110] | <i>PTn_RATE_LENGTH</i> | PT15 Configuration |
| [0x0114] | <i>PTn_TRAIN</i> | PT15 Pulse Train Mode Bit Pattern |
| [0x0118] | <i>PTn_LOOP</i> | PT15 Loop Control |

| Offset | Register | Description |
|----------|--------------------|------------------------|
| [0x011C] | <i>PTn_RESTART</i> | PT15 Automatic Restart |

21.8.1 Register Details

Table 21-2: Pulse Train Engine Global Enable/Disable Register

| PT Global Enable/Disable Control | | | PTG_ENABLE | [0x0000] |
|----------------------------------|-------------|--------|------------|--|
| Bits | Field | Access | Reset | Description |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. |
| 15 | enable_pt15 | R/W | 0 | Enable PT15 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |
| 14 | enable_pt14 | R/W | 0 | Enable PT14 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |
| 13 | enable_pt13 | R/W | 0 | Enable PT13 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |
| 12 | enable_pt12 | R/W | 0 | Enable PT12 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |
| 11 | enable_pt11 | R/W | 0 | Enable PT11 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |
| 10 | enable_pt10 | R/W | 0 | Enable PT10 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |
| 9 | enable_pt9 | R/W | 0 | Enable PT9 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |
| 8 | enable_pt8 | R/W | 0 | Enable PT8 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |
| 7 | enable_pt7 | R/W | 0 | Enable PT7 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> |

| PT Global Enable/Disable Control | | | PTG_ENABLE | | [0x0000] |
|----------------------------------|------------|--------|------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 6 | enable_pt6 | R/W | 0 | Enable PT6 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> | |
| 5 | enable_pt5 | R/W | 0 | Enable PT5 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> | |
| 4 | enable_pt4 | R/W | 0 | Enable PT4 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> | |
| 3 | enable_pt3 | R/W | 0 | Enable PT3 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> | |
| 2 | enable_pt2 | R/W | 0 | Enable PT2 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> | |
| 1 | enable_pt1 | R/W | 0 | Enable PT1 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> | |
| 0 | enable_pt0 | R/W | 0 | Enable PT0 0: Disable 1: Enable <i>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</i> | |

If *PTn_LOOP.count* loop counter is set to a non-zero number, when the loop counter counts down to zero then the pulse train engine stops, and the corresponding enable bit is cleared.

Table 21-3: Pulse Train Engine Resync Register

| PT Resync | | | PTG_RESYNC | | [0x0004] |
|-----------|-------|--------|------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | |
| 15 | pt15 | WO | - | Resync Control for PT15 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |

| PT Resync | | | PTG_RESYNC | | [0x0004] |
|-----------|-------|--------|------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 14 | pt14 | WO | - | Resync Control for PT14 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 13 | pt13 | WO | - | Resync Control for PT13 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 12 | pt12 | WO | - | Resync Control for PT12 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 11 | pt11 | WO | - | Resync Control for PT11 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 10 | pt10 | WO | - | Resync Control for PT10 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |

| PT Resync | | | PTG_RESYNC | | [0x0004] |
|-----------|-------|--------|------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 9 | pt9 | WO | - | Resync Control for PT9 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 8 | pt8 | WO | - | Resync Control for PT8 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 7 | pt7 | WO | - | Resync Control for PT7 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 6 | pt6 | WO | - | Resync Control for PT6 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 5 | pt5 | WO | - | Resync Control for PT5 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |

| PT Resync | | | PTG_RESYNC | | [0x0004] |
|-----------|-------|--------|------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 4 | pt4 | WO | - | Resync Control for PT4 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 3 | pt3 | WO | - | Resync Control for PT3 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 2 | pt2 | WO | - | Resync Control for PT2 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 1 | pt1 | WO | - | Resync Control for PT1 Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |
| 0 | pt0 | WO | - | Resync Control for PTO Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i> | |

Table 21-4:Pulse Train Engine Stopped Interrupt Flag Register

| PT Stopped Interrupt Flag | | | PTG_INTFL | | [0x0008] |
|---------------------------|-------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | |

| PT Stopped Interrupt Flag | | | PTG_INTFL | | [0x0008] |
|---------------------------|-------|--------|-----------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 15 | pt15 | R/W1C | 0 | PT15 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 14 | pt14 | R/W1C | 0 | PT14 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 13 | pt13 | R/W1C | 0 | PT13 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 12 | pt12 | R/W1C | 0 | PT12 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 11 | pt11 | R/W1C | 0 | PT11 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 10 | pt10 | R/W1C | 0 | PT10 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 9 | pt9 | R/W1C | 0 | PT9 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 8 | pt8 | R/W1C | 0 | PT8 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 7 | pt7 | R/W1C | 0 | PT7 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 6 | pt6 | R/W1C | 0 | PT6 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |

| PT Stopped Interrupt Flag | | | PTG_INTFL | | [0x0008] |
|---------------------------|-------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 5 | pt5 | R/W1C | 0 | PT5 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 4 | pt4 | R/W1C | 0 | PT4 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 3 | pt3 | R/W1C | 0 | PT3 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 2 | pt2 | R/W1C | 0 | PT2 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 1 | pt1 | R/W1C | 0 | PT1 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |
| 0 | pt0 | R/W1C | 0 | PT0 Stopped Status Flag This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped. | |

Table 21-5: Pulse Train Engine Interrupt Enable Register

| PT Interrupt Enable | | | PTG_INTEN | | [0x000C] |
|---------------------|-------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | |
| 15 | pt15 | R/W | 0 | PT15 Interrupt Enable 0: Disabled. 1: Enabled. | |
| 14 | pt14 | R/W | 0 | PT14 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register. 0: Disabled. 1: Enabled. | |
| 13 | pt13 | R/W | 0 | PT13 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register. 0: Disabled. 1: Enabled. | |

| PT Interrupt Enable | | | | | PTG_INTEN | [0x000C] |
|---------------------|-------|--------|-------|--|-----------|----------|
| Bits | Field | Access | Reset | Description | | |
| 12 | pt12 | R/W | 0 | PT12 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 11 | pt11 | R/W | 0 | PT11 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 10 | pt10 | R/W | 0 | PT10 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 9 | pt9 | R/W | 0 | PT9 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 8 | pt8 | R/W | 0 | PT8 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 7 | pt7 | R/W | 0 | PT7 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 6 | pt6 | R/W | 0 | PT6 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 5 | pt5 | R/W | 0 | PT5 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 4 | pt4 | R/W | 0 | PT4 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |
| 3 | pt3 | R/W | 0 | PT3 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <i>PTG_INTFL</i> register. 0: Disabled. 1: Enabled. | | |

| PT Interrupt Enable | | | | | PTG_INTEN | [0x000C] |
|---------------------|-------|--------|-------|--|-----------|----------|
| Bits | Field | Access | Reset | Description | | |
| 2 | pt2 | R/W | 0 | PT2 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register. 0: Disabled. 1: Enabled. | | |
| 1 | pt1 | R/W | 0 | PT1 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register. 0: Disabled. 1: Enabled. | | |
| 0 | pt0 | R/W | 0 | PT0 Interrupt Enable Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register. 0: Disabled. 1: Enabled. | | |

21.8.1.1 Pulse Train Engine Safe Enable Register

A 32-bit value written to this register performs an immediate binary OR with the contents of [PTG_ENABLE](#). The result is immediately stored in the [PTG_ENABLE](#).

Table 21-6: Pulse Train Engine Safe Enable Register

| Pulse Train Engine Safe Enable | | | | | PTG_SAFE_EN | [0x0010] |
|--------------------------------|-------|--------|-------|---|-------------|----------|
| Bits | Field | Access | Reset | Description | | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | | |
| 15 | pt15 | WO | - | Safe Enable Control for PT15 Writing a 1 sets PTG_ENABLE.enable_pt15 . 1: Enable corresponding pulse train 0: No effect | | |
| 14 | pt14 | WO | - | Safe Enable Control for PT14 Writing a 1 sets PTG_ENABLE.enable_pt14 . 1: Enable corresponding pulse train 0: No effect | | |
| 13 | pt13 | WO | - | Safe Enable Control for PT1 Writing a 1 sets PTG_ENABLE.enable_pt13 . 1: Enable corresponding pulse train 0: No effect | | |
| 12 | pt12 | WO | - | Safe Enable Control for PT12 Writing a 1 sets PTG_ENABLE.enable_pt12 . 1: Enable corresponding pulse train 0: No effect | | |
| 11 | pt11 | WO | - | Safe Enable Control for PT11 Writing a 1 sets PTG_ENABLE.enable_pt11 . 1: Enable corresponding pulse train 0: No effect | | |
| 10 | pt10 | WO | - | Safe Enable Control for PT10 Writing a 1 sets PTG_ENABLE.enable_pt10 . 1: Enable corresponding pulse train 0: No effect | | |

| Pulse Train Engine Safe Enable | | | PTG_SAFE_EN | | [0x0010] |
|--------------------------------|-------|--------|-------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 9 | pt9 | WO | - | Safe Enable Control for PT9 Writing a 1 sets PTG_ENABLE.enable_pt9 . 1: Enable corresponding pulse train 0: No effect | |
| 8 | pt8 | WO | - | Safe Enable Control for PT8 Writing a 1 sets PTG_ENABLE.enable_pt8 . 1: Enable corresponding pulse train 0: No effect | |
| 7 | pt7 | WO | - | Safe Enable Control for PT7 Writing a 1 sets PTG_ENABLE.enable_pt7 . 1: Enable corresponding pulse train 0: No effect | |
| 6 | pt6 | WO | - | Safe Enable Control for PT6 Writing a 1 sets PTG_ENABLE.enable_pt6 . 1: Enable corresponding pulse train 0: No effect | |
| 5 | pt5 | WO | - | Safe Enable Control for PT5 Writing a 1 sets PTG_ENABLE.enable_pt5 . 1: Enable corresponding pulse train 0: No effect | |
| 4 | pt4 | WO | - | Safe Enable Control for PT4 Writing a 1 sets PTG_ENABLE.enable_pt4 . 1: Enable corresponding pulse train 0: No effect | |
| 3 | pt3 | WO | - | Safe Enable Control for PT3 Writing a 1 sets PTG_ENABLE.enable_pt3 . 1: Enable corresponding pulse train 0: No effect | |
| 2 | pt2 | WO | - | Safe Enable Control for PT2 Writing a 1 sets PTG_ENABLE.enable_pt2 . 1: Enable corresponding pulse train 0: No effect | |
| 1 | pt1 | WO | - | Safe Enable Control for PT1 Writing a 1 sets PTG_ENABLE.enable_pt1 . 1: Enable corresponding pulse train 0: No effect | |
| 0 | pt0 | WO | - | Safe Enable Control for PT0 Writing a 1 sets PTG_ENABLE.enable_pt0 . 1: Enable corresponding pulse train 0: No effect | |

21.8.1.2 Pulse Train Engine Safe Enable Register

A 32-bit value written to this register performs an immediate binary OR with the contents of [PTG_ENABLE](#). The result is immediately stored in the [PTG_ENABLE](#).

Table 21-7: Pulse Train Engine Safe Disable Register

| Pulse Train Engine Safe Disable | | | | PTG_SAFE_DIS | [0x0014] |
|---------------------------------|-------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | |
| 15 | pt15 | WO | - | Safe Disable Control for PT15 Writing a 1 clears PTG_ENABLE.enable_pt15 . 1: Disable corresponding pulse train 0: No effect | |
| 14 | pt14 | WO | - | Safe Disable Control for PT14 Writing a 1 clears PTG_ENABLE.enable_pt14 . 1: Disable corresponding pulse train 0: No effect | |
| 13 | pt13 | WO | - | Safe Disable Control for PT13 Writing a 1 clears PTG_ENABLE.enable_pt13 . 1: Disable corresponding pulse train 0: No effect | |
| 12 | pt12 | WO | - | Safe Disable Control for PT12 Writing a 1 clears PTG_ENABLE.enable_pt12 . 1: Disable corresponding pulse train 0: No effect | |
| 11 | pt11 | WO | - | Safe Disable Control for PT11 Writing a 1 clears PTG_ENABLE.enable_pt11 . 1: Disable corresponding pulse train 0: No effect | |
| 10 | pt10 | WO | - | Safe Disable Control for PT10 Writing a 1 clears PTG_ENABLE.enable_pt10 . 1: Disable corresponding pulse train 0: No effect | |
| 9 | pt9 | WO | - | Safe Disable Control for PT9 Writing a 1 clears PTG_ENABLE.enable_pt9 . 1: Disable corresponding pulse train 0: No effect | |
| 8 | pt8 | WO | - | Safe Disable Control for PT8 Writing a 1 clears PTG_ENABLE.enable_pt8 . 1: Disable corresponding pulse train 0: No effect | |
| 7 | pt7 | WO | - | Safe Disable Control for PT7 Writing a 1 clears PTG_ENABLE.enable_pt7 . 1: Disable corresponding pulse train 0: No effect | |
| 6 | pt6 | WO | - | Safe Disable Control for PT6 Writing a 1 clears PTG_ENABLE.enable_pt6 . 1: Disable corresponding pulse train 0: No effect | |
| 5 | pt5 | WO | - | Safe Disable Control for PT5 Writing a 1 clears PTG_ENABLE.enable_pt5 . 1: Disable corresponding pulse train 0: No effect | |

| Pulse Train Engine Safe Disable | | | | PTG_SAFE_DIS | [0x0014] |
|---------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 4 | pt4 | WO | - | Safe Disable Control for PT4 Writing a 1 clears <i>PTG_ENABLE.enable_pt4</i> . 1: Disable corresponding pulse train 0: No effect | |
| 3 | pt3 | WO | - | Safe Disable Control for PT3 Writing a 1 clears <i>PTG_ENABLE.enable_pt3</i> . 1: Disable corresponding pulse train 0: No effect | |
| 2 | pt2 | WO | - | Safe Disable Control for PT2 Writing a 1 clears <i>PTG_ENABLE.enable_pt2</i> . 1: Disable corresponding pulse train 0: No effect | |
| 1 | pt1 | WO | - | Safe Disable Control for PT1 Writing a 1 clears <i>PTG_ENABLE.enable_pt1</i> . 1: Disable corresponding pulse train 0: No effect | |
| 0 | pt0 | WO | - | Safe Disable Control for PT0 Writing a 1 clears <i>PTG_ENABLE.enable_pt0</i> . 1: Disable corresponding pulse train 0: No effect | |

Table 21-8: Pulse Train Engine Configuration Register

| Pulse Train Configuration | | | | PTn_RATE_LENGTH | [0x0020] |
|---------------------------|--------------|--------|---------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:27 | mode | R/W | 0b00001 | Square Wave or Pulse Train Output Mode Sets either pulse train mode with length, or Square Wave mode. 0: Pulse train mode, 32-bits long 1: Square Wave mode 2: Pulse train mode, 2-bits long 3: Pulse train mode, 3-bits long ... etc ... 31: Pulse train mode, 31-bits long Note: If this field is set to 1, Square Wave mode, the PTn_LENGTH register is not used. | |
| 26:0 | rate_control | R/W | 0 | Pulse Train Enable and Rate Control Defines the rate at which the output for <i>PTn</i> changes state by setting the divisor of the PT Clock, where: $f_{PTn} = f_{PTE_CLK} / \text{rate_control}$ 0: Output halted 1: $f_{PTn} = f_{PTE_CLK}$ 2: $f_{PTn} = f_{PTE_CLK} / 2$ 3: $f_{PTn} = f_{PTE_CLK} / 3$... 0x7FFF: $f_{PTn} = f_{PTE_CLK} / 0x7FFF$ | |

Table 21-9: Pulse Train Mode Bit Pattern Register

| Pulse Train Mode Bit Pattern | | | PTn_TRAIN | | [0x0024] |
|------------------------------|-----------|--------|-----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | ptn_train | R/W | 0 | Pulse Train Mode Bit Pattern Write the repeating bit pattern that is shifted out, LSB first, when configured in pulse train mode. Set the bit pattern length with the PTn_RATE_LENGTH.mode field. Note: This register is ignored in Square Wave mode. Note: 0x0000 0000 and 0x0001 0000 are invalid values for this register. | |

Table 21-10: Pulse Train n Loop Configuration Register

| Pulse Train Loop Configuration | | | PTn_LOOP | | [0x0028] |
|--------------------------------|-------|--------|----------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:28 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | |
| 27:16 | delay | R/W | 0 | Pulse Train Delay Between Loops Sets the delay, in number of Peripheral Clock cycles, that the output pauses between loops. The bitfield count is decremented after the delay. If firmware writes a 0 to bitfield count, this field is ignored. | |
| 15:0 | count | R/W | 0 | Pulse Train Loop Countdown Sets the number of times a pulse train pattern is repeated until it automatically stops. Reading this field returns the number of loops remaining. When this field counts down to zero, the corresponding PTn_INTFL flag is set. Write 0 to have the pulse train pattern repeat indefinitely. Ignored in Square Wave mode. | |

Table 21-11: Pulse Train n Automatic Restart Configuration Register

| Pulse Train Automatic Restart Configuration | | | PTn_RESTART | | [0x002C] |
|---|-------------------|--------|-------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | |
| 15 | on_pt_y_loop_exit | R/W | 0 | Enable Automatic Restart for This Pulse Train on PTy Stop Event 0: Disable automatic restart 1: When PTy has a Stop Event, automatically restart this pulse train from the beginning of its pattern. | |
| 14:11 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | |

| Pulse Train Automatic Restart Configuration | | | | PTn_RESTART | [0x002C] |
|---|-------------------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 12:8 | pt_y_select | R/W | 0 | Select PTy Select the pulse train number to be associated with PTy. This engine must be in pulse train mode. Ob00000: PT0 Ob00001: PT1 Ob00010: PT2 Ob00011: PT3 Ob00100: PT4 Ob00101: PT5 Ob00110: PT6 Ob00111: PT7 Ob01000: PT8 Ob01001: PT9 Ob01010: PT10 Ob01011: PT11 Ob01100: PT12 Ob01101: PT13 Ob01110: PT14 Ob01111: PT15 Ob1xxxx: Reserved. | |
| 7 | on_pt_x_loop_exit | R/W | 0 | Enable Automatic Restart for this Pulse Train on a PTn Stop Event 0: Disable automatic restart 1: When PTn has a Stop Event, automatically restart pulse train from the beginning of its pattern. | |
| 6:5 | - | RO | 0 | Reserved for Future Use Do not modify this field from its default value. | |
| 4:0 | pt_x_select | R/W | 0 | Select PTn Select the pulse train number to be associated with PTn. This engine must be in pulse train mode. Ob00000: PT0 Ob00001: PT1 Ob00010: PT2 Ob00011: PT3 Ob00100: PT4 Ob00101: PT5 Ob00110: PT6 Ob00111: PT7 Ob01000: PT8 Ob01001: PT9 Ob01010: PT10 Ob01011: PT11 Ob01100: PT12 Ob01101: PT13 Ob01110: PT14 Ob01111: PT15 Ob1xxxx: Reserved. | |

Preliminary Draft 08/31/2020

22. CRC

The Cyclic Redundancy Check (CRC) engine performs CRC calculations on data stored in SRAM. The CRC engine cannot perform a CRC of data stored in flash memory.

The features include:

- User-definable polynomials up to x^{32} (33 terms)
- DMA support
- Optional automatic byte swap of data in and calculated
- MBig or Little endian support

An n-bit CRC can detect the following types of errors:

- Single-bit errors
- Two-bit errors for block lengths less than $2k$ where k is the order of the longest irreducible factor of the polynomial
- Odd numbers of errors for polynomials with the parity polynomial ($x+1$) as one of its factors (polynomials with an even number of terms)
- Burst errors less than n bits

Overall, all except 1 out of 2^n errors are detected:

- 99.998% for a 16-bit CRC
- 99.9999998% for a 32-bit CRC

22.1 Instances

Instances of the peripheral are listed in *Table 22-1. MAX78000 CRC Instances*.

Table 22-1. MAX78000 CRC Instances

| Instance | Maximum Terms | DMA Support | Big/Little Endian |
|----------|-----------------|-------------|-------------------|
| CRC0 | 33 (2^{32}) | Yes | Yes |

22.2 Usage

The CRC value is typically appended to the end of a data exchange between a transmitter and receiver. The transmitter appends the calculated CRC to the end of the transmission. The receiver independently calculates a CRC on the data it received, the result should be a known constant if the data and CRC were received error-free.

Software will configure the polynomial, starting CRC value, and the endianness of the data. After that the software or the standard DMA will transfer the data to be CRCed in 8-bit, 16-bit, or 32-bit words. Hardware sets `CRC_CTRL.busy` to 1 while the CRC engine is running; software must not read from or write to any of the registers while the CRC is busy. CRC will be updated immediately after each data word. It is up to the software or DMA to keep track of how many words need to be transferred to know when the CRC is ready.

The reset value of `CRC_VAL.value` is 0xFFFF FFFF.

Because the receiving end calculates a new CRC on both the data and received CRC, you must send the received CRC in the correct order, so the highest-order term of the CRC is shifted through the generator first. Because data is typically shifted through the generator LSB first, this means the CRC is reversed bitwise, with the highest-order term of the remainder in the LSB position. Software CRC algorithms typically handle this by calculating everything backwards. They reverse the polynomial and do right shifts on the data. The resulting CRC ends up being bit swapped and in the correct format.

By default the CRC is calculated with the LSB first (`CRC_CTRL.msb_select =0`.) When calculating the CRC on MSB first set `CRC_CTRL.msb_select =1`.

When calculating the CRC on data LSB first, the polynomial should be reversed so that the coefficient of the highest power term is in the LSB position. The largest term x^n is implied (always one) and should be omitted when writing to the **CRC_POLY** register. This is necessary because the polynomial is always one bit larger than the resulting CRC, so a 32-bit CRC has a polynomial with 33 terms ($x^0 \dots x^{32}$).

Table 22-2. Organization of Calculated Result in **CRC_VAL.value**

| CRC_CTRL.msb | CRC_CTRL.byte_swap_out | ORDER |
|---------------------|-------------------------------|--|
| 0 | 0 | The CRC value returned is the raw value |
| 1 | 0 | The CRC value returned is mirrored, but not byte swapped |
| 0 | 1 | The CRC value returned is byte swapped, but not mirrored |
| 1 | 1 | The CRC value returned is mirrored and then byte swapped |

By default, the CRC accelerator does right shifts and calculates the CRC on the LSB of the data first. The CRC can be calculated on the MSB of the data first by setting **CRC_CTRL.msb_select** = 1. To calculate the CRC MSB first, you must left justify the polynomial in the **CRC_POLY** register. The hardware implies the MSB of the polynomial just as it did when shifting the LSB first. The LSB of the polynomial should be set, this defines the length of the CRC. The initial state of the CRC should also be left justified. When the CRC calculation is complete, it is necessary to right shift the CRC to right justify it if the polynomial is less than 32-bits

22.3 Polynomial Generation

The CRC can be configured for any polynomial up to x^{32} (33 terms) by writing to the **CRC_POLY.poly** field. Table 22-3. *Common CRC Polynomials* shows common polynomials and their check constants.

The reset value of **CRC_POLY.poly** that is used by the 32-bit CRC polynomial is the *CRC-32 Ethernet* polynomial. This polynomial is used by Ethernet, PPP, and file compression utilities such as zip or gzip.

The polynomial register, **CRC_POLY.poly**, should only be written when the **CRC_CTRL.busy** field is 0.

Table 22-3. Common CRC Polynomials

| Algorithm | Polynomial Expression | Order | Polynomial | Check |
|-----------------|---|-------|-------------|-------------|
| CRC-32 Ethernet | $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+x^0$ | LSB | 0xEDB8 8320 | 0xDEBB 20E3 |
| CRC-CCITT | $x^{16}+x^{12}+x^5+x^0$ | LSB | 0x0000 8408 | 0x0000 F0B8 |
| CRC-16 | $x^{16}+x^{15}+x^2+x^0$ | LSB | 0x0000 A001 | 0x0000 B001 |
| USB Data | $x^{16}+x^{15}+x^2+x^0$ | MSB | 0x8005 0000 | 0x800D 0000 |
| Parity | x^1+x^0 | LSB | 0x0000 0001 | N/A |

22.4 Calculations Using Software Writes to FIFO

Software can perform CRC calculations by writing directly to **CRC_DATAIN.data**. Each write to that register triggers the computation of a new CRC. Software is responsible for loading the desired number of values into **CRC_DATAIN.data** and reading the result from **CRC_VAL.value**.

Use the following procedure to calculate a CRC value by writing to the FIFO:

1. Set `CRC_CTRL.en` = 0 to disable the peripheral.
2. Configure the fields for the desired input and output data formats:
 - a. `CRC_CTRL.msb`
 - b. `CRC_CTRL.byte_swap_in`
 - c. `CRC_CTRL.byte_swap_out`
3. Configure the CRC polynomial by writing to `CRC_POLY.poly`.
4. Initialize the CRC values by writing to `CRC_VAL.value`.
5. Set `CRC_CTRL.en` = 1 to enable the peripheral.
6. Write a value to be processed to `CRC_DATAIN.data`. Hardware will set `CRC_CTRL.busy` =1.
7. At the end of the calculation, hardware will:
 - a. Clear `CRC_CTRL.busy` =0.
 - b. Load the computed value into `CRC_VAL.value`.
8. Repeat steps 6 and 7 until all the data has been processed.
9. Clear `CRC_CTRL.en` = 0 to disable the peripheral.
10. Read the 32-bit CRC value from `CRC_VAL.value` [31:0] or 16-bit CRC value from `CRC_VAL.value` [15:0].

22.5 Calculations Using DMA

The CRC engine will request new data from the DMA controller while `CRC_CTRL.en` = 1 and `CRC_CTRL.dma_en` = 1 and it is not actively calculating a CRC. The corresponding DMA interrupt should be enabled to signal when all the desired data has been processed and the result is in `CRC_VAL.value`.

Use the following procedure to calculate a CRC value using the DMA:

1. Set `CRC_CTRL.en` = 0 to disable the peripheral.
2. Configure the DMA
3. Set `CRC_CTRL.dma_en` = 1 to enable DMA mode.
4. Configure the fields for the desired input and output data formats:
 - a. `CRC_CTRL.msb`
 - b. `CRC_CTRL.byte_swap_in`
 - c. `CRC_CTRL.byte_swap_out`
5. Configure the CRC polynomial by writing to `CRC_POLY.poly`.
6. Initialize the CRC values by writing to `CRC_VAL.value`.
7. Set `CRC_CTRL.en` = 1 to enable the peripheral. Hardware will clear `CRC_CTRL.busy` =0. The DMA will transparently load data into `CRC_DATAIN.data`.
8. At the end of the DMA operation hardware will:
 - a. Clear `CRC_CTRL.busy` =0.
 - b. Load the new value into `CRC_VAL.value`.
 - c. Generate a DMA IRQ.
9. Clear `CRC_CTRL.en` = 0 to disable the peripheral.
10. Read the 32-bit CRC value from `CRC_VAL.value` or 16-bit CRC value from the lower 15 bits of the `CRC_VAL.value`.

22.6 Interrupt Events

The peripheral does not generate interrupt events.

22.7 Wakeup Events

The peripheral does not generate wakeup events.

22.8 Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in

[Table 22-4: CRC Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 22-4: CRC Register Summary

| Offset | Name | Description |
|--------|----------------------------|-------------------------|
| 0x0000 | CRC_CTRL | CRC Control Register |
| 0x0004 | CRC_DATAIN | CRC Data Input Register |
| 0x0008 | CRC_POLY | CRC Polynomial Register |
| 0x000C | CRC_VAL | CRC Value Register |

22.8.1 Register Details

Table 22-5: CRC Control Register

| CRC Control | | | CRC_CTRL | 0x0000 |
|-------------|---------------|--------|----------|--|
| Bits | Field | Access | Reset | Description |
| 31:17 | - | RO | 0 | Reserved Do not modify |
| 16 | busy | R | 0 | CRC Busy 0: Not busy 1: Busy |
| 15:5 | - | RO | 0 | Reserved Do not modify |
| 4 | byte_swap_out | R/W | 0 | Byte Swap CRC Value Output 0: CRC_VAL.value is not byte swapped 1: CRC_VAL.value is byte swapped |
| 3 | byte_swap_in | R/W | 0 | Byte Swap CRC Data Input 0: CRC_DATAIN.data is processed least significant byte first 1: CRC_DATAIN.data is processed most significant byte first |
| 2 | msb | R/W | 0 | Most Significant Bit Order 0: LSB data first 1: MSB bit data first (mirrored) |
| 1 | dma_en | R/W | 0 | DMA Enable 0: Disabled 1: Enabled. DMA request will be generated if the data output FIFO is full. |
| 0 | en | R/W | 0 | CRC Enable 0: Disabled 1: Enabled |

Table 22-6: CRC Data Input Register

| CRC Data Input | | | CRC_DATAIN | | 0x0004 |
|----------------|-------|--------|------------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | W | 0 | CRC Data Input This register can be written as a byte, halfword, or word. <ul style="list-style-type: none"> • If <i>CRC_CTRL.byte_swap_in</i> = 0 the LSB written will be CRCed first. • If <i>CRC_CTRL.byte_swap_in</i> = 1 the MSB written will be CRCed first. • If <i>CRC_CTRL.msb</i> = 1 each byte will be mirrored as it is CRCed. • If <i>CRC_CTRL.msb</i> = 0 the raw byte value will be CRCed. Do not write to this register if <i>CRC_CTRL.busy</i> = 1 or <i>CRC_CTRL.en</i> = 0. | |

Table 22-7: CRC Polynomial Register

| CRC Polynomial | | | CRC_POLY | | 0x0008 |
|----------------|-------|--------|-------------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | poly | R/W | 0xEDB8 8320 | CRC Polynomial When writing this register: <ul style="list-style-type: none"> • <i>CRC_CTRL.msb</i> = 0: The value is stored as-is. • <i>CRC_CTRL.msb</i> = 1: The value written is mirrored before it is stored in the register When reading this register: <ul style="list-style-type: none"> • <i>CRC_CTRL.msb</i> = 0: The value is read as-is. • <i>CRC_CTRL.msb</i> = 1: The value in the register is mirrored when it is read. | |

Table 22-8: CRC Value Register

| CRC Value | | | CRC_VAL | | 0x000C |
|-----------|-------|--------|---------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | value | R/W | 0 | Current CRC Value Software can write to this register to set the initial state of the accelerator. This register should only be read or written when <i>CRC_CTRL.BUSY</i> =0. When writing this register: <ul style="list-style-type: none"> • <i>CRC_CTRL.msb</i> = 0: The value is stored as-is. • <i>CRC_CTRL.msb</i> = 1: The value written is mirrored before it is stored in the register The data order of the calculated result is shown in Table 22-2. Organization of Calculated Result . | |

23. AES

The provided hardware AES accelerator performs calculations on blocks up to 128 bits.

The features include:

- Supports multiple key sizes:
 - ◆ 128 Bits
 - ◆ 192 Bits
 - ◆ 256 Bits
- DMA support for both RX and TX channels
- Supports multiple key sources:
 - ◆ Encryption using the external AES key
 - ◆ Decryption using the external AES key
 - ◆ Decryption using the locally generated decryption key

23.1 Instances

Instances of the peripheral are listed in *Table 23-1*. Disable the peripheral by clearing *AESn_CTRL.en* = 0 before writing to any register field.

Table 23-1. MAX78000 AES Instances

| Instance | 128-Bit Key | 192-Bit Key | 256-Bit Key | DMA Support |
|----------|-------------|-------------|-------------|-------------|
| AES0 | Yes | Yes | Yes | Yes |

23.2 Encryption of 128-Bit Blocks of Data Using FIFO

AES operations are typically performed on 128-bit of data at a time. The simplest use case is to have software encrypt 128-bit blocks of data. The *AESn_CTRL.start* field is unused in this case.

1. Generate key.
2. Wait for hardware to clear *AESn_STATUS.busy* = 0.
3. Clear *AESn_CTRL.en* = 0 to disable the peripheral.
4. If *AESn_STATUS.input_em* = 0, set *AESn_CTRL.input_flush* = 1 to flush the input FIFO.
5. If *AESn_STATUS.output_em* = 0, set *AESn_CTRL.output_flush* = 1 to flush the output FIFO.
6. Set *AESn_CTRL.key_size* to desired setting
7. Configure *AESn_CTRL.type* = 00 (encryption with external key)
8. If interrupts are desired set *AESn_INTEN.done* = 1 so that an interrupt will trigger at the end of the AES calculation.
9. Set *AESn_CTRL.en* = 1 to enable the peripheral.
10. Write 4×32-bit words of data to *AESn_FIFO.data*. Hardware will start the AES calculation.
11. If *AESn_INTEN.done* = 1, an interrupt will trigger after the AES calculation is complete.
12. If *AESn_INTEN.done* = 0, software should poll until *AESn_STATUS.busy* = 0.
13. Read 4 32-bit words from *AESn_FIFO.data* (least significant word first).
14. Repeat steps 9-13 until all 128-bit blocks have been processed.

23.3 Encryption of 128-Bit Blocks Using DMA

For this example, it is assumed that the DMA both reads and writes data to/from the AES. This is not a requirement. The AES could use DMA on one side and software on the other if that is desired for the application. It is required that for each

DMA_TX_REQ the DMA will write 4×32-bit words of data into the AES. It is required that for each DMA_RX_REQ the DMA will read 4×32-bit words of data out of the AES.

The `AESn_CTRL.start` field is unused in this case. The state of `AESn_STATUS.busy` and `AESn_INTFL.done` are indeterminate during DMA operations. Software must clear `AESn_INTEN.done` = 0 when using the DMA mode. Use the appropriate DMA interrupt instead to determine when the DMA operation is complete and the results can be read from `AESn_FIFO.data`.

Assuming the DMA is continuously filling the Data Input FIFO, the calculations will complete in this many cycles:

- 128-Bit Key - 181 SYS_CLK cycles
- 192-Bit Key - 213 SYS_CLK cycles
- 256-Bit Key - 245 SYS_CLK cycles

The procedure to use DMA encryption/decryption is:

1. Generate key.
2. Initialize the AES RX and TX channels for the DMA controller.
3. Wait for hardware to clear `AESn_STATUS.busy` = 0.
4. Clear `AESn_CTRL.en` = 0 to disable the peripheral.
5. If `AESn_STATUS.input_em` = 0, set `AESn_CTRL.input_flush` = 1 to flush the input FIFO.
6. If `AESn_STATUS.output_em` = 0, set `AESn_CTRL.output_flush` = 1 to flush the output FIFO.
7. Set `AESn_CTRL.key_size` to desired setting
8. Configure `AESn_CTRL.type` = 00 (encryption with external key)
9. Ensure `AESn_INTEN.done` = 0 during DMA operations.
10. Set `AESn_CTRL.en` = 1 to enable the peripheral.
11. The DMA fills the FIFO and hardware begins the AES calculation.
12. When an AES calculation is completed the AES will signal to the DMA that the Data Output FIFO is full and that it should be emptied. If the DMA does not empty the FIFO prior to the next calculation being complete hardware will set `AESn_STATUS.output_full` = 1.

Step 11 and step 12 will be repeated if the DMA has new data to write to the Data Input FIFO.

*Note that the interface from the DMA to the AES only works when the amount of data is a multiple of 128-bits.
For non-multiples of 128-bits the remainder after calculating all of the 128-bit blocks will need to be encrypted manually.*

23.4 Encryption of Blocks Less Than 128-Bits

The AES engine automatically starts a calculation when 128 bits (four writes of 32 bits) occurs. Operations of less than 128-bits will use the start field to initiate the AES calculation.

1. Generate key.
2. Wait for hardware to clear *AESn_STATUS.busy* = 0.
3. Clear *AESn_CTRL.en* = 0 to disable the peripheral.
4. If *AESn_STATUS.input_em* = 0, set *AESn_CTRL.input_flush* = 1 to flush the input FIFO.
5. If *AESn_STATUS.output_em* = 0, set *AESn_CTRL.output_flush* = 1 to flush the output FIFO.
6. Set *AESn_CTRL.key_size* to desired setting
7. Configure *AESn_CTRL.type* = 00 (encryption with external key)
8. If interrupts are desired, set *AESn_INTEN.done* = 1 so that an interrupt will trigger at the end of the AES calculation.
9. Set *AESn_CTRL.en* = 1 to enable the peripheral.
10. Write from one to three 128 bits of data to *AESn_FIFO.data* (least significant word first).
11. Start the calculation manually by setting *AESn_CTRL.start* = 1.
12. If *AESn_INTEN.done* = 1, an interrupt will trigger after the AES calculation is complete.
13. If *AESn_INTEN.done* = 0, software should poll until *AESn_STATUS.busy* = 0.
14. Read 4 32-bit words from *AESn_FIFO.data* (least significant word first).

23.5 Decryption

Decryption of data is very similar to encryption. The only difference is in the setting of *AESn_CTRL.type*. There are 2 possible settings of this field for decryption:

- Decrypt with external key
- Decrypt with internal decryption key

The internal decryption key is generated during an encryption operation. It may be necessary to complete a dummy encryption prior to doing the first decryption to ensure that it has been generated.

23.6 Interrupt Events

The peripheral generates interrupts for the events shown in *Table 23-2. MAX78000 Interrupt Events*. Unless noted otherwise, each instance has its own independent set of interrupts, and higher-level flag and enable fields.

Multiple events may set an interrupt flag and will generate an interrupt if the corresponding interrupt enable is set. The flags must be cleared by software in the ISR.

Table 23-2. MAX78000 Interrupt Events

| Event | Local Interrupt Flag | Local Interrupt Enable |
|--------------------------|------------------------------|------------------------------|
| Data Output FIFO Overrun | <i>AESn_INTFL.ov</i> | <i>AESn_INTEN.ov</i> |
| Key Zero | <i>AESn_INTFL.key_zero</i> | <i>AESn_INTEN.key_zero</i> |
| Key Change | <i>AESn_INTFL.key_change</i> | <i>AESn_INTEN.key_change</i> |
| Calculation Done | <i>AESn_INTFL.done</i> | <i>AESn_INTEN.done</i> |

23.6.1 Data Output FIFO Overrun

When an AES calculation is completed the AES will signal to the DMA that the Data Output FIFO is full and that it should be emptied. If the DMA does not empty the FIFO prior to the next calculation being complete a data output FIFO overrun event occurs and the corresponding local interrupt flag is set.

23.6.2 Key Zero

Attempting a calculation with a key of all zeros generates a key zero event.

23.6.3 Key Change

Writing to any key register while *AESn_STATUS.busy* = 1 generates a key change event.

23.6.4 Calculation Done

The transition of *AESn_STATUS.busy* = 1 to *AESn_STATUS.busy* = 0 generates a calculation done event. The calculation done event interrupt must be disabled when using the DMA.

23.7 Wakeup Events

The peripheral does not generate wakeup events.

23.8 Registers

See *Table 2-4: APB Peripheral Base Address Map* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own independent set of the registers shown in

Table 23-3: AES Register Summary. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See *Table 2-1: Field Access Definitions* for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 23-3: AES Register Summary

| Offset | Name | Description |
|----------|--------------------|-------------------------------|
| [0x0000] | <i>AESn_CTRL</i> | AES Control Register |
| [0x0004] | <i>AESn_STATUS</i> | AES Status Register |
| [0x0008] | <i>AESn_INFL</i> | AES Interrupt Flag Register |
| [0x000C] | <i>AESn_INTEN</i> | AES Interrupt Enable Register |
| [0x0010] | <i>AESn_FIFO</i> | AES Data FIFO |

23.8.1 Register Details

Table 23-4: AES Control Register

| AES Control | | | <i>AESn_CTRL</i> | | 0x0000 |
|-------------|-------|--------|------------------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 31:10 | - | RO | 0 | Reserved Do not modify this field. | |
| 9:8 | type | R/W | 0 | Encryption Type 00: Encryption using the external AES key 01: Decryption using the external AES key 10: Decryption using the locally generated decryption key 11: Reserved | |

| AES Control | | | AESn_CTRL | | 0x0000 |
|-------------|--------------|--------|-----------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 7:6 | key_size | R/W | 0 | Encryption Key Size 00: 128 Bits 01: 192 Bits 10: 256 Bits 11: Reserved | |
| 5 | output_flush | W1 | 0 | Flush Data Output FIFO This field will always read 0. 0: No Action 1: Flush | |
| 4 | input_flush | W1 | 0 | Flush Data Input FIFO This field will always read 0. 0: No Action 1: Flush | |
| 3 | start | W1 | 0 | Start AES Calculation This field forces the start of an AES calculation regardless of the state of the data input FIFO. This allows doing an AES calculation on less than 128-bits of data since normally an AES calculation starts when the data input FIFO is full. This field will always read 0. 0: No Action 1: Start calculation | |
| 2 | dma_tx_en | R/W | 0 | DMA Request To Write Data Input FIFO 0: Disabled 1: Enabled. DMA request will be generated if the data input FIFO is empty. | |
| 1 | dma_rx_en | R/W | 0 | DMA Request To Read Data Output FIFO 0: Disabled 1: Enabled. DMA request will be generated if the data output FIFO is full. | |
| 0 | en | R/W | 0 | AES Enable 0: Disabled 1: Enabled. | |

Table 23-5: AES Status Register

| AES Status | | | AESn_STATUS | | 0x0004 |
|------------|-------------|--------|-------------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | RO | 0 | Reserved Do not modify this field. | |
| 4 | output_full | R | 0 | Output FIFO Full 0: Not full 1: Full | |
| 3 | output_em | R | 0 | Output FIFO Empty 0: Not empty 1: Empty | |
| 2 | input_full | R | 0 | Input FIFO Full 0: Not full 1: Full | |
| 1 | input_em | R | 0 | Input FIFO Empty 0: Not empty 1: Empty | |
| 0 | busy | R | 0 | AES Busy 0: Not busy 1: Busy | |

Table 23-6: AES Interrupt Flag Register

| AES Interrupt Flag | | | AESn_INTFL | | 0x0008 |
|--------------------|------------|--------|------------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved Do not modify this field. | |
| 3 | ov | W1C | 0 | Data Output FIFO Overrun Event Interrupt 0: No event 1: Event occurred | |
| 2 | key_zero | W1C | 0 | Key Zero Event Interrupt 0: No event 1: Event occurred | |
| 1 | key_change | W1C | 0 | Key Change Event Interrupt 0: No event 1: Event occurred | |
| 0 | done | W1C | 0 | Calculation Done Event Interrupt 0: No event 1: Event occurred | |

Table 23-7: AES Interrupt Enable Register

| AES Interrupt Enable | | | AESn_INTEN | | 0x000C |
|----------------------|------------|--------|------------|--|--------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | RO | 0 | Reserved Do not modify this field. | |
| 3 | ov | W1C | 0 | Data Output FIFO Overrun Event Interrupt Enable 0: Enabled 1: Disabled | |
| 2 | key_zero | W1C | 0 | Key Zero Event Interrupt Enable 0: Enabled 1: Disabled | |
| 1 | key_change | W1C | 0 | Key Change Event Interrupt Enable 0: Enabled 1: Disabled | |
| 0 | done | W1C | 0 | Calculation Done Event Interrupt Enable This event interrupt must be disabled when using the DMA. 0: Enabled 1: Disabled | |

Table 23-8: AES FIFO Register

| AES Data | | | AESn_FIFO | | 0x0010 |
|----------|-------|--------|-----------|---|--------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | AES FIFO Writing this register puts data to the data input FIFO. Hardware automatically starts a calculation after 4 words are written to this FIFO. The data should be written with the least significant word first. Reading this register pulls data from the data output FIFO. The least significant word is read first. | |

24. TRNG Engine

The Maxim-supplied Universal Cryptographic Library (UCL), provides a function to generate random numbers intended to meet the requirements of commonly security validations. The entropy from one or more internal noise sources continually feeds a TRNG, the output of which is then processed by software and hardware to generate the number returned by the UCL function. Maxim will work directly with the customer's accredited testing laboratory to provide any information regarding the TRNG that is needed to support the customer's validation requirements.

The general information in this section is provided only for completeness; customers are expected to use the Maxim UCL for the generation of random number.

The TRNG engine can also be used to generate AES keys by firmware using a Hardware Key Derivation Function and using the TRNG as input to the HKDF.

24.1 TRNG Registers

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 24-1: TRNG Register Summary](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 24-1: TRNG Register Summary

| Offset | Register | Name |
|--------|-----------------------------|-----------------------|
| 0x0000 | TRNG_CTRL | TRNG Control Register |
| 0x0004 | TRNG_STATUS | TRNG Status Register |
| 0x0008 | TRNG_DATA | TRNG Data Register |

24.1.1 TRNG Register Details

Table 24-2: TRNG Control Register

| Cryptographic Control | | TRNG_CTRL | | | [0x0000] |
|-----------------------|--------|-----------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 31:2 | - | RO | 0 | Reserved Do not modify | |
| 1 | rnd_ie | R/W | 0 | Random Number Interrupt Enable This bit enables an interrupt to be generated when TRNG_STATUS.rdy = 1. 0: Disabled 1: Enabled | |
| 0 | - | RO | 0 | Reserved Do not modify | |

Table 24-3: TRNG Status Register

| TRNG Status | | TRNG_STATUS | | | [0x0004] |
|-------------|------|-------------|-------|----------------------------------|----------|
| Bits | Name | Access | Reset | Description | |
| 31:1 | - | RO | 0 | Reserved Do not modify | |

| TRNG Status | | | | TRNG_STATUS | [0x0004] |
|-------------|------|--------|-------|---|----------|
| Bits | Name | Access | Reset | Description | |
| 0 | rdy | R | 0 | Random Number Ready This bit is automatically cleared to 0 and a new random number is generated when TRNG_DATA.data is read. 0: Random number generation in progress. The content of TRNG_DATA.data is invalid. 1: TRNG_DATA.data contains new 32-bit random data. An interrupt will be generated if TRNG_CTRL.rnd_ie = 1 | |

Table 24-4: TRNG Data Register

| TRNG Data | | | | TRNG_DATA | [0x0008] |
|-----------|------|--------|-------|--|----------|
| Bits | Name | Access | Reset | Description | |
| 31:0 | data | RO | 0 | TRNG Data The 32-bit random number generated is available here when TRNG_STATUS.rdy = 1. | |

25. Bootloader

The ROM bootloader provides for program loading and verification. The physical interface between the external host and the bootloader is by default the UART.

The secure bootloader (SBL) employs a hash-based message authentication code (HMAC SHA-256) to guarantee both the authenticity and of downloaded program files and the integrity of internal program memory after each reset.

All versions of the bootloader provide the ability to block read/write access to program memory.

Bootloader features:

- Common Functionality of Bootloader and Secure Bootloader
- Checksum verification of ROM image before further ROM execution
- SWD disabled in LOCKED and PERMLOCKED states
- Programmable via UART or SWD interface
- UART operates at 115200 bps
- LOCKED mode disables SWD and disallows any change to flash via bootloader
- Unlock erases all flash and secrets before unlocking SWD
- Optional PERMLOCKED state only allows for program validation Lock

Secure Bootloader (SBL) features:

- Secure HMAC SHA256 with secret key based transactions.
- Trusted secure boot provides automatic program memory verification and authentication before execution after every reset
- Integrity and authentication verification of program memory downloads
- Optional challenge/response gating entry to bootloader

25.1 Instances

The dedicated pins and features of the bootloader are shown [Table 25-1:MAX78000 Bootloader Instances](#).

Table 25-1:MAX78000 Bootloader Instances

| Part Number | Activation Pins | | Bootloader | Secure Bootloader | Secure Boot | Flash Memory Page Size |
|-------------|-----------------|--------|------------|-------------------|-------------|------------------------|
| | UART0 RX | SWDCLK | | | | |
| MAX78000EXG | P0.0 | P0.29 | No | Yes | Yes | 8KB |

25.2 Bootloader Operating States

Each bootloader supports the modes shown in *Table 25-2: Bootloader Operating States and Prompts*. Each bootloader mode has a unique prompt.

Table 25-2: Bootloader Operating States and Prompts

| State | Bootloader | Secure Bootloader | Recognized Commands | Prompt |
|-------------------|------------|-------------------|--|---|
| <i>UNLOCKED</i> | Yes | Yes | All Commands U/L/P | “ULDR> “ <0x55> <0x4C> <0x44> <0x52> <0x3E> <0x20> |
| <i>LOCKED</i> | Yes | Yes | Only L/P | “LLDR> “ <0x4C> <0x4C> <0x44> <0x52> <0x3E> <0x20> |
| <i>PERMLOCKED</i> | Yes | Yes | Only P | “PLLDR> “ <0x50> <0x4C> <0x4C> <0x44> <0x52> <0x3E> <0x20> |
| <i>CHALLENGE</i> | No | Yes | <i>GC – Get Challenge</i> <i>SR – Send Response</i> | “CR> “ <0x43> <0x52> <0x3E> <0x20> |
| <i>APPVERIFY</i> | No | Yes | N/A | N/A |

The *LOCK – Lock Device* and *PLOCK – Permanent Lock* commands do not change the bootloader prompt or take effect until the bootloader is reset.

25.2.1 UNLOCKED

The UNLOCKED state provides access to load secure keys and configuration information. Program loading, verification, and status is available in the UNLOCKED state. The SWD interface is available for use.

Transitioning from the LOCKED to UNLOCKED states erases all program memory. In the SBL it also clears the challenge/response and application keys stored by the SBL.

The challenge and application keys can be erased by executing the Unlock command while in the UNLOCKED state and resetting the device. This eliminates the need to transition through the LOCKED state.

25.2.2 LOCKED

The LOCKED state disables access to program memory other than to verify it. The application and challenge response keys cannot be changed without first changing to the UNLOCKED state.

For the SBL, If the optional challenge key is activated the bootloader will start in the CHALLENGE state. Successfully completing the challenge/response will transition to the previous PERMLOCKED or LOCKED state.

The application key should be configured before executing the *LOCK – Lock Device* command.

25.2.3 PERMLOCKED

The PERMLOCKED state disables access to program memory other than to verify it with a simple SHA256 hash. The commands available in PERMLOCKED state are shown in *Table 25-3: PERMLOCK Command Summary*.

Table 25-3: PERMLOCK Command Summary

| Command |
|-------------------------|
| <i>H – Check Device</i> |
| <i>I – Get ID</i> |

For the SBL, If the optional challenge key is activated the bootloader will start in the CHALLENGE state. Successfully completing the challenge/response will transition to the previous PERMLOCKED state.

The application key should be configured before executing the [PLOCK – Permanent Lock](#) command.

25.2.4 CHALLENGE (Secure Bootloader Only)

The CHALLENGE state provides an extra layer of security by requiring the host to authenticate itself using the HMAC SHA-256 key before executing any bootloader commands. If enabled, the device enters CHALLENGE mode following a reset if the external bootloader pins are active. CHALLENGE mode can be identified by the “CR>” prompt.

In CHALLENGE mode the host first requests a 128-bit random number (the challenge) from the bootloader. The host encrypts the challenge using the mutually known HMAC SHA256 key and sends it (the response) back to bootloader. The correct response transitions from CHALLENGE to the previous state of the bootloader. An incorrect response keeps the bootloader in the CHALLENGE state. The host must request a new challenge and send a response based on the new challenge. There is no limit to the number of challenge attempts.

25.2.5 APPVERIFY (Secure Bootloader only)

The APPVERIFY is an internal state that describes how the SBL verifies the integrity of program memory using a secret-key HMAC SHA-256 hash. It is not directly accessible by the SBL command set. The SBL performs an APPVERIFY:

- When executing a secure boot
- Immediately before executing the SBL [LOCK – Lock Device](#) command
- Immediately before executing the SBL [PLOCK – Permanent Lock](#) command

Failure of the APPVERIFY process during a secure boot indicates corrupted or tampered program memory and disables code execution. If the SBL is in the LOCKED state it can transition to the UNLOCKED state, erasing the program memory and keys so the device can be reprogrammed. There is no recovery from a secure boot failure in the PERMLOCKED state and the device must be discarded.

Follow this procedure to initialize the SBL for the APPVERIFY:

1. The host creates the Motorola SREC file as described in [Creating the Motorola SREC File](#).
2. The host activates the SBL as described in the [Bootloader Activation](#) section.
3. Ensure the device is in the UNLOCKED state.
4. Execute the WL command with the length value calculated in step 1
5. Execute the L command to load the Motorola SREC file
6. Execute the V command to verify the Motorola SREC file was correctly loaded.
7. Execute the LK command to load the HMAC SHA-256 secret key.
8. Execute the VK command to verify the HMAC SHA-256 secret key was correctly loaded.
9. Execute the AK command. Device will automatically verify program memory on all subsequent resets and attempts to execute the Lock and Plock commands.

25.3 Creating the Motorola SREC File

The Motorola SREC file must include the program bytes and the MAC required for the APPVERIFY process. Address records must be 32-bit aligned and the length of each line must be a multiple of 4 bytes. Any unused memory locations within the program must be defined with a constant value.

The information here is presented for completeness; Maxim Integrated can provide customers with a complete toolset for generating a Motorola SREC file that meets the SBL requirements.

Note: The length of the Motorola SREC file will not be the same as code length used for the WL command, as explained below.

The procedure for generating the SREC file is:

1. Define the 128-bit HMAC secret key
2. Generate binary image
3. Pad the binary image with constant value to next 32-byte boundary. The address of the last pad byte is the code length argument for the WL command.
4. Calculate the 32-byte HMAC SHA256 using the secret key over the length of the padded binary image
5. Append 32-byte HMAC SHA256 to the binary image, after the last pad byte
6. Generate Motorola SREC file

25.4 Bootloader Activation

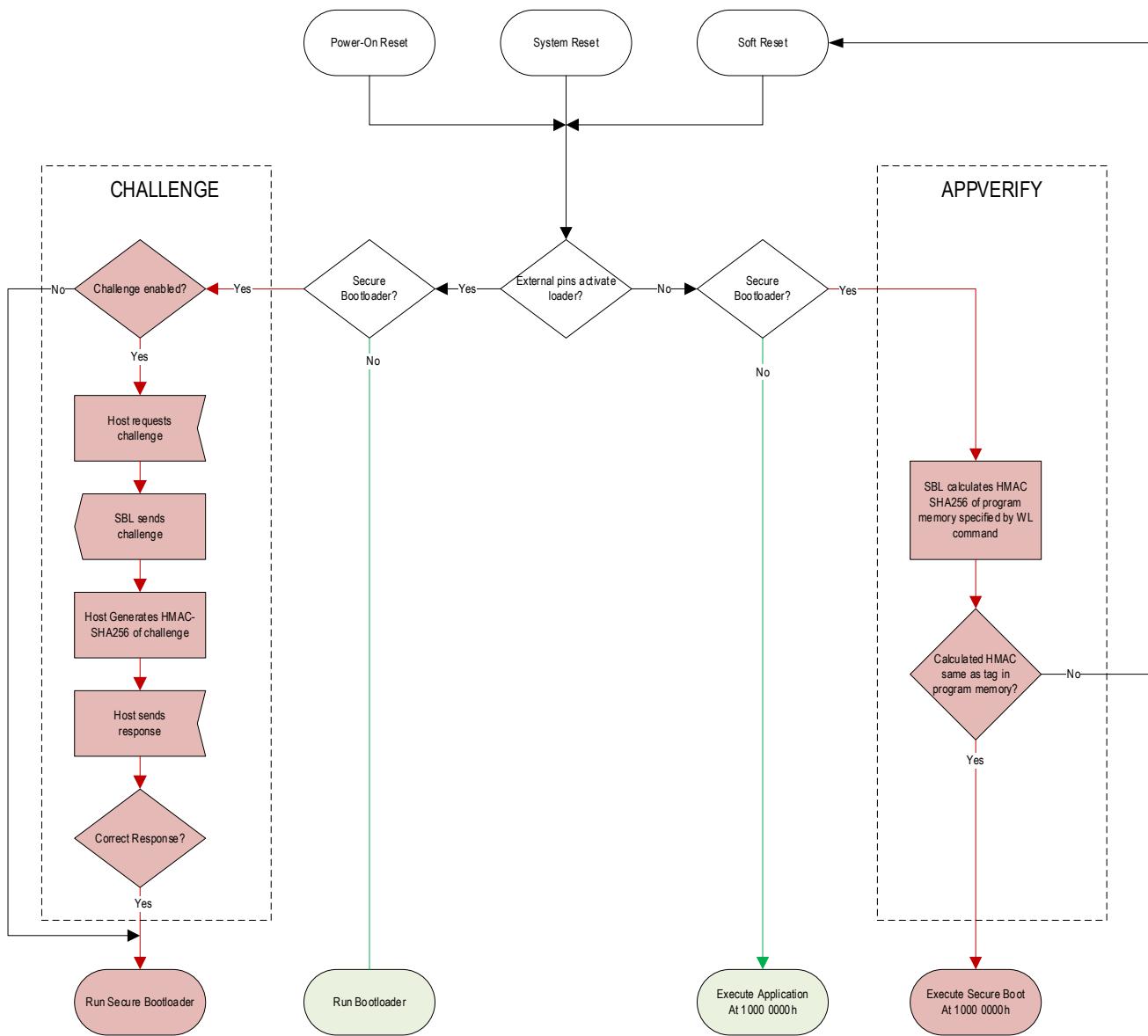
Perform the following sequence to activate the bootloader:

1. Host asserts the UART0 RX pin and SWDCLK pins low as shown in [Table 25-1:MAX78000 Bootloader Instances](#).
2. Host asserts RSTN pin low.
3. Host deasserts the RSTN pin
4. Bootloader samples the UART0 RX and SWDCLK pins. If they are both low the hardware will activate the bootloader.
5. Bootloader performs its system initialization and configures the bootloader for 115200 bps.
6. The bootloader outputs the status prompt on the UART0 TX pin. The prompt is unique for each bootloader state as shown in [Table 25-2: Bootloader Operating States and Prompts](#).
7. The host releases the UART0 RX and SWDCLK pins once the host confirms the correct bootloader prompt.
8. Host begins bootloader session by sending commands on the UART0 RX pin.

25.5 Bootloader

The bootloader is invoked following a reset when the bootloader activation. The flow chart of the operation following a reset of the device is shown [Figure 25-1: MAX78000 Combined Bootloader Flow](#). Features exclusive to the SBL are highlighted in red.

Figure 25-1: MAX78000 Combined Bootloader Flow



25.6 Secure Bootloader

The secure version of the bootloader provides additional features for secure and authenticated loading. These features are highlighted in *Figure 25-1: MAX78000 Combined Bootloader Flow*.

25.6.1 Secure Boot

The SBL performs a *Secure Boot* by entering the APPVERIFY state following reset in which the bootloader activation pins are not active. Failure of the secure boot will place the device in a reset loop to prevent execution of corrupted or tampered code. The SBL also enters APPVERIFY before completing the *LOCK – Lock Device* or *PLOCK – Permanent Lock* commands to ensure that the correct program memory and application key are loaded.

Failure of the secure boot will force the device into a continual reset state.

25.6.2 Secure Challenge/Response Authentication

The optional secure Challenge/Response authentication provides an extra layer of security by requiring the host to authenticate itself using the mutual HMAC SHA-256 key before executing any bootloader commands. If the challenge key is activated, the device enters CHALLENGE mode following a reset if the external bootloader pins are active. The bootloader will display the CHALLENGE mode prompt shown in [Table 25-2: Bootloader Operating States and Prompts](#).

Only two commands are available in the CHALLENGE state:

Table 25-4: CHALLENGE Command Summary

| Command |
|------------------------------------|
| GC – Get Challenge |
| SR – Send Response |

The host first requests a 128-bit random number (the challenge) from the bootloader. The host encrypts the challenge using the HMAC SHA256 key (the response) and sends it back to bootloader. The correct response transitions the bootloader from CHALLENGE mode to the LOCKED or PERMLOCKED states, depending on the last state of the bootloader.

Follow this procedure to enable the Challenge/Response feature in the UNLOCKED state:

1. Host generates the challenge/response HMAC SHA-256 secret key.
2. Host executes the LK command to load the challenge/response secret key. The key is sent in plaintext and should be done in a secure environment.
3. Host executes the VK command to verify the challenge/response secret key was correctly loaded.

The Challenge/Response will be required after the next device reset. It does not affect current operation until the next reset.

Follow this procedure to successfully perform the Challenge/Response:

1. Host executes the GC command.
2. Bootloader generates a 128-bit challenge and sends it to the host.
3. Host performs HMAC SHA-256 of the bootloader challenge to create the response.
4. The host executes the SR command with the calculated response. The SR command must be the first command sent to the bootloader after a GC command.

A correct response will return the prompt of the last bootloader state. An incorrect response will return an error message and the challenge response prompt again. The host can perform steps 1-3 again to request another challenge from the bootloader. There is no limit on the number of challenge/response attempts.

Following a successful response, the bootloader will return the prompt appropriate to the last state of the SBL.

25.7 Command Protocol

The bootloader presents a mode-specific prompt based on the current state of the loader as shown as in [Table 25-2: Bootloader Operating States and Prompts](#). The general format of commands is the ASCII character(s) of the command, followed by a <CR><LF> which is hexadecimal <0x0D><0x0A>. Commands with arguments always have a space (0x20) between the command mnemonic and the argument.

Commands arguments that are files always have the length specified in the file, so it is not necessary to follow the file with a <0x0D><0x0A>.

In general, arguments not related to security commands are prefixed with “0x” to denote hexadecimal input. Arguments for security commands in general are not prefixed with “0x”.

Always refer to the command description for the required format of the command.

25.8 General Commands

Table 25-5: MAX78000 General Command Summary

| Command |
|-------------------------------|
| <i>L – Load</i> |
| <i>P – Page Erase</i> |
| <i>V – Verify</i> |
| <i>LOCK – Lock Device</i> |
| <i>PLOCK – Permanent Lock</i> |
| <i>UNLOCK – Unlock Device</i> |
| <i>H – Check Device</i> |
| <i>I – Get ID</i> |
| <i>S – Status</i> |
| <i>Q – Quit</i> |

25.8.1 General Command Details

Table 25-6: L - Load

| L - Load | Load Motorola SREC File into Program Memory |
|----------------------|---|
| Description | Load a Motorola SREC formatted file into Flash program memory. See Creating the Motorola SREC File for the details of the format required for the SBL. After typing the L command the bootloader will respond with “Ready to load SREC”, then transmit the file. The end of the file is detected automatically, so there is no need to send <0x0D><0x0A> at the end. The length reported by the success response the padded image plus the 32-bytes of the HMAC; this is different than the length used for the WL command. |
| Modes | U |
| Command | L<0x0D><0x0A> Ready to load SREC<0x0D><0x0A> [Motorola SREC File] |
| Response: Success | Load success, image loaded with the following parameters:<0x0D><0x0A> Base address: 0xnnnnnnnn<0x0D><0x0A> Length: 0xnnnnnnnn<0x0D><0x0A> |
| Response: Failure | Load failed.<0x0D><0x0A> |

Table 25-7: P – Page Erase

| P – Page Erase | Erase Page of Flash Program Memory |
|-------------------|---|
| Description | Erases the page of memory associated with the 32-bit input address. Addresses must be aligned on the device-specific page boundaries. |
| Modes | U |
| Command | P 0xnnnnnnnn<0x0D><0x0A> |
| Response: Success | Erase Page Address: 0xnnnnnnnn<0x0D><0x0A>OK<0x0D><0x0A> |
| Response: Failure | Bad page address input<0x0D><0x0A> or Erase failed<0x0D><0x0A> or Invalid Page Address: 0xnnnnnnnn<0x0D><0x0A> |

Table 25-8: V – Verify

| V – Verify | Verify Flash Program Memory Against Motorola SREC File |
|-------------------|--|
| Description | Verifies contents of Flash program memory against a Motorola SREC file. |
| Modes | U |
| Command | V<0x0D><0x0A> Ready to verify SREC<0x0D><0x0A> [Motorola SREC File] |
| Response: Success | Verify success, image verified with the following parameters: <0x0D><0x0A> Base address: 0xnnnnnnnn<0x0D><0x0A> Length: 0xnnnnnnnn<0x0D><0x0A> |
| Response: Failure | Verify failed.<0x0D><0x0A> |

Table 25-9: LOCK – Lock Device

| LOCK – Lock Device | Lock Device |
|--------------------|---|
| Description | Locks the device and disables SWD on the next device reset. See <i>LOCKED</i> section for a detailed description of this command. The effects of the Lock command do not take effect until the next time the device is reset. The bootloader will continue to display the locked prompt but the <i>S – Status</i> command will show the Locked mode is active. The Lock command should be followed by the Q command (which generates a device reset) for the Lock command to take effect. The SBL performs an APPVERIFY check before executing the Lock command. Failure of the Lock command means that the APPVERIFY check failed. |
| Modes | U |
| Command | LOCK<0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Failed<0x0D><0x0A> |

Table 25-10: PLOCK – Permanent Lock

| PLOCK – Permanent Lock | Permanently Lock Device |
|------------------------|---|
| Description | Permanently locks the device if the argument matches the device ID. The effects of the Plock command do not take effect until the next time the device is reset. The bootloader will continue to display the LOCKED or UNLOCKED state prompt but the <i>S – Status</i> command will show the LOCKED or UNLOCKED state is active. The Lock command should be followed by the Q command (which generates a device reset) for the Lock command to take effect. The SBL performs an APPVERIFY check before executing the Plock command. Failure of the Plock command means that the APPVERIFY check failed. |
| Modes | U/L |
| Command | PLOCK <USN><0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Failed<0x0D><0x0A> |

Table 25-11: UNLOCK – Unlock Device

| UNLOCK – Unlock Device | Unlock Device |
|-------------------------------|--|
| Description | Changes bootloader state to UNLOCKED if in LOCKED state. Erases all program memory and all bootloader keys. The SWD interface is re-enabled. |
| Modes | U/L |
| Command | UNLOCK<0x0D><0x0A> |
| Response: Success | None. The device automatically resets itself and the bootloader will display the UNLOCKED mode prompt the next time the bootloader is activated. |
| Response: Failure | None. |

Table 25-12: H – Check Device

| H – Check Device | Perform SHA-256 Hash Over Memory Range |
|-------------------------|--|
| Description | Performs a simple SHA-256 (not HMAC SHA-256) hash of bytes starting at 32-bit address 0xnnnnnnnn to 0xmmmmmmmm. Minimum hash input size is 512 bytes. Function returns 32-byte hash value. |
| Modes | U/L/P |
| Command | H 0xnnnnnnnn 0xmmmmmmmm<0x0D><0x0A> |
| Response: Success | > yy<0x0D><0x0A> |
| Response: Failure | <0x0D><0x0A> |

Table 25-13: I – Get ID

| I – Get ID | Read Universal Serial Number |
|-------------------|--|
| Description | Returns the 13-byte unique USN of the device. |
| Modes | U/L/P |
| Command | I<0x0D><0x0A> USN: nnnnnnnnnnnnnnnnnnnnnnnnnn<0x0D><0x0A> |
| Response: Success | None |
| Response: Failure | None |

Table 25-14: S – Status

| S – Status | Read Device Status |
|-------------------|---|
| Description | <p>Returns the state of the loader and the application key and challenge key features. This will change during the same session when the command is executed. unlike the prompt which only changes after reset:</p> <p>The Lock <response> is: “Inactive” if the device is in the unlocked state. “Active” if the device is in the locked or permanent lock state.</p> <p>The PLock <response> is: “Inactive” if the device is in the unlocked or locked state. “Active” if the device is in the permanent lock state.</p> <p>In addition, the SBL will display:</p> <p>The Application Length <response> is: “Not Set” if the Write Code Length command has not previously loaded a non-zero value “0xnnnnnnnn” which is the previously entered value using the Write Code Length command.</p> <p>The Application Key <response> is: “None Inactive” if no application key has been loaded using the LK command. “Loaded Inactive” if the application key has been loaded but the application key feature has not been activated by the AK command “Loaded Active” If the application key has been loaded and the application key feature has been activated.</p> <p>The Challenge Key <response> is: “None Inactive” if no challenge key has been loaded using the LK command. “Loaded Inactive” if the challenge key has been loaded but the challenge key feature has not been activated by the AK command “Loaded Active” if the challenge key has been loaded and the challenge key feature has been activated.</p> |
| Modes | U |
| Command | <pre>S<0xD><0xA> Status<0xD><0xA> Lock: <response><0xD><0xA> PLock: <response><0xD><0xA> Application Length: <response><0xD><0xA> Application Key: <response><0xD><0xA> Challenge Key: <response><0xD><0xA></pre> |
| Response: Success | None. |
| Response: Failure | None. |

Table 25-15: Q – Quit

| Q – Quit | Quit Bootloader Session |
|-------------------|---|
| Description | Terminates the bootloader session and forces a reset of the device. |
| Modes | U/L/P |
| Command | Q<0xD><0xA> |
| Response: Success | None |
| Response: Failure | None |

25.9 Secure Commands

Table 25-16: MAX78000 Secure Command Summary

| Command |
|--------------------------------------|
| <i>LK – Load Application Key</i> |
| <i>LC – Load Challenge Key</i> |
| <i>VK – Verify Application Key</i> |
| <i>VC – Verify Challenge Key</i> |
| <i>AK – Activate Application Key</i> |
| <i>AC – Activate Challenge</i> |
| <i>WL – Write Code Length</i> |

25.9.1 Secure Command Details

Table 25-17: LK – Load Application Key

| LK – Load Application Key | Load Application HMAC-SHA256 Key |
|---------------------------|--|
| Description | Write 128-bit HMAC secret key to non-volatile memory. |
| Modes | U |
| Command | LK yyyy...yy<0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Bad key input<0x0D><0x0A> or Key already loaded<0x0D><0x0A> |

Table 25-18: LK – Load Challenge Key

| LC – Load Challenge Key | Load Challenge Key |
|-------------------------|--|
| Description | Write 128-bit challenge key to non-volatile memory. |
| Modes | U |
| Command | LC yyyy...yy<0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Bad key input<0x0D><0x0A> or Key already loaded<0x0D><0x0A> |

Table 25-19: VK – Verify Application Key

| VK – Verify Application Key | VK – Verify Application Key |
|-----------------------------|---|
| Description | Verify the Application Key against a value provided by the Host. |
| Modes | U |
| Command | VK yyyy...yy<0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Bad key input<0x0D><0x0A> or Error, no key loaded<0x0D><0x0A> or Key Mismatch<0x0D><0x0A> |

Table 25-20: VC – Verify Challenge Key

| VC – Verify Challenge Key | VC – Verify Challenge Key |
|---------------------------|---|
| Description | Verify the Challenge Key against a value provided by the Host. |
| Modes | U |
| Command | VC yyy<0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Bad key input<0x0D><0x0A> or Error, no key loaded<0x0D><0x0A> or Key Mismatch<0x0D><0x0A> |

Table 25-21: AK – Activate Application Key

| Activate Application Key | |
|---------------------------------|--|
| Description | Activate application key. All application software loads must be encrypted with the application key. The UNLOCK command deactivates the application key. |
| Modes | U |
| Command | AK<0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Key activate failed<0x0D><0x0A> or Error, no key loaded<0x0D><0x0A> |

Table 25-22: AC – Activate Challenge Key

| AC – Activate Challenge Mode | Activate Challenge Mode |
|------------------------------|--|
| Description | Activate CHALLENGE mode. All subsequent bootloader sessions in LOCKED or PERMLOCKED states will start in CHALLENGE mode. The “Key activate failed” response indicates the device has already activated the challenge/response feature. The host should use the SBL to re-enter the UNLOCKED state to deactivate the challenge mode and reenter the keys and activate the challenge mode again. |
| Modes | U |
| Command | AC<0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Key activate failed<0x0D><0x0A> or Error, no key loaded<0x0D><0x0A> |

Table 25-23: WL – Write Code Length

| WL – Write Code Length | Write Code Length |
|------------------------|---|
| Description | Write the length of the application code in bytes as calculated in Creating the Motorola SREC File . The “Write length failed” response indicates the WL command has already been performed. The host should use the SBL to re-enter the UNLOCKED state to clear the WL value and repeat the command. |
| Modes | U |
| Command | WL 0xnnnnnnnn<0x0D><0x0A> |
| Response: Success | Length set to: 0xnnnnnnnn<0x0D><0x0A> |
| Response: Failure | Bad length input<0x0D><0x0A> OR Write length failed<0x0D><0x0A> |

25.10 Challenge/Response Commands

Table 25-24: MAX78000 Challenge/Response Command Summary

| Register Name |
|---------------------------|
| <i>GC – Get Challenge</i> |
| <i>SR – Send Response</i> |

25.10.1 Challenge/Response Command Details

Table 25-25: GC – Get Challenge

| GC – Get Challenge | Get Challenge |
|--------------------|---|
| Description | Bootloader generates a 16-byte hexadecimal ASCII challenge and transmits it to host. The challenge is sent MSB first. |
| Modes | L/P |
| Command | GC<0x0D><0x0A> |
| Response: Success | 0123456789ABCDEF0123456789ABCDEF<0x0D><0x0A> |
| Response: Failure | None |

Table 25-26: SR – Send Response

| SR – Send Response | Send Response |
|--------------------|--|
| Description | Host calculates HMAC SHA-256 on the 16-byte challenge and sends the 32-byte hexadecimal ASCII response to SBL. The response is sent MSB first. |
| Modes | L/P |
| Command | SR 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF<0x0D><0x0A> |
| Response: Success | OK<0x0D><0x0A> |
| Response: Failure | Bad response input<0x0D><0x0A> Or Verification failed<0x0D><0x0A> Or Error, request challenge<0x0D><0x0A> |

26. Convolutional Neural Network (CNN)

Important Note: This is a draft of the CNN user guide chapter for the MAX78000. Regular updates are planned until the chapter is complete and peer reviewed. Currently the chapter consists of an outline and the Registers and Fields for the CNN.

26.1 Overview

The CNN accelerator consists of 64 parallel processors with 512KB of SRAM-based storage. Each processor includes a pooling unit and a convolutional engine with dedicated weight memory. Four processors share one data memory. These are further organized into groups of 16 processors that share common controls. A group of 16 processors operates as a slave to another group or independently. Data is read from SRAM associated with each processor and written to any data memory located within the accelerator. Any given processor has visibility of its dedicated weight memory and to the data memory instance it shares with the three others.

The features of the CNN accelerator include:

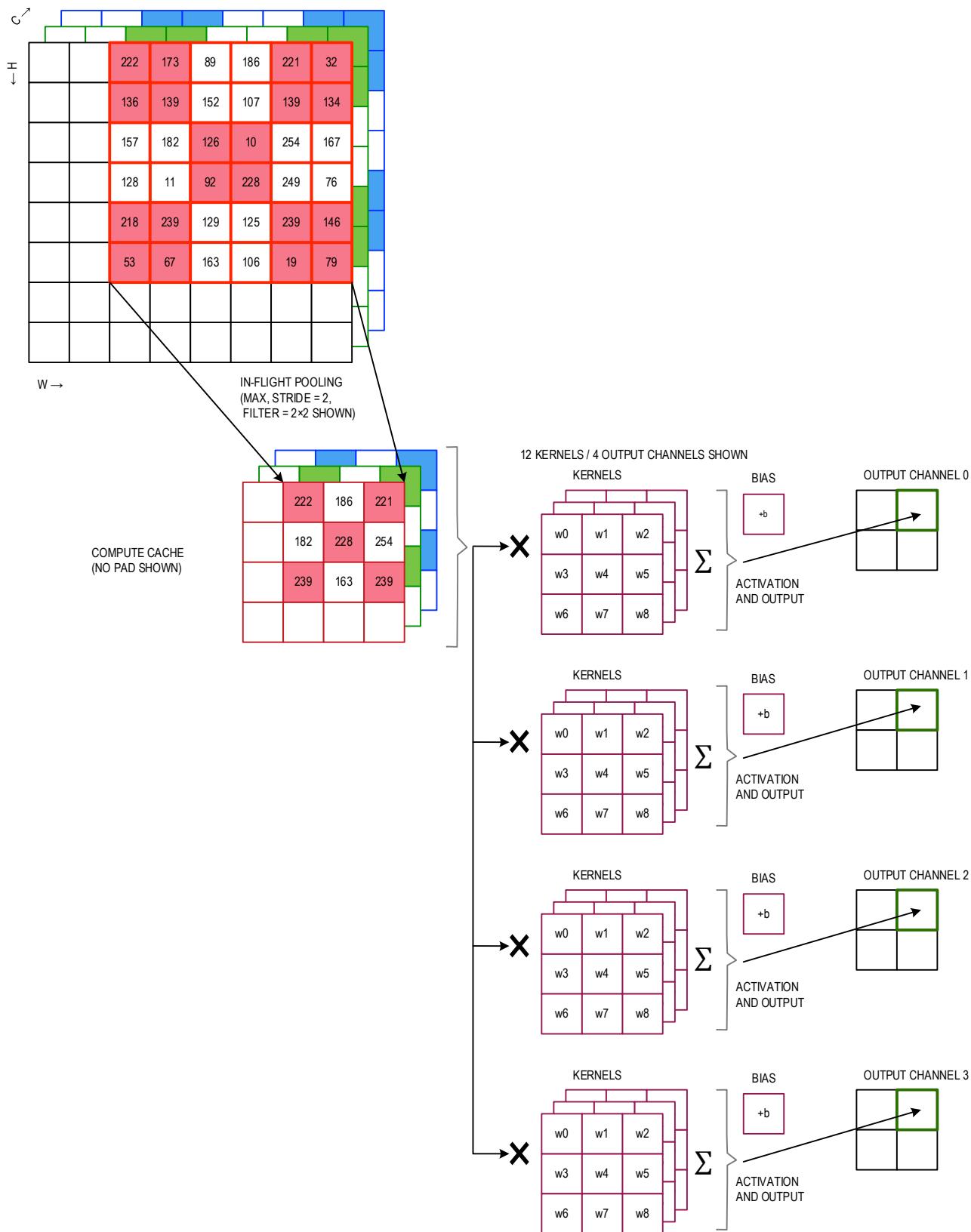
- 512KB SRAM data storage
 - ◆ Configured as 8Kx8-bit integers x64 channels or 32Kx8-bit integers x4 channels for input layers
 - ◆ Hardware load and unload assist
- 64 parallel physical channel processors
 - ◆ Organized as 4x16 processors
 - ◆ 8-bit integer data path with option for 32-bit integers on the output layer
 - ◆ Per-channel processor enable/disable
 - ◆ Expandable to 1024 parallel logical channel processors
- 1x1 or 3x3 2D kernel sizes
- Configurable 1D kernel size to 1x9
- Full resolution sum-of-product arithmetic for 1024 8-bit integer channels
- Operating frequency up to 50MHz
- Nominal 1 output channel per clock, maximum 4 output channels per clock (passthru)
- Configurable input layer image size
 - ◆ 32K pixels, 16 channels, non-streaming
 - ◆ 8K pixels, 4 channels, non-streaming
 - ◆ 1024 x 1024 pixels, 4 channels, streaming
- Hidden layers
 - ◆ Up to 8K 8-bit integer data per channel, x64 channels, non-streaming
 - ◆ 8K bytes can be split equally across 1 to 16 logical channels, non-streaming
 - ◆ 1M 8-bit integer data per channel, x64 channels, streaming
 - ◆ 1M bytes can be split equally across eight layers, streaming
- Optional interrupt on CNN completion
- User-accessible BIST on all SRAM storage
- User-accessible zeroization of all SRAM storage
- Single-step operation with full data SRAM access for CNN operation debug
- Flexible power management
 - ◆ Independent x16 processor supply enables
 - ◆ Independent x16 processor mask retention enables
 - ◆ Independent x16 data path clock enables
 - ◆ Active Arm peripheral bus clock gating with per x16 processor override
 - ◆ CNN clock frequency scaling (divide by 2, 4, 8, 16)
 - ◆ Chip-level voltage control for performance power optimization
- Configurable weight storage
 - ◆ SRAM-based weight storage with selectable data retention
 - ◆ Configurable from 442K 8-bit integer weights to 3.456M 1-bit logical weights
 - Organized as 768X9X64 8-bit integer weights to 768x72x64 1-bit logical weights
 - Can be configured on a per-layer basis
 - ◆ Programmable per x16 processor weight RAM start address, start pointer, and mask count
 - ◆ Optional weight load hardware assist for packed weight storage
- 32 independently configurable layer groups
 - ◆ Each group can contain element-wise, and/or pooling, and/or convolution operations for a minimum of 32 and a maximum of 96 layers
 - ◆ Processor and mask enables (16 channels)
 - ◆ Input data format
 - ◆ Per-layer data streaming
 - Stream start - relative to prior stream
 - Dual-stream processing delay counters - 1 column, 1 row delta counter
 - Data SRAM circular buffer size
 - ◆ Input data size (row, column)
 - ◆ Row and column padding 0 to 4 bytes

- ◆ Number of input channels 1 to 1024
- ◆ Kernel bit width size (1, 2, 4, 8)
- ◆ Kernel SRAM start pointer and count
- ◆ Inflight input image pooling
 - Pool mode - none, maximum or average
 - Pool size - 1x1 to 16x16
- ◆ Stride - 1 row, 1 column to 4 rows, 4 columns
- ◆ Data SRAM read pointer base address
- ◆ Data SRAM write pointer configuration
 - Base address
 - Independent offsets for output channel storage in SRAM
 - Programmable stride increment offset
- ◆ Bias - 2048 8-bit integers with option for 512 32-bit integers
- ◆ Pre-activation output scaling from 0 to 8 bits
- ◆ Output activation - none, ReLU, absolute value
- ◆ Passthru - 8-bit or 32-bit integers
- ◆ Element-wise operations (add, subtract, xor, or) with optional convolution - up to 16 elements
- ◆ Deconvolution (upsampling)
- ◆ Flattening for MLP processing
- ◆ 1x1 convolution

A typical CNN operation consists of pooling followed by a convolution. While these are traditionally expressed as separate layers, pooling can be done “in-flight” on the MAX78000 for greater efficiency.

To minimize data movement, the accelerator is optimized for convolutions with in-flight pooling on a sequence of layers. The MAX78000 also supports in-flight element-wise operations, pass-through layers and 1-D convolutions without element-wise operations. [Figure 26-1: CNN Overview](#) shows a high-level diagram of the MAX78000’s Convolutional Neural Network flow.

Figure 26-1: CNN Overview



Preliminary Draft 08/31/2020

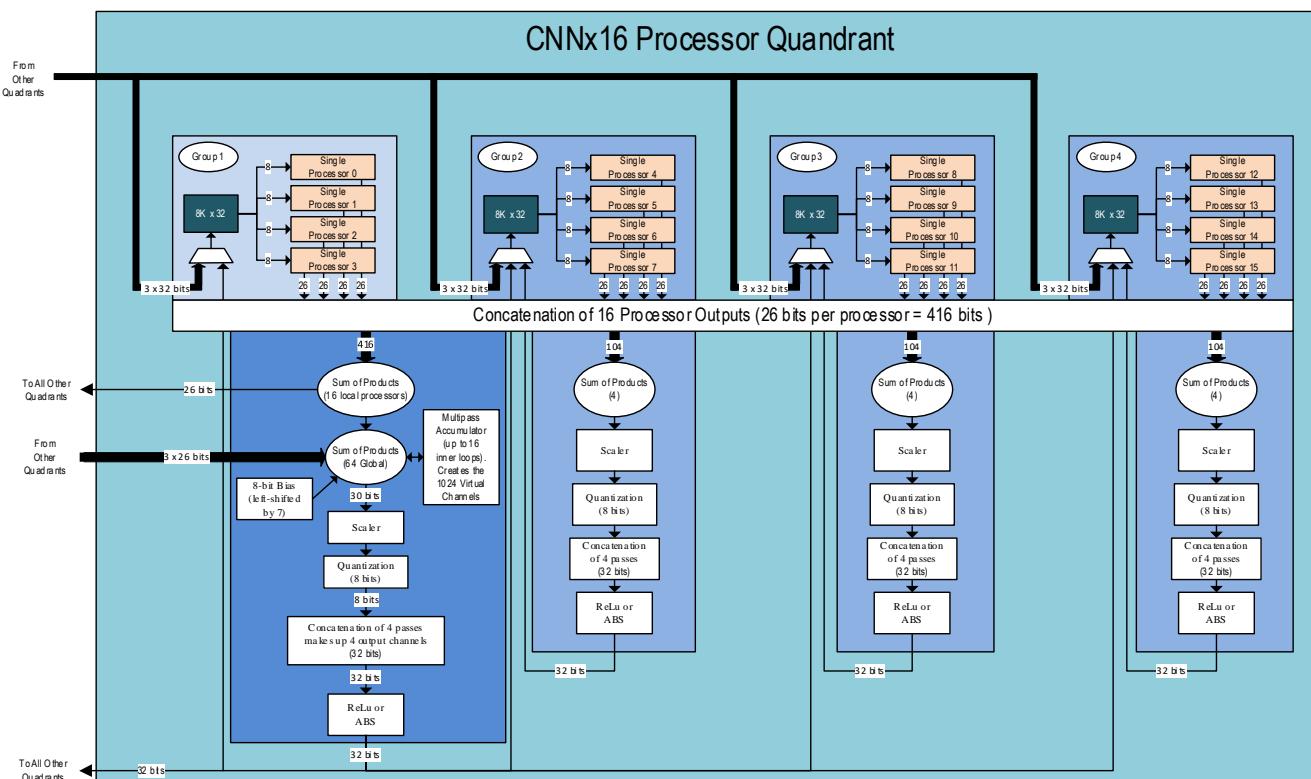
26.2 Instances

There is one instance of the CNN. The CNN contains four CNNx16 processor quadrants, generically referred to as CNNx16_n. Each CNNx16_n processor quadrant contains sixteen processors grouped in four groups of four. These groups, labeled Group 1 to Group 4 in the CNNx16_Processor Quadrant block diagram below and are referred to as CNNx16_n_q0 to CNNx16_n_q3 in the documentation.

26.2.1 Block Diagram

Figure 26-2 shows a block diagram of a single CNNx16 Processor Quadrant. The MAX78000 includes four CNNx16 processor quadrants. The processor groups are labeled in *Figure 26-2* as Group 1 to Group 4.

Figure 26-2: CNNx16_n Processor Quadrant Block Diagram



26.3 Memory Configuration

The CNN includes four CNNx16 processor arrays and each processor array includes 512KB of SRAM, 192KB of Mask RAM (MRAM), 32KB of Bias RAM (BRAM) and 192KB of Tornado RAM (TRAM). *Figure 26-3* shows the APB mapping for the CNN and the Quad CNNx16n processor arrays. *Table 26-1* shows the memory address range for each type of available memory in each CNNx16n processor array.

Figure 26-3: CNN Global and Quad CNNx16n Processor Array APB Memory Map

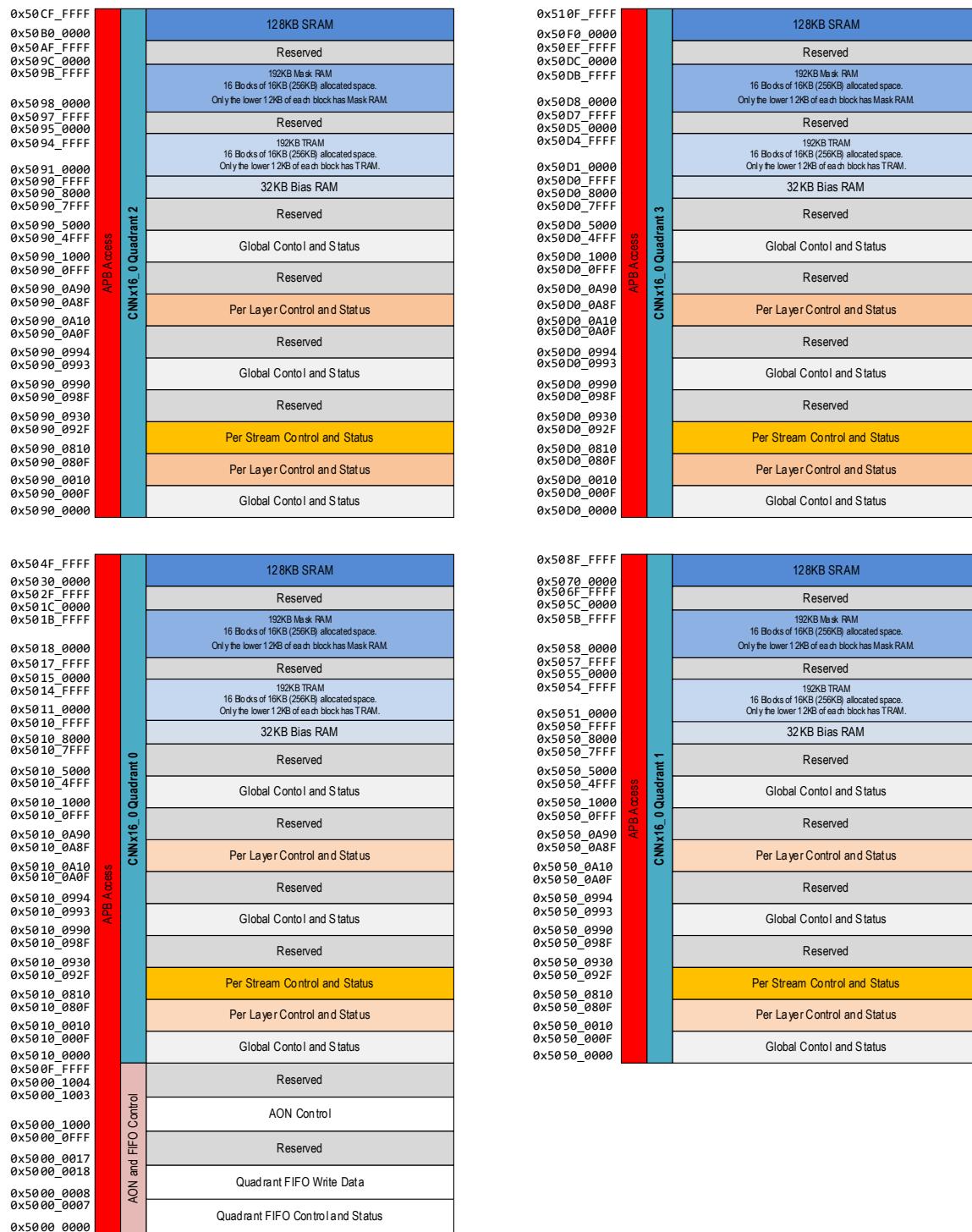


Table 26-1: CNN Memory Configuration (APB Accessible) for the Quad CNNx16n

| CNN Quadrant# | Memory Type | Size | Start Address | End Address |
|---------------|-------------|-------|---------------|--------------|
| CNNx16_0 | Bias RAM | 32KB | 0x5010 8000 | 0x5010 FFFF |
| | TRAM | 192KB | 0x5011 0000 | 0x5014 FFFF |
| | Mask RAM | 192KB | 0x5018 0000 | 0x501B FFFF |
| | SRAM | 128KB | 0x5030 0000 | 0x5032 FFFF |
| CNNx16_1 | Bias RAM | 32KB | 0x5050 8000 | 0x5050 FFFF |
| | TRAM | 192KB | 0x5051 0000 | 0x5054 FFFF |
| | Mask RAM | 192KB | 0x5058 0000 | 0x505B FFFF |
| | SRAM | 128KB | 0x5070 0000 | 0x5082 FFFF |
| CNNx16_2 | Bias RAM | 32KB | 0x5090 8000 | 0x5090 FFFF |
| | TRAM | 192KB | 0x5091 0000 | 0x5094 FFFF |
| | Mask RAM | 192KB | 0x5098 0000 | 0x509B FFFF |
| | SRAM | 128KB | 0x50B0 0000 | 0x50B2 FFFF |
| CNNx16_3 | Bias RAM | 32KB | 0x50D0 8000 | 0x50D0 FFFF |
| | TRAM | 192KB | 0x50D1 0000 | 0x50D4 FFFF |
| | Mask RAM | 192KB | 0x50D8 0000 | 0x50DB FFFF |
| | SRAM | 128KB | 0x50F0 0000 | 0x510F2 FFFF |

26.3.1 *CNNx16_n TRAM Details*

Each CNNx16 processor array includes 192KB of Tornado RAM (TRAM), however the memory allocated to this region spans 256KB addressable memory. The TRAM is arranged as 16 blocks of 16KB of addressable memory space, however, only the first 12KB of each of these 16 blocks contains TRAM.

Table 26-2, Table 26-3, Table 26-4, and Table 26-5, below, show each of the four CNNx16 Processor Array's TRAM start address and the valid TRAM end address for the 16 TRAM blocks. Memory addresses between the *Valid TRAM Block End Address* and the *Block End Address* are invalid memory addresses and must not be used.

Table 26-2: CNNx16 Processor Array 0 TRAM Mapping Details (APB Accessible)

| CNN Quadrant# | TRAM Block | Size | Start Address | Valid TRAM Block End Address | Block End Address |
|---------------|------------|------|---------------|------------------------------|-------------------|
| CNNx16_0 | 0 | 12KB | 0x5011 0000 | 0x5011 2FFF | 0x5011 3FFF |
| | 1 | 12KB | 0x5011 4000 | 0x5011 6FFF | 0x5011 7FFF |
| | 2 | 12KB | 0x5011 8000 | 0x5011 AFFF | 0x5011 BFFF |
| | 3 | 12KB | 0x5011 C000 | 0x5011 EFFF | 0x5011 FFFF |
| | 4 | 12KB | 0x5012 0000 | 0x5012 2FFF | 0x5012 3FFF |
| | 5 | 12KB | 0x5012 4000 | 0x5012 6FFF | 0x5012 7FFF |
| | 6 | 12KB | 0x5012 8000 | 0x5012 AFFF | 0x5012 BFFF |
| | 7 | 12KB | 0x5012 C000 | 0x5012 EFFF | 0x5012 FFFF |
| | 8 | 12KB | 0x5013 0000 | 0x5013 2FFF | 0x5013 3FFF |
| | 9 | 12KB | 0x5013 4000 | 0x5013 6FFF | 0x5013 7FFF |
| | 10 | 12KB | 0x5013 8000 | 0x5013 AFFF | 0x5013 BFFF |
| | 11 | 12KB | 0x5013 C000 | 0x5013 EFFF | 0x5013 FFFF |
| | 12 | 12KB | 0x5014 0000 | 0x5014 2FFF | 0x5014 3FFF |
| | 13 | 12KB | 0x5014 0000 | 0x5014 6FFF | 0x5014 7FFF |
| | 14 | 12KB | 0x5014 0000 | 0x5014 AFFF | 0x5014 BFFF |
| | 15 | 12KB | 0x5014 0000 | 0x5014 EFFF | 0x5014 FFFF |

Table 26-3: CNNx16 Processor Array 1 TRAM Mapping Details (APB Accessible)

| CNN Quadrant# | TRAM Block | Size | Start Address | Valid TRAM Block End Address | Block End Address |
|---------------|------------|------|---------------|------------------------------|-------------------|
| CNNx16_1 | 0 | 12KB | 0x5051 0000 | 0x5051 2FFF | 0x5051 3FFF |
| | 1 | 12KB | 0x5051 4000 | 0x5051 6FFF | 0x5051 7FFF |
| | 2 | 12KB | 0x5051 8000 | 0x5051 AFFF | 0x5051 BFFF |
| | 3 | 12KB | 0x5051 C000 | 0x5051 EFFF | 0x5051 FFFF |
| | 4 | 12KB | 0x5052 0000 | 0x5052 2FFF | 0x5052 3FFF |
| | 5 | 12KB | 0x5052 4000 | 0x5052 6FFF | 0x5052 7FFF |
| | 6 | 12KB | 0x5052 8000 | 0x5052 AFFF | 0x5052 BFFF |
| | 7 | 12KB | 0x5052 C000 | 0x5052 EFFF | 0x5052 FFFF |
| | 8 | 12KB | 0x5053 0000 | 0x5053 2FFF | 0x5053 3FFF |
| | 9 | 12KB | 0x5053 4000 | 0x5053 6FFF | 0x5053 7FFF |
| | 10 | 12KB | 0x5053 8000 | 0x5053 AFFF | 0x5053 BFFF |
| | 11 | 12KB | 0x5053 C000 | 0x5053 EFFF | 0x5053 FFFF |
| | 12 | 12KB | 0x5054 0000 | 0x5054 2FFF | 0x5054 3FFF |
| | 13 | 12KB | 0x5054 0000 | 0x5054 6FFF | 0x5054 7FFF |

| CNN Quadrant# | TRAM Block | Size | Start Address | Valid TRAM Block End Address | Block End Address |
|---------------|------------|------|---------------|------------------------------|-------------------|
| | 14 | 12KB | 0x5054 0000 | 0x5054 AFFF | 0x5054 BFFF |
| | 15 | 12KB | 0x5054 0000 | 0x5054 EFFF | 0x5054 FFFF |

Preliminary Draft 08/31/2020

Table 26-4: CNNx16 Processor Array 2 TRAM Mapping Details (APB Accessible)

| CNN Quadrant# | TRAM Block | Size | Start Address | Valid TRAM Block End Address | Block End Address |
|---------------|------------|------|---------------|------------------------------|-------------------|
| CNNx16_2 | 0 | 12KB | 0x5091 0000 | 0x5091 2FFF | 0x5091 3FFF |
| | 1 | 12KB | 0x5091 4000 | 0x5091 6FFF | 0x5091 7FFF |
| | 2 | 12KB | 0x5091 8000 | 0x5091 AFFF | 0x5091 BFFF |
| | 3 | 12KB | 0x5091 C000 | 0x5091 EFFF | 0x5091 FFFF |
| | 4 | 12KB | 0x5092 0000 | 0x5092 2FFF | 0x5092 3FFF |
| | 5 | 12KB | 0x5092 4000 | 0x5092 6FFF | 0x5092 6FFF |
| | 6 | 12KB | 0x5092 8000 | 0x5092 AFFF | 0x5092 BFFF |
| | 7 | 12KB | 0x5092 C000 | 0x5092 EFFF | 0x5092 FFFF |
| | 8 | 12KB | 0x5093 0000 | 0x5093 2FFF | 0x5093 3FFF |
| | 9 | 12KB | 0x5093 4000 | 0x5093 6FFF | 0x5093 7FFF |
| | 10 | 12KB | 0x5093 8000 | 0x5093 AFFF | 0x5093 BFFF |
| | 11 | 12KB | 0x5093 C000 | 0x5093 EFFF | 0x5093 FFFF |
| | 12 | 12KB | 0x5094 0000 | 0x5094 2FFF | 0x5094 3FFF |
| | 13 | 12KB | 0x5094 0000 | 0x5094 6FFF | 0x5094 7FFF |
| | 14 | 12KB | 0x5094 0000 | 0x5094 AFFF | 0x5094 BFFF |
| | 15 | 12KB | 0x5094 0000 | 0x5094 EFFF | 0x5094 FFFF |

Table 26-5: CNNx16 Processor Array 3 TRAM Mapping Details (APB Accessible)

| CNN Quadrant# | TRAM Block | Size | Start Address | Valid TRAM Block End Address | Block End Address |
|---------------|------------|------|---------------|------------------------------|-------------------|
| CNNx16_3 | 0 | 12KB | 0x50D1 0000 | 0x50D1 2FFF | 0x50D1 3FFF |
| | 1 | 12KB | 0x50D1 4000 | 0x50D1 6FFF | 0x50D1 7FFF |
| | 2 | 12KB | 0x50D1 8000 | 0x50D1 AFFF | 0x50D1 BFFF |
| | 3 | 12KB | 0x50D1 C000 | 0x50D1 EFFF | 0x50D1 FFFF |
| | 4 | 12KB | 0x50D2 0000 | 0x50D2 2FFF | 0x50D2 3FFF |
| | 5 | 12KB | 0x50D2 4000 | 0x50D2 6FFF | 0x50D2 7FFF |
| | 6 | 12KB | 0x50D2 8000 | 0x50D2 AFFF | 0x50D2 BFFF |
| | 7 | 12KB | 0x50D2 C000 | 0x50D2 EFFF | 0x50D2 FFFF |
| | 8 | 12KB | 0x50D3 0000 | 0x50D3 2FFF | 0x50D3 3FFF |
| | 9 | 12KB | 0x50D3 4000 | 0x50D3 6FFF | 0x50D3 7FFF |
| | 10 | 12KB | 0x50D3 8000 | 0x50D3 AFFF | 0x50D3 BFFF |
| | 11 | 12KB | 0x50D3 C000 | 0x50D3 EFFF | 0x50D3 FFFF |
| | 12 | 12KB | 0x50D4 0000 | 0x50D4 2FFF | 0x50D4 3FFF |
| | 13 | 12KB | 0x50D4 0000 | 0x50D4 6FFF | 0x50D4 7FFF |
| | 14 | 12KB | 0x50D4 0000 | 0x50D4 AFFF | 0x50D4 BFFF |
| | 15 | 12KB | 0x50D4 0000 | 0x50D4 EFFF | 0x50D4 FFFF |

26.3.2 *CNNx16_n Mask RAM Details*

Each CNNx16 processor array includes 192KB of Mask RAM, however the memory allocated to this region spans 256KB address space. The MRAM is arranged as 16 blocks of 16KB of addressable memory space, however, only the first 12KB of each of these 16 blocks contains usable Mask RAM.

The following tables below (*Table 26-6*,

Table 26-7, [Table 26-8](#), and

Table 26-9 show each of the four CNNx16 Processor Array's Mask RAM start address and the valid Mask RAM end address for the 16 Mask RAM blocks. Memory addresses between the *Valid MRAM Block End Address* and the *Block End Address* are invalid memory addresses and cannot be used.

Table 26-6: CNNx16 Processor Array 0 MRAM Mapping Details (APB Accessible)

| CNN Quadrant# | Mask RAM Block | Size | MRAM Block Start Address | Valid MRAM Block End Address | Block End Address |
|---------------|----------------|------|--------------------------|------------------------------|-------------------|
| CNNx16_0 | 0 | 12KB | 0x5018 0000 | 0x5018 2FFF | 0x5018 3FFF |
| | 1 | 12KB | 0x5018 4000 | 0x5018 6FFF | 0x5018 7FFF |
| | 2 | 12KB | 0x5018 8000 | 0x5018 AFFF | 0x5018 BFFF |
| | 3 | 12KB | 0x5018 C000 | 0x5018 EFFF | 0x5018 FFFF |
| | 4 | 12KB | 0x5019 0000 | 0x5019 2FFF | 0x5019 3FFF |
| | 5 | 12KB | 0x5019 4000 | 0x5019 6FFF | 0x5019 7FFF |
| | 6 | 12KB | 0x5019 8000 | 0x5019 AFFF | 0x5019 BFFF |
| | 7 | 12KB | 0x5019 C000 | 0x5019 EFFF | 0x5019 FFFF |
| | 8 | 12KB | 0x501A 0000 | 0x501A 2FFF | 0x501A 3FFF |
| | 9 | 12KB | 0x501A 4000 | 0x501A 6FFF | 0x501A 7FFF |
| | 10 | 12KB | 0x501A 8000 | 0x501A AFFF | 0x501A BFFF |
| | 11 | 12KB | 0x501A C000 | 0x501A EFFF | 0x501A FFFF |
| | 12 | 12KB | 0x501B 0000 | 0x501B 2FFF | 0x501B 3FFF |
| | 13 | 12KB | 0x501B 0000 | 0x501B 6FFF | 0x501B 7FFF |
| | 14 | 12KB | 0x501B 0000 | 0x501B AFFF | 0x501B BFFF |
| | 15 | 12KB | 0x501B 0000 | 0x501B EFFF | 0x501B FFFF |

Table 26-7: CNNx16 Processor Array 1 Mask RAM Mapping Details (APB Accessible)

| CNN Quadrant# | Mask RAM Block | Size | MRAM Block Start Address | Valid MRAM Block End Address | Block End Address |
|---------------|----------------|------|--------------------------|------------------------------|-------------------|
| CNNx16_1 | 0 | 12KB | 0x5058 0000 | 0x50D8 2FFF | 0x5058 3FFF |
| | 1 | 12KB | 0x5058 4000 | 0x50D8 6FFF | 0x5058 7FFF |
| | 2 | 12KB | 0x5058 8000 | 0x50D8 AFFF | 0x5058 BFFF |
| | 3 | 12KB | 0x5058 C000 | 0x50D8 EFFF | 0x5058 FFFF |
| | 4 | 12KB | 0x5059 0000 | 0x50D9 2FFF | 0x5059 3FFF |
| | 5 | 12KB | 0x5059 4000 | 0x50D9 6FFF | 0x5059 7FFF |
| | 6 | 12KB | 0x5059 8000 | 0x50D9 AFFF | 0x5059 BFFF |
| | 7 | 12KB | 0x5059 C000 | 0x50D9 EFFF | 0x5059 FFFF |
| | 8 | 12KB | 0x505A 0000 | 0x50DA 2FFF | 0x505A 3FFF |
| | 9 | 12KB | 0x505A 4000 | 0x50DA 6FFF | 0x505A 7FFF |
| | 10 | 12KB | 0x505A 8000 | 0x50DA AFFF | 0x505A BFFF |
| | 11 | 12KB | 0x505A C000 | 0x50DA EFFF | 0x505A FFFF |
| | 12 | 12KB | 0x505B 0000 | 0x50DB 2FFF | 0x50DB 3FFF |
| | 13 | 12KB | 0x505B 0000 | 0x50DB 6FFF | 0x50DB 7FFF |
| | 14 | 12KB | 0x505B 0000 | 0x50DB AFFF | 0x50DB BFFF |
| | 15 | 12KB | 0x505B 0000 | 0x50DB EFFF | 0x50DB FFFF |

Table 26-8: CNNx16 Processor Array 2 Mask RAM Mapping Details (APB Accessible)

| CNN Quadrant# | Mask RAM Block | Size | Start Address | Valid MRAM Block End Address | Block End Address |
|---------------|----------------|------|---------------|------------------------------|-------------------|
| CNNx16_2 | 0 | 12KB | 0x5098 0000 | 0x5098 2FFF | 0x5098 3FFF |
| | 1 | 12KB | 0x5098 4000 | 0x5098 6FFF | 0x5098 7FFF |
| | 2 | 12KB | 0x5098 8000 | 0x5098 AFFF | 0x5098 CFFF |
| | 3 | 12KB | 0x5098 C000 | 0x5098 EFFF | 0x5098 FFFF |
| | 4 | 12KB | 0x5099 0000 | 0x5099 2FFF | 0x5099 3FFF |
| | 5 | 12KB | 0x5099 4000 | 0x5099 6FFF | 0x5099 7FFF |
| | 6 | 12KB | 0x5099 8000 | 0x5099 AFFF | 0x5099 BFFF |
| | 7 | 12KB | 0x5099 C000 | 0x5099 EFFF | 0x5099 FFFF |
| | 8 | 12KB | 0x509A 0000 | 0x509A 2FFF | 0x509A 3FFF |
| | 9 | 12KB | 0x509A 4000 | 0x509A 6FFF | 0x509A 7FFF |
| | 10 | 12KB | 0x509A 8000 | 0x509A AFFF | 0x509A BFFF |
| | 11 | 12KB | 0x509A C000 | 0x509A 3FFF | 0x509A FFFF |
| | 12 | 12KB | 0x509B 0000 | 0x509B 2FFF | 0x509B 3FFF |
| | 13 | 12KB | 0x509B 0000 | 0x509B 6FFF | 0x509B 7FFF |
| | 14 | 12KB | 0x509B 0000 | 0x509B AFFF | 0x509B BFFF |
| | 15 | 12KB | 0x509B 0000 | 0x509B EFFF | 0x509B FFFF |

Table 26-9: CNNx16 Processor Array3 Mask RAM Mapping Details (APB Accessible)

| CNN Quadrant# | Mask RAM Block | Size | Memory Start Address | Valid MRAM Block End Address | Block End Address |
|---------------|----------------|------|----------------------|------------------------------|-------------------|
| CNNx16_3 | 0 | 12KB | 0x50D8 0000 | 0x50D8 2FFF | 0x50D8 3FFF |
| | 1 | 12KB | 0x50D8 4000 | 0x50D8 6FFF | 0x50D8 7FFF |
| | 2 | 12KB | 0x50D8 8000 | 0x50D8 AFFF | 0x50D8 CFFF |
| | 3 | 12KB | 0x50D8 C000 | 0x50D8 EFFF | 0x50D8 FFFF |
| | 4 | 12KB | 0x50D9 0000 | 0x50D9 2FFF | 0x50D9 3FFF |
| | 5 | 12KB | 0x50D9 4000 | 0x50D9 6FFF | 0x50D9 7FFF |
| | 6 | 12KB | 0x50D9 8000 | 0x50D9 AFFF | 0x50D9 BFFF |
| | 7 | 12KB | 0x50D9 C000 | 0x50D9 EFFF | 0x50D9 FFFF |
| | 8 | 12KB | 0x50DA 0000 | 0x50DA 2FFF | 0x50DA 3FFF |
| | 9 | 12KB | 0x50DA 4000 | 0x50DA 6FFF | 0x50DA 7FFF |
| | 10 | 12KB | 0x50DA 8000 | 0x50DA AFFF | 0x50DA BFFF |
| | 11 | 12KB | 0x50DA C000 | 0x50DA EFFF | 0x50DA FFFF |
| | 12 | 12KB | 0x50DB 0000 | 0x50DB 2FFF | 0x50DB 3FFF |
| | 13 | 12KB | 0x50DB 0000 | 0x50DB 6FFF | 0x50DB 7FFF |
| | 14 | 12KB | 0x50DB 0000 | 0x50DB AFFF | 0x50DB BFFF |
| | 15 | 12KB | 0x50DB 0000 | 0x50DB EFFF | 0x50DB FFFF |

26.3.3 CNNx16_n SRAM Details

26.4 CNN Global Registers (CNN)

See [Table 2-4: APB Peripheral Base Address Map](#) for the base address of this peripheral/module. There is one instance of the CNN in the MAX78000 and the global CNN registers are shown in [Table 26-10: Global CNN Register Summary](#).

See [Table 2-1: Field Access Definitions](#) for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 26-10: Global CNN Register Summary

| Offset | Register | Description |
|----------|-------------------------------|--------------------------------|
| [0x0000] | CNN_FIFO_CTRL | CNN FIFO Control Register |
| [0x0004] | CNN_FIFO_STAT | CNN FIFO Status Register |
| [0x0008] | CNN_FIFO_WRO | CNN FIFO 0 Write Register |
| [0x000C] | CNN_FIFO_WR1 | CNN FIFO 1 Write Register |
| [0x0010] | CNN_FIFO_WR2 | CNN Fast FIFO 2 Write Register |
| [0x0014] | CNN_FIFO_WR3 | CNN FIFO 3 Write Register |
| [0x1000] | CNN_AOD_CTRL | CNN AoD Control Register |

26.4.1 Global CNN Register Details

Table 26-11: CNN FIFO Control Register

| CNN FIFO Control | | | CNN_FIFO_CTRL | | [0x0000] |
|------------------|---------------------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:28 | almost_empty_int_en | R/W | 0 | FIFO Almost Empty Interrupt Enable Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, Bit 2 maps to CNNx16_n_q2, Bit1 maps to CNNx16_n_q1 and Bit 0 maps to CNNx16_n_q0. Set this field to 1 to enable an IRQ event based on the CNN_FIFO_CTRL.empty_thresh flag. Enable the IRQ for all quadrants by setting this field to 0b1111. | |
| 27:24 | almost_full_int_en | R/W | 0 | FIFO Almost Full Interrupt Enable Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, Bit 2 maps to CNNx16_n_q2, Bit1 maps to CNNx16_n_q1 and Bit 0 maps to CNNx16_n_q0. Set this field to 1 to enable an IRQ event based on the FIFO almost full threshold flag, CNN_FIFO_CTRL.full_thresh . Enable the IRQ for all quadrants by setting this field to 0b1111. | |
| 23:20 | empty_int_en | R/W | 0 | FIFO Empty Interrupt Enable Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, Bit 2 maps to CNNx16_n_q2, Bit1 maps to CNNx16_n_q1 and Bit 0 maps to CNNx16_n_q0. Set this field to 1 to enable an IRQ event based on the FIFO empty flag. Enable the IRQ for all quadrants by setting this field to 0b1111 | |
| 19:16 | full_int_en | R/W | 0 | FIFO Full Interrupt Enable Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, Bit 2 maps to CNNx16_n_q2, Bit1 maps to CNNx16_n_q1 and Bit 0 maps to CNNx16_n_q0. Set this field to 1 to enable an IRQ event based on the FIFO full flag. Enable the IRQ for all quadrants by setting this field to 0b1111. | |
| 15:12 | fifo_en | R/W | 0 | Per FIFO Enable Set this field to 1 to enable the corresponding FIFO for the specified CNNx16_n quadrant. Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, Bit 2 maps to CNNx16_n_q2, Bit1 maps to CNNx16_n_q1 and Bit 0 maps to CNNx16_n_q0. <i>Note: Unused FIFOs must be disabled.</i> | |
| 11 | fifo_cpl | R/W | 0 | FIFO Coupling Setting this bit to 1 forces the FIFOs to operate in lockstep. Data available status is dependent on all FIFOs having identical write pointer values. 0: FIFOs are not coupled 1: FIFOs are coupled and operate in lockstep | |
| 10 | - | R/W | 0 | Reserved Do not modify | |
| 9:7 | empty_thresh | R/W | 0 | FIFO Almost Empty Threshold If the difference between the write and read pointer (read_pointer - write_pointer) falls below this number of bytes, the almost empty flag is set. 0 to 7: Sets the FIFO Almost Empty Threshold to the value written. | |
| 6:5 | - | R/W | 0 | Reserved Do not modify | |

| CNN FIFO Control | | | CNN_FIFO_CTRL | | [0x0000] |
|------------------|-------------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 4:2 | full_thresh | R/W | 0 | FIFO Almost Full Threshold This flag is set if the difference between the write and read pointer (read_pointer - write_pointer) exceeds this number of bytes, the almost full flag is set. 0 to 7: Sets the FIFO Almost Full Threshold to the value written. | |
| 1:0 | rdy_sel | R/W | b11 | APB Wait State Selection This field determine the number of wait states added during an APB access. 0b00: 0 wait states 0b01: 1 wait state 0b10: 2 wait states 0b11: 3 wait states <i>Note: Write operations load the data one clock before the end of the cycle.</i> | |

Table 26-12: CNN FIFO Status Register

| CNN FIFO Status | | | CNN_FIFO_STAT | | [0x0004] |
|-----------------|--------------------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:22 | - | R/W | 0 | Reserved Do not modify | |
| 21 | rptr_eq | RO | 0 | FIFO Read Pointers Equal This bit is set when all active FIFO read pointers are equal | |
| 20 | wptr_eq | RO | 0 | FIFO Write Pointers Equal This bit is set when all active FIFO write pointers are equal | |
| 19 | fifos_almost_empty | RO | 0 | Aggregate FIFO Almost Empty Status Logical OR of individual FIFO almost empty statuses. | |
| 18 | fifos_almost_full | RO | 0 | Aggregate FIFO Almost Full Status Logical AND of individual FIFO almost full statuses. | |
| 17 | fifos_empty | RO | 0 | Aggregate FIFO Empty Status Logical OR of individual FIFO empty statuses. | |
| 16 | fifos_full | RO | 0 | Aggregate FIFO Full Status Logical AND of individual FIFO full statuses. | |
| 15:12 | almost_empty | RO | 0 | Per FIFO Almost Empty Status If a bit in this field reads 1, the corresponding FIFO is almost empty. Each bit corresponds to a FIFO with almost_empty[0] mapped to CNNx16_n FIFO0. This status is persistent until the condition changes, or until application firmware disables the FIFO. | |
| 11:8 | almost_full | RO | 0 | Per FIFO Almost Full Status If a bit in this field reads 1, the corresponding FIFO is almost full. Each bit corresponds to a FIFO with almost_full[0] mapped to CNNx16_n FIFO0. This status is persistent until the condition changes, or until application firmware disables the FIFO. | |

| CNN FIFO Status | | | CNN_FIFO_STAT | | [0x0004] |
|-----------------|-------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 7:4 | empty | RO | 0 | Per FIFO Empty Status If a bit in this field reads 1, the corresponding FIFO is empty. Each bit corresponds to a FIFO with empty[0] mapped to CNNx16_n FIFO0. This status is persistent until the condition changes, or until application firmware disables the FIFO. | |
| 3:0 | full | RO | 0 | Per FIFO Full Status If a bit in this field reads 1, the corresponding FIFO is full. Each bit corresponds to a FIFO with full[0] mapped to CNNx16_n FIFO0. The status is persistent until the condition changes, or until application firmware disables the FIFO. | |

Table 26-13: CNN FIFO 0 Write Register

| CNN FIFO 0 Write | | | CNN_FIFO_WRO | | [0x0008] |
|------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | FIFO 0 Data CNNx16_n FIFO 0 Data register | |

Table 26-14: CNN FIFO 1 Write Register

| CNN FIFO 1 Write | | | CNN_FIFO_WR1 | | [0x000C] |
|------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | FIFO 1 Data CNNx16_n FIFO 2 Data register | |

Table 26-15: CNN FIFO 2 Write Register

| CNN FIFO 2 Write | | | CNN_FIFO_WR2 | | [0x0010] |
|------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | FIFO 2 Data CNNx16_n FIFO 2 Data register | |

Table 26-16: CNN FIFO 3 Write Register

| CNN FIFO 3 Write | | | CNN_FIFO_WR3 | | [0x0014] |
|------------------|-------|--------|--------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | data | R/W | 0 | FIFO 3 Data CNNx16_n FIFO 3 Data register | |

Table 26-17: CNN Always On Domain Control Register

| CNN Always On Domain Control | | | CNN_AOD_CTRL | | [0x1000] |
|------------------------------|-------|--------|--------------|----------------------------------|----------|
| Bits | Field | Access | Reset | Description | |
| 31:20 | - | R/W | 0 | Reserved Do not modify | |

| CNN Always On Domain Control | | | CNN_AOD_CTRL | | [0x1000] |
|------------------------------|---------|--------|--------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 19:16 | rm | R/W | 0 | Mask RAM Read Margin MSB Each bit of this field maps to one of the four CNNx16_n quadrants. Bit 0 maps to CNNx16_n_q0, Bit 1 maps to CNNx16_n_q1, bit 2 maps to CNNx16_n_q2 and bit 3 maps to CNNx16_n_q3. | |
| 15:12 | pd | R/W | 0 | Mask RAM Power Down Enable Each bit of this field maps to a CNNx16_n quadrant. Setting the quadrant's bit position to 1 shuts down power to the Mask RAM in the CNNx16_n quadrant. Write 0b1111 to this field to power down all CNNx16_n quadrants' Mask RAM. When a CNNx16_n quadrant's Mask RAM is put into power down mode, the contents of the Mask RAM are <i>not</i> maintained. 0: In any bit position, the corresponding CNNx16_n's Mask RAM is not in a power down state. 1: In any bit position, the corresponding CNNx16_n's Mask RAM is in a power down state. <i>Note: This field is independent of the Low Power Mode settings. Refer to Operating Modes for more information on the system low power modes and their effect on the CNN and CNNx16_n quadrant memories.</i> | |
| 11:8 | dsleep | R/W | 0 | Mask RAM Deep Sleep Enable Each bit of this field maps to a CNNx16_n quadrant. Setting the quadrant's bit position to 1 puts the specified CNNx16_n's MRAM in a deep sleep state. Write 0b1111 to this field to put all four of the CNNx16_n quadrant's Mask RAM in deep sleep mode. When a CNNx16_n quadrant's Mask RAM is in deep sleep, the contents of the Mask RAM are retained, but the memory cannot be accessed. 0: In any bit position, the corresponding CNNx16_n's Mask RAM is not in deep sleep. 1: In any bit position, the corresponding CNNx16_n's Mask RAM is in deep sleep. <i>Note: This field is independent of the Low Power Mode settings. Refer to Operating Modes for more information on the system low power modes and their effect on the CNN and CNNx16_n quadrant memories.</i> | |
| 7:2 | - | R/W | 0 | Reserved Do not modify | |
| 1:0 | rdy_sel | R/W | b11 | APB Wait State Selection This field determine the number of wait states added during an APB access. 0: 0 wait states 1: 1 wait state 2: 2 wait states 3: 3 wait states <i>Note: Write operations load the data one clock before the end of the cycle.</i> | |

26.5 CNNx16 Processor Array (CNNx16_n) Registers

The CNN includes four $\times 16$ Processor Arrays, referred to as CNNx16_n. Each processor array includes a dedicated FIFO interface allowing configuration, status and write access to each of the four CNNx16_n's individual memory spaces. The table below shows the Base Address for each of the four CNNx16_n processor arrays.

Table 26-18: CNNx16_n Instances and Base Offset Address

| Base Offset Address | Processor Array | Description |
|---------------------|-----------------|-----------------------------------|
| [0x0010 0000] | CNNx16_0 | CNN $\times 16$ Processor Array 0 |
| [0x0050 0000] | CNNx16_1 | CNN $\times 16$ Processor Array 1 |
| [0x0090 0000] | CNNx16_2 | CNN $\times 16$ Processor Array 2 |
| [0x00D0 0000] | CNNx16_3 | CNN $\times 16$ Processor Array 3 |

26.5.1 CNNx16_n Instances and Base Offset Addresses

Table 26-18: CNNx16_n Instances and Base Offset Address shows the base address for each of the four CNNx16 processor arrays. Each CNNx16 processor array has its own, independent set of registers shown in *Table 26-19: CNNx16_n Processor Array Registers*. The base address for a specific CNNx16_n processor array is calculated by adding the CNNx16_n base offset address to the global CNN base address shown in *Table 2-4: APB Peripheral Base Address Map*.

Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See *Table 2-1: Field Access Definitions* for an explanation of the read and write access of each field. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific resets.

Table 26-19: CNNx16_n Processor Array Registers

| Offset | Register | Description |
|-------------------|-------------------------------|--|
| [0x0000] | <i>CNNx16_n_CTRL</i> | CNNx16_n Control Register |
| [0x0004] | <i>CNNx16_n_SRAM</i> | CNNx16_n SRAM Control Register |
| [0x0008] | <i>CNNx16_n_LCNT_MAX</i> | CNNx16_n Layer Count Maximum Register |
| [0x000C] | <i>CNNx16_n_TEST</i> | CNNx16_n SRAM Test Register |
| [0x0990] | <i>CNNx16_n_IFRM</i> | CNNx16_n Input FIFO Frame Size Register |
| [0x1000] | <i>CNNx16_n_MLAT</i> | CNNx16_n Muchselator Data Register |
| [0x0010]-[0x008C] | <i>CNNx16_n_Ly_RCNT</i> | CNNx16_n_Ly Row Count Register |
| [0x0090]-[0x010C] | <i>CNNx16_n_Ly_CCNT</i> | CNNx16_n_Ly Column Count Register |
| [0x0110]-[0x018C] | <i>CNNx16_n_Ly_ONED</i> | CNNx16_n_Ly One Dimensional Control Register |
| [0x0190]-[0x020C] | <i>CNNx16_n_Ly_PRCNT</i> | CNNx16_n_Ly Pool Row Count Register |
| [0x0210]-[0x028C] | <i>CNNx16_n_Ly_PCCNT</i> | CNNx16_n_Ly Pool Column Count Register |
| [0x0290]-[0x030C] | <i>CNNx16_n_Ly_STRIDE</i> | CNNx16_n_Ly Stride Count Register |
| [0x0310]-[0x038C] | <i>CNNx16_n_Ly_WPTR_BASE</i> | CNNx16_n_Ly Write Pointer Base Address Register |
| [0x0390]-[0x040C] | <i>CNNx16_n_Ly_WPTR_TOFF</i> | CNNx16_n_Ly Write Pointer Timeslot Offset Register |
| [0x0410]-[0x048C] | <i>CNNx16_n_Ly_WPTR_MOFF</i> | CNNx16_n_Ly Write Pointer Mask Offset Register |
| [0x0490]-[0x050C] | <i>CNNx16_n_Ly_WPTR_CHOFF</i> | CNNx16_n_Ly Write Pointer Multi-Pass Channel Offset Register |
| [0x0510]-[0x058C] | <i>CNNx16_n_Ly_RPTR_BASE</i> | CNNx16_n_Ly Read Pointer Base Address Register |
| [0x0590]-[0x060C] | <i>CNNx16_n_Ly_LCTRLO</i> | CNNx16_n_Ly Layer Control 0 Register |
| [0x0610]-[0x068C] | <i>CNNx16_n_Ly_MCNT</i> | CNNx16_n_Ly Mask Count Register |
| [0x0690]-[0x070C] | <i>CNNx16_n_Ly_TPTR</i> | CNNx16_n_Ly TRAM Pointer Register |
| [0x0710]-[0x078C] | <i>CNNx16_n_Ly_EN</i> | CNNx16_n_Ly Enable Register |
| [0x0790]-[0x080F] | <i>CNNx16_n_Ly_POST</i> | CNNx16_n_Ly Post Processing Register |

| Offset | Register | Description |
|-------------------|------------------------------------|---|
| [0x0A10]-[0x0A8C] | CNNx16_n_Ly_LCTRL1 | CNNx16_n_Ly Layer Control 1 Register |
| [0x0810]-[0x082C] | CNNx16_n_Sz_STRMO | CNNx16_n_Sz Stream Control 0 Register |
| [0x0890]-[0x08AC] | CNNx16_n_Sz_STRM1 | CNNx16_n_Sz Stream Control 1 Register |
| [0x0910]-[0x092C] | CNNx16_n_Sz_FBUF | CNNx16_n_Sz Stream Frame Buffer Size Register |

26.5.2 CNN Per x16 Processor Register Details

Table 26-20: CNNx16_n Control Register

| CNNx16_n Control | | CNNx16_n_CTRL | | [0x0000] |
|------------------|----------|---------------|-------|---|
| Bits | Field | Access | Reset | Description |
| 31 | qupac | R/W | 0 | QuPac Mode The quad processors pack bit enables parallel processing of the same data by each of the 16 processors in the CNNx16_n. 0: Normal processing mode 1: x16 parallel processing mode <i>Note: FIFO mode must be enabled, CNNx16_n_CTRL.ffifoen set to 1, prior to enabling QuPac mode.</i> |
| 30 | timeshft | R/W | 0 | Pooling Stage Time Shift When set, one wait state is added to the pooling stage to allow design time closure at a higher clock frequency. |
| 29:24 | fclk_dly | R/W | 0 | FIFO Clock Delay Selects the clock delay of the FIFO clock relative to the primary CNN clock. A setting of 0 adds in the minimum delay and a value of 0x3F add the maximum delay. |
| 23 | fifogrp | R/W | 0 | FIFO Group Output Enables sending all "little data" channels to the first 4 processors. When this bit is not set, each byte of FIFO data is directed to the first little data channel of each CNNx16_n processor. |
| 22 | ffifo_en | R/W | 0 | Fast FIFO Enable Enables the tightly coupled data path between the MAX78000's CNN_TX Fast FIFO and the CNN data SRAMs. The CNN_TX_FIFO is 32 bits wide with 8 bits being dedicated to each of 4 channels. Channel routing is controlled by the state of the CNNx16_n_CTRL.fifogrp control bit. |
| 21 | simple1b | R/W | 0 | Simple 1-Bit Weights Enable simple logic for 1-bit weights. Setting this bit disables the wide accumulators used to calculate the convolution products and activates simple one-bit logic functions. |
| 20 | mexpress | R/W | 0 | Mask Memory Packed Memory Enable Enable loading of the mask memories using packed data. With this bit set, a change in state of the two least significant bits of the MRAM address triggers a reload of the address counter. |
| 19 | lilbuf | R/W | 0 | Stream Mode Circular Buffer Enable Setting this bit restricts the associated read buffer's bounds to a circular buffer starting at the CNNx16_n_Ly_RPTR_BASE address and terminating at the CNNx16_n_Sz_FBUF address. When set, the circular buffer is used on all streaming layers. |

| CNNx16_n Control | | | CNNx16_n_CTRL | | [0x0000] |
|------------------|------------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 18:17 | mlatch_sel | R/W | 0 | SRAM Packed Channel Select Selects which of the four channels in the SRAM read data is packed. 0: Selects channel 0, SRAM data bits 7:0 1: Selects channel 1 SRAM data bits 15:8 2: Selects channel 2 SRAM data bits 23:16 3: Select channel 3 SRAM data bits 31:24 | |
| 16 | mlat_Id | R/W | 0 | Muchselator Load Data Writing the bit from a 0 to a 1 forces the CNNx16_n_Ly_WPTR_BASE address to be loaded into the Muchselator address counter and selects the counter as the SRAM address source. SRAM reads are only possible when CNNx16_n_CTRL.cnn_en is reset to 0. | |
| 15 | fifo_en | R/W | 0 | CNNx16_n_FIFO Enable When set, data for the first (input) layer is taken from the CNN_FIFO_WRn FIFO register. One 4 byte wide FIFO is dedicated for each of the four processor arrays. The FIFOs are accessed through the APB memory map, and each can be used in a single byte wide channel mode, a single 4 byte wide channel mode, or 4 single byte wide channel mode. Mode is determined by the data configuration written to the FIFO via the APB, the CNNx16_n_CTRL.bigdata/CNNx16_n_Ly_LCTRL0.parallel configuration, and the channel enables. | |
| 14 | stream_en | R/W | 0 | Streaming Mode Enable When set, streaming mode is enabled for the CNNx16_n processor array. Streaming behavior is defined by the CNNx16_n Processor Stream Registers Refer to Streaming Mode Configuration for additional information. <i>Note: Unexpected behavior is likely when all four CNNx16_n processor arrays are not configured identically for streaming/non-streaming operation. Each CNN_x16_n processor array should be configured identically for streaming or non-streaming operation. Refer to Streaming Mode Configuration for further details.</i> | |
| 13 | poolrnd | R/W | 0 | Average Pooling Enable When this bit is set and average pooling is enabled, pooled values are rounded up for remainders greater or equal to 0.5 and down for remainders less than 0.5. 0: Average Pooling Disabled 1: Average Pooling Enabled | |
| 12 | cnn_irq | R/W | 0 | CNN Interrupt Enable This read/write bit indicates when the CNN interrupt request is active. It can be written to zero to reset the interrupt request. This interrupt signals the completion of CNN processing and should be masked in the MCU interrupt control logic if not required. It can also be written to 1 to force an interrupt. 0: CNN IRQ not active, write 0 to clear the CNN IRQ 1: CNN IRQ active, write 1 to generate a CNN IRQ | |

| CNNx16_n Control | | | CNNx16_n_CTRL | | [0x0000] |
|------------------|-----------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 11:9 | ext_sync | R/W | 0 | CNNx16_n External Sync Select Each of these bits enable the external sync input from one of the CNN_x16_n processors. These bits allow the CNNx16_n processors to optionally operate in synchronization with one of the other CNNx16_n processors. In the general case, when all 64 processors are operating on a single convolution, CNNx16_0 processor 0 is selected by all four of the CNNx16_n's as the master byte setting ext_sync = 0b001. Combinations of processors can be configured as long as the groups are made up by sequential processors, without gaps. | |
| 8 | oneshot | R/W | 0 | One Shot Layer Mode With this bit set, only one layer is processed when the CNNx16_n_CTRL.cnn_en bit is set. To advance to the next layer, the CNNx16_n_CTRL.cnn_en bit must be reset to 0 and then set to 1. The low to high transition causes the CNN state machine to advance through the next layer. Memories can be interrogated between layers when CNNx16_n_CTRL.cnn_en is 0. 0: One shot layer mode disabled 1: One shot layer mode enabled | |
| 7 | apbclk_en | R/W | 0 | APB Clock Always On Setting this bit forces the APB clock always on. When this bit is set to 0, clocks are only generated to the APB registers during write operations. 0: APB clocks are only on when APB register writes occur 1: APB clocks to the CNN APB registers is always on | |
| 6 | bigdata | R/W | 0 | Big Data Enable This bit globally selects the input data format that uses four data bytes per read for the groups of 4 processors. In other words, the four bytes allocated for a group of four processors is multiplexed to the first processor of the group. | |
| 5 | pool_en | R/W | 0 | Pooling Enable This bit globally enables pooling for all layers. 0: Global pooling disabled 1: Global pooling enabled for all layers. <i>Note: If this bit is set and the CNNx16_n_CTRL.calcmax bit is not set, the per-layer CNNx16_n_Ly_LCTRL0.maxpl_en bits are in effect.</i> | |
| 4 | calcmax | R/W | 0 | Max Pooling Enable This bit globally enables max pooling for all layers when the CNNx16_n_CTRL.pool_en bit is set. <i>Note that this bit will be in effect, per layer, when the global CNNx16_n_CTRL.pool_en bit is 0 and the per-layer CNNx16_n_Ly_LCTRL0.pool_en bits are set.</i> | |
| 3 | clk_en | R/W | 0 | Data Processing Clock Enable Setting this bit enables the clocks to the . This field <i>does not</i> affect the clocks to the APB registers. Refer to the CNNx16_n_CTRL.apbclken bit for the description of the APB clock behavior. 0: Clocks disabled to the Data Processing registers 1: Clocks enabled to the Data Processing registers | |

| CNNx16_n Control | | | CNNx16_n_CTRL | | [0x0000] |
|------------------|---------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 2:1 | rdy_sel | R/W | b11 | APB Wait State Selection This field determine the number of wait states added during an APB access. 0b00: 0 wait states 0b01: 1 wait state 0b10: 2 wait states 0b11: 3 wait states <i>Note: Write operations load the data one clock before the end of the cycle.</i> | |
| 0 | cnn_en | R/W | 0 | CNN Enable Setting this bit to 1 initiates CNN processing. Processing is triggered by this field to 1. Firmware must set this field to 0 and back to 1 to perform subsequent CNN processing operations. 0: CNN processing stopped 1: Start CNN processing <i>Note: This field must be written to 0 before writing it to 1 for subsequent CNN processing to start.</i> | |

Table 26-21: CNNx16_n SRAM Control Register

| CNNx16_n SRAM Control | | | CNNx16_n_SRAM | | [0x0004] |
|-----------------------|--------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 22 | lsram | R/W | 0 | Bias Memory Light Sleep Setting this bit forces the Bias Memory into light sleep when the Bias Memory is not selected by the CNN system or the APB. | |
| 21 | lstram | R/W | 0 | TRAM Light Sleep Setting this bit forces the TRAMs into light sleep when the TRAM is not selected by the CNN system or the APB. | |
| 20 | lsmram | R/W | 0 | Mask RAM Light Sleep Setting this bit forces the Mask RAMs into light sleep when the MRAM is not selected by the CNN system or the APB. | |
| 19 | lsdram | R/W | 0 | Data RAM Light Sleep Setting this bit forces the Data RAMs into light sleep when the SRAM is not selected by the CNN system or the APB. | |
| 18:17 | - | R/W | 0 | Reserved for Future Use Do not modify | |
| 16 | pd | R/W | 0 | CNNx16_n Memory Power Down Enable Set this field to 1 to put the CNNx16_n's SRAM, TRAM, MRAM and Bias memory into a power down mode. All memory contents are lost in power down state. 0: CNNx16 memories are not powered down 1: Power down the CNNx16_n's memories | |

| CNNx16_n SRAM Control | | | | CNNx16_n_SRAM | [0x0004] |
|-----------------------|----------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 15 | ds | R/W | 0 | CNNx16_n Memory Deep Sleep Enable Set this field to 1 to put the CNNx16_n's SRAM, TRAM, MRAM and Bias memory into a deep sleep state. In deep sleep the memories contents are retained, but peripheral logic is powered down and the memory cannot be accessed. All memory outputs are set to output 0. 0: Memories are not in Deep Sleep state. 1: Put the CNNx16_n's memories into deep sleep state. <i>Note: This field has no effect If the CNNx16_n_SRAM.pd field is set to 1 (memories powered down).</i> | |
| 14 | - | R/W | 0 | Reserved Do not modify. | |
| 13:11 | wpulse | R/W | 0 | Write Pulse Width This field determines the bit line pulse width applied to the memory during writes. 0b000: Use the minimum bit line pulse width 0b111: Use the maximum bit line pulse width <i>Note: Values of wpulse between 0 and 7 incrementally set the bit line pulse width between the minimum and maximum values.</i> | |
| 10 | wneg_en | | | Write Negative Voltage Enable This bit enables the CNNx16_n_SRAM.wneg_vol . If this field is 0, the system controls the negative voltage applied to the bit lines. 0: Write Negative Voltage controlled by system 1: Write Negative Voltage Time controlled by the setting in the CNNx16_n_SRAM.wneg_vol field. | |
| 9:8 | wneg_vol | R/W | 0 | Write Negative Voltage This field sets the SRAM negative voltage level applied to the bit lines. This field is only used when the CNNx16_n_SRAM.wneg_en field is set to 1. 0: V _{DD} – 80mV 1: V _{DD} – 120mV 2: V _{DD} – 180mV 3: V _{DD} – 220mV | |
| 7:6 | ra | R/W | 0 | Read Assist Voltage This field controls the Read Assist value for the SRAM bit lines. 0: V _{DD} 1: V _{DD} – 20mV 2: V _{DD} – 40mV 3: V _{DD} – 60mV These bits determine the WL underdrive (Read Assist) value. ra[1:0] = 00 limits the WL voltage to VDD, ra[1:0] = 01 limits the WL to VDD-20mV, ra[1:0] = 10 limits WL to VDD-40mV, and ra[1:0] = 11 limits the WL voltage to VDD-60mV. | |

| CNNx16_n SRAM Control | | | | CNNx16_n_SRAM | [0x0004] |
|-----------------------|------------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 5:2 | rmargin | R/W | b0011 | SRAM Read Margin When CNNx16_n_SRAM.rm_en is set, this field determines the length of the memory access time. 0b0000: Slowest SRAM access time 0b0001: 0b0010: 0b0011: Fastest SRAM access time (Reset Default) 0b0100-0b1111: Reserved for Future Use <i>Note: The value of this field has no effect unless the CNNx16_n_CTRL.rm_en field is set to 1.</i> <i>Note: Do not set the value of this field between 0b0100 and 0b1111, these values are reserved for future use.</i> | |
| 1 | rmargin_en | R/W | b1 | SRAM Read Margin Enable Set this field to 1 to use the SRAM Access Time setting in the CNNx16_n_CTRL.ram_acc_time field. 0: SRAM access time is set by the hardware 1: SRAM access time controlled using the CNNx16_n_CTRL.ram_acc_time field. | |
| 0 | extacc | R/W | 0 | SRAM Extended Access Time Enable Set this bit to 1 to enable SRAM access time maximum. Enabling longer SRAM access time increases the power consumption of the SRAM. 0: SRAM Power Optimized, reduced performance 1: SRAM Extended Access, higher power <i>Note, this setting can be used to extend access time but force the memories to consume additional power when active. This bit applies to all SRAMs in the CNNx16_n processor.</i> | |

Table 26-22: CNNx16_n Layer Count Maximum Register

| CNNx16_n Layer Count Maximum | | | | CNNx16_n_LCNT_MAX | [0x0008] |
|------------------------------|-------|--------|-------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 31:5 | - | R/W | 0 | Reserved Do not modify. | |
| 4:0 | lcnt | R/W | 0 | Layer Count Maximum Set this field to the maximum layer number for processing by the CNNx16_n. When the CNNx16_n is enabled, processing starts at layer 0 and completes processing at the layer number set by this field. 0-31: Set to last layer for processing by the CNNx16_n <i>Note: The CNNx16_n must be inactive(CNNx16_n_CTRL.cnn_en=0) when setting this field.</i> | |

Table 26-23: CNNx16_n SRAM Test Register

| CNNx16_n SRAM Test | | | CNNx16_n_TEST | | [0x000C] |
|--------------------|-----------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 28 | zerodone | RO | 0 | BIST Zeroization Complete This field is set to 1 by hardware when any of the zero run bits are set to 1 by firmware and the hardware completes the zeroization. This bit is read only. Clear this bit by setting each of the zero run bits to 0. 1: BIST zeroization complete | |
| 27 | bistdone | R/W | 0 | BIST Run Complete This field is set to 1 by hardware when any of the BIST run bits are set to 1 by firmware and the hardware completes the BIST operation. This bit is read only. Clear this bit by setting each of the BIST run bits to 0. 1: BIST operation complete | |
| 26 | bistfail | R/W | 0 | BIST Run Failure Detected This field is set to 1 by hardware if a BIST run operation was run and a BIST failure occurred. This bit is read only. Clear this bit by setting each of the BIST run bits to 0. 0: If the CNNx16_n_TEST.bistdone bit reads 1, this bit indicates no BIST failures were detected. 1: BIST failure detected | |
| 25 | ballzdone | R/W | 0 | Bias RAM Zeroization Complete This field indicates a SRAM zeroization is completed. This field is reset by hardware when firmware writes the CNNx16_n_TEST.bzerorun field to 0. 1: Bias RAM zeroization complete | |
| 24 | tallzdone | R/W | 0 | TRAM Zeroization Complete This field indicates a TRAM zeroization is completed. This field is reset by hardware when firmware writes the CNNx16_n_TEST.tzerorun field to 0. 1: TRAM zeroization complete | |
| 23 | mallzdone | R/W | 0 | Mask RAM Zeroization Complete This field indicates a Mask RAM zeroization is completed. This field is reset by hardware when firmware writes the CNNx16_n_TEST.mzerorun field to 0. 1: Mask RAM zeroization complete | |
| 22 | sallzdone | R/W | 0 | SRAM Zeroization Complete This field indicates a SRAM zeroization is completed. This field is reset by hardware when firmware writes the CNNx16_n_TEST.szerorun field to 0. 1: SRAM zeroization complete | |
| 21 | ballbdone | R/W | 0 | Bias RAM BIST Complete This field indicates a Bias RAM BIST run is completed. This field is reset by hardware when firmware writes the CNNx16_n_TEST.bbistrun field to 0. 1: Bias RAM BIST complete | |
| 20 | tallbdone | R/W | 0 | TRAM BIST Complete This field indicates a TRAM BIST run is completed. This field is reset by hardware when firmware writes the CNNx16_n_TEST.tbistrun field to 0. 1: TRAM BIST complete | |

| CNNx16_n SRAM Test | | | CNNx16_n_TEST | | [0x000C] |
|--------------------|-----------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 19 | mallbdone | R/W | 0 | Mask RAM BIST Complete This field indicates a Mask RAM BIST run is completed. This field is reset by hardware when firmware writes the CNNx16_n_TEST.mbistrun field to 0. 1: Mask RAM BIST complete | |
| 18 | sallbdone | RO | 0 | SRAM BIST Complete This field indicates a SRAM BIST run is completed. This field is reset by hardware when firmware writes the CNNx16_n_TEST.sbistrun field to 0. 1: SRAM BIST complete | |
| 17 | ballbfail | RO | 0 | Bias RAM BIST Result When a Bias RAM BIST operation was started by setting CNNx16_n_TEST.bbistrun bit to 1, and the operation is completed by hardware (CNNx16_n_TEST.ballbfail is set to 1 by hardware), this field indicates the result of the Bias RAM BIST operation. Reset this field by writing a 0 to the CNNx16_n_TEST.bbistrun field. 0: Bias RAM BIST Passed 1: Bias RAM BIST Failed, indicating an error occurred in one of the Bias RAMs. <i>Note: This field is only valid after a Bias RAM BIST operation has started and completed (CNNx16_n_TEST.ballbfail = 1).</i> | |
| 16 | tallbfail | R/W | 0 | SRAM BIST Result When a TRAM BIST operation was started by setting CNNx16_n_TEST.tbistrun bit to 1, and the operation is completed by hardware (CNNx16_n_TEST.tallbfail is set to 1 by hardware), this field indicates the result of the TRAM BIST operation. Reset this field by writing a 0 to the CNNx16_n_TEST.tbistrun field. 0: TRAM BIST Passed 1: TRAM BIST Failed, indicating an error occurred in one of the four SRAMs. <i>Note: This field is only valid after a TRAM BIST operation has started and completes (CNNx16_n_TEST.tallbfail = 1).</i> | |
| 15 | mallbfail | RO | 0 | Mask RAM BIST Result When a Mask RAM BIST operation was started by setting CNNx16_n_TEST.mbistrun to one, and the operation is completed by hardware (CNNx16_n_TEST.mallbfail is set by hardware to 1), this field indicates the result of the SRAM BIST operation. Reset this field by writing a 0 to the CNNx16_n_TEST.mbistrun field. 0: Mask RAM BIST Passed 1: Mask RAM BIST Failed, indicating an error occurred in one of the 16 Mask RAMs. <i>Note: This field is only valid after a Mask RAM BIST operation has started and completed (CNNx16_n_TEST.mallbfail = 1).</i> | |
| 14 | sallbfail | RO | 0 | SRAM BIST Result When a SRAM BIST operation was started by setting CNNx16_n_TEST.sbistrun to one, and the operation is completed by hardware (CNNx16_n_TEST.sallbfail is set by hardware to 1), this field indicates the result of the SRAM BIST operation. Reset this field by writing a 0 to the CNNx16_n_TEST.sbistrun field 0: SRAM BIST Passed 1: SRAM BIST Failed, indicating an error occurred in one of the four SRAMs. <i>Note: This field is only valid after a SRAM BIST operation has started and completed (CNNx16_n_TEST.sallbfail = 1).</i> | |

| CNNx16_n SRAM Test | | | | CNNx16_n_TEST | [0x000C] |
|--------------------|----------|--------|-------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 13:8 | bistsel | R/W | 0 | BIST Controller Status Selection The bits select an individual BIST controller status to be reported in the associated 32 bits of the memory read data bus. <i>CNNx16_n_TEST.bistsel[5]</i> selects the SRAM or Bias Memory BIST group controller statuses, with: <i>CNNx16_n_TEST.bistsel[2:0]</i> selecting the individual SRAM/Bias Memory instance with <i>CNNx16_n_TEST.bistsel[2:0]</i> = 100 selecting the Bias Memory Instance. Control bit <i>CNNx16_n_TEST.bistsel[4]</i> selects the MRAM BIST controller statuses, with <i>CNNx16_n_TEST.bistsel[3:0]</i> selecting the individual RAM instance. Control bit <i>CNNx16_n_TEST.bistsel[3]</i> selects the TRAM BIST controller statuses, with <i>CNNx16_n_TEST.bistsel[3:0]</i> selecting the individual RAM instance. | |
| 7 | bramz | R/W | 0 | BIAS Memory Zeroize Setting this bit to 1 will force the BIST to initialize all Bias memory locations to 0. Completion status can be found in the: <ul style="list-style-type: none"> • <i>CNNx16_n_TEST.sallzdone</i> and • <i>CNNx16_n_TEST.zerodone</i> This bit must be written to 0 to reset the operation prior to writing it to 1. Writing 1 consecutively does not run the memory zeroization again. | |
| 6 | bbistrun | R/W | 0 | Run Bias Memory BIST Setting this bit to 1 will run the BIST for all Bias Memory instances in the CNNx16_n processor. The BIST will run to completion and the status is reported in: <ul style="list-style-type: none"> • <i>CNNx16_n_TEST.ballbdone</i> • <i>CNNx16_n_TEST.ballbfail</i> • <i>CNNx16_n_TEST.bistdone</i> • <i>CNNx16_n_TEST.bistfail</i> If more detailed status is required from the BIST execution, the <i>CNNx16_n_TEST.bistsel</i> field can be used to extract status from an individual BIST controller. This bit must be written to 0 to reset the BIST operation prior to writing it to 1. Writing 1 consecutively does not run the BIST again. | |
| 5 | tramz | R/W | 0 | TRAM Zeroize Setting this bit to 1 will force the BIST to initialize all TRAM memory locations to 0. Completion status can be found in the <i>CNNx16_n_TEST.tallzdone</i> and <i>CNNx16_n_TEST.zerodone</i> status bit. This bit is edge triggered and must be toggled from zero to one to run the BIST. | |

| CNNx16_n SRAM Test | | | CNNx16_n_TEST | | [0x000C] |
|--------------------|----------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 4 | tbistrun | R/W | 0 | TRAM BIST Run Setting this bit to 1 will run the BIST for all TRAM instances in the CNNx16_n processor. The BIST will run to completion and the status reported in: <ul style="list-style-type: none"> • CNNx16_n_TEST.tallbdone • CNNx16_n_TEST.tallbfail • CNNx16_n_TEST.bistdone • CNNx16_n_TEST.bistfail If more detailed status is required from the BIST execution, the CNNx16_n_TEST.bistsel field can be used to extract status from an individual BIST controller. This bit must be written to 0 to reset the BIST operation prior to writing it to 1. Writing 1 consecutively does not run the BIST again. | |
| 3 | mramz | R/W | 0 | MRAM Zeroize Setting this bit to 1 will force the BIST to initialize all MRAM memory locations to 0. Completion status can be found in the: <ul style="list-style-type: none"> • CNNx16_n_TEST.mallzdone and • CNNx16_n_TEST.zerodon This bit must be written to 0 to reset the operation prior to writing it to 1. Writing 1 consecutively does not run the memory zeroization again. | |
| 2 | mbistrun | R/W | 0 | MRAM BIST Run Setting this bit to 1 will run the BIST for all MRAM instances in the CNNx16_n processor. The BIST will run to completion and the status is reported in: <ul style="list-style-type: none"> • CNNx16_n_TEST.mallbdone • CNNx16_n_TEST.mallbfail • CNNx16_n_TEST.bistdone • CNNx16_n_TEST.bistfail If more detailed status is required from the BIST execution, the CNNx16_n_TEST.bistsel field can be used to extract status from an individual BIST controller. This bit must be written to 0 to reset the BIST operation prior to writing it to 1. Writing 1 consecutively does not run the BIST again. | |
| 1 | sramz | R/W | 0 | SRAM Zeroize Setting this bit to 1 will force the BIST to initialize all SRAM memory locations to 0. Completion status can be found in the <ul style="list-style-type: none"> • CNNx16_n_TEST.sallzdone and • CNNx16_n_TEST.zerodone This bit must be written to 0 to reset the operation prior to writing it to 1. Writing 1 consecutively does not run the memory zeroization again. | |

| CNNx16_n SRAM Test | | | CNNx16_n_TEST | | [0x000C] |
|--------------------|----------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 0 | sbistrun | R/W | 0 | <p>SRAM BIST Run Setting this bit to 1 will run the BIST for all SRAM instances in the CNNx16_n processor. The BIST will run to completion and the status is reported in:</p> <ul style="list-style-type: none"> • CNNx16_n_TEST.sallbdone • CNNx16_n_TEST.sallbfail • CNNx16_n_TEST.bistdone • CNNx16_n_TEST.bistfail <p>If more detailed status is required from the BIST execution, the CNNx16_n_TEST.bistse field can be used to extract status from an individual BIST controller.</p> <p>This bit must be written to 0 to reset the BIST operation prior to writing it to 1. Writing 1 consecutively does not run the BIST again.</p> | |

26.5.2.1 CNNx16_n Per Layer Registers (CNNx16_n_Ly)

Each of the four CNNx16 Processor Arrays support up to 32 layers. Each layer is controlled using the CNNx16_n's Layer registers. Each layer includes one instance of each layer register. Each layer register is 32 bits wide (4 bytes). Register names for a given layer are defined by replacing "y" with the layer number ranging from 0 to 31 (32 layers maximum). The offset address of each layer's specific register is determined by the layer's base offset address and adding 4 times the layer number in hex. For example, for the CNNx16_n_Ly Row Count Register, the base offset address is [0x0010]. For layer 21, register CNNx16_n_L21_RCNT, the address offset is [0x0010] + (4 × 0x15) = [0x0064].

Table 26-24: CNNx16_n_Ly Row Count Register

| CNNx16_n_Ly Row Count | | | CNNx16_n_Ly_RCNT | | [0x0010]-[0x008C] |
|-----------------------|----------|--------|------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:18 | - | R/W | 0 | <p>Reserved Do not modify</p> | |
| 17:16 | rcnt_pad | R/W | 0 | <p>Pad Rows This field sets the number of pad rows included at the beginning and end of the frame. <i>Note: The CNNx16_n_Ly_RCNT.rcnt_max is inclusive of two times this value, as the same number of pad rows are included at the beginning and end of the frame. This R/W register is accessible through the CNN APB interface.</i></p> | |
| 15:10 | - | R/W | 0 | <p>Reserved Do not modify</p> | |
| 9:0 | rcnt_max | R/W | 0 | <p>Layer Row Count Maximum This field sets the maximum row count to be processed. Processing begins with row 0 and completes when processing of the row determined by this field is complete. Set this field to include two times the CNNx16_n_Ly_RCNT.rcnt_pad value. $rcnt_max = (2 \times rcnt_pad) + \text{number of rows}$ <i>Note: The value programmed into this field is the effective image row value including pad, but excluding rows not processed due to stride restrictions.</i></p> | |

Table 26-25: CNNx16_n_Ly Column Count Register

| CNNx16_n_Ly Column Count | | | CNNx16_n_Ly_CCNT | | [0x0090]-[0x010C] |
|--------------------------|----------|--------|------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:18 | - | R/W | 0 | Reserved Do not modify | |
| 17:16 | ccnt_pad | R/W | 0 | Layer Column Pad Count This field determines the number of pad columns included at the beginning and end of the frame. Note that the CNNx16_n_Ly_CCNT.ccnt_max is inclusive of two times this value, as the same number of pad columns are included at the beginning and end of the row. This R/W register is accessible through the CNN APB interface. | |
| 15:10 | - | R/W | 0 | Reserved Do not modify | |
| 9:0 | ccnt_max | R/W | 0 | Layer Column Count Maximum When the CNN is enabled, these bits determine the maximum per layer column count to be processed. Processing begins with column 0 and completes when processing of the column determined by CNNx16_n_Ly_CCNT.ccnt_max is complete. The value programmed into this register, via the CNN APB interface, is the effective image column value including pad, but excluding columns not processed due to stride restrictions. | |

Table 26-26: CNNx16_n_Ly One Dimensional Control Register

| CNNx16_n_Ly One Dimensional Control | | | CNNx16_n_Ly_ONED | | [0x0110]-[0x018C] |
|-------------------------------------|-----------|--------|------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:22 | - | RO | 0 | Reserved Do not modify | |
| 21:18 | ewise_cnt | R/W | 0 | Element-Wise Channel Count Determines the number of element-wise channels to be processed. 0: 1 channel 1: 2 channels 2: 3 channels ... 14: 15 channels 15: 16 channels | |
| 17 | 2d_conv | R/W | 0 | 2D Convolution of Element-Wise Result Set this field to 1 to enable 2D convolution of the element-wise result. Standard 2D processing applies. 0: 2D Convolution disabled 1: Enable 2D convolution | |
| 16 | prepool | R/W | 0 | Pre-Pooling of Input Data Set this field to 1 to enable pre-pooling of the input data prior to the element wise operation selected with CNNx16_n_Ly_ONED.ewise_func . 0: Input data is not pre-pooled prior to element-wise function. 1: Pre-pooling of input data prior to element-wise operation enabled | |

| CNNx16_n_Ly One Dimensional Control | | | CNNx16_n_Ly_ONED | | [0x0110]-[0x018C] |
|-------------------------------------|------------|--------|------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 15:14 | ewise_func | R/W | 0 | Element-Wise Function Select Selects the element-wise function performed on the input data if . 0b00: Subtract 0b01: Add 0b10: Bitwise OR 0b11: Bitwise XOR | |
| 13 | ewise_en | R/W | 0 | Element-Wise Enable Set this field to 1 to enable the element-wise operations. Prior to enabling element-wise operations, both the CNNx16_n_Ly_POST.ts cnt_max field and the CNNx16_n_Ly_POST.ts_en fields must be set. 0: Element-wise operation disabled 1: Enable element-wise operation | |
| 12 | oned_en | R/W | 0 | One Dimensional Processing Enable Enables 1D input data processing. If the CNNx16_n_Ly_RCNT.rcnt_max field is non-zero, the row count is used to index the input image, otherwise, the column count, CNNx16_n_Ly_CCNT.ccnt_max , value is used. | |
| 11:8 | oned_width | R/W | 0 | One Dimensional Convolution Mask Width One dimensional convolution mask width (0-9 are valid values). One based value with a width > 0 enabling 1D convolution operation. | |
| 7:4 | oned_sad | R/W | 0 | One Dimensional Convolution Start Mask Address One dimensional convolution start mask address (offset within 9 byte mask width) used in conjunction with the mask start address (CNNx16_n_Ly_MCNT.mc nt_sad) to determine that 1D convolution mask starting address. | |
| 3:0 | tscnt_max | R/W | 0 | Maximum Time Slot Count Maximum timeslot count register. When CNNx16_n_Ly_POST.ts en is set, this value determines the number of timeslots required to output data that has been generated in parallel by the CNNx16_n processors. This count is used for passthru, 1x1, elementwise, and mask sharing (CNNx16_n_Ly_LCTRL0.m slave and CNNx16_n_Ly_LCTRL0.s slave) operations. | |

Table 26-27: CNNx16_n_Ly Pool Row Count Register

| CNNx16_n_Ly Pool Row Count | | | CNNx16_n_Ly_PRCNT | | [0x0190]-[0x020C] |
|----------------------------|-----------|--------|-------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | R/W | 0 | Reserved Do not modify | |
| 3:0 | prcnt_max | R/W | 0 | Pool Row Count Max When the CNN is enabled with one of the CNNx16_n_CTRL.pool_en (global) or CNNx16_n_Ly_LCTRL0.pool_en (per layer) bits set, this field determines the per layer pool row count to be processed. Processing begins with pool row 0 and completes when processing of the pooling row determined by <i>prcnt_max</i> is complete, or the effective pool row count exceed the image row count specified in CNNx16_n_Ly_RCNT.rcnt_max . These count values are added to the row count to determine the effective address of the pooled data. This R/W register is accessible through the CNN APB interface. | |

Table 26-28: CNNx16_n_Ly Pool Column Count Register

| CNNx16_n_Ly Pool Column Count | | | CNNx16_n_Ly_PCCNT | | [0x0210]-[0x028C] |
|-------------------------------|-----------|--------|-------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:4 | - | R/W | 0 | Reserved Do not modify | |
| 3:0 | pccnt_max | R/W | 0 | Pool Column Count Max When the CNN is enabled with one of the <i>CNNx16_n_CTRL.pool_en</i> (global) or <i>CNNx16_n_Ly_LCTRL0.pool_en</i> (per layer) bits set, these bits determine the per layer pool column count to be processed. Processing begins with pool column 0 and completes when processing of the pooling column determined by <i>pccnt_max</i> is complete, or the effective pool column count exceed the image column count specified in <i>CNNx16_n_Ly_CCNT.ccnt_max</i> . These count values are added to the column count to determine the effective address of the pooled data. This R/W register is accessible through the CNN APB interface. | |

Table 26-29: CNNx16_n_Ly Stride Count Register

| CNNx16_n_Ly Stride Count | | | CNNx16_n_Ly_STRIDE | | [0x0290]-[0x030C] |
|--------------------------|--------|--------|--------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:2 | - | R/W | 0 | Reserved Do not modify | |
| 1:0 | stride | R/W | 0 | Stride This field determine the stride length across and down the image. Processing begins with row 0 and column 0. A stride of one is applied until the top row or left column of the receptive field lands in the unpadded image. At that point, the istride value programmed via the APB interface is applied to the column and/or row of the field in the unpadded image. The stride is applied as long as the field remains within the bounds of the padded image. | |

Table 26-30: CNNx16_n_Ly Write Pointer Base Address Register

| CNNx16_n_Ly Write Pointer Base Address | | | CNNx16_n_Ly_WPTR_BASE | | [0x0310]-[0x038C] |
|--|-----------|--------|-----------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |
| 16:0 | wptr_base | R/W | 0 | Write Pointer Base This per layer register sets the CNN convolution result SRAM write pointer base address. The base address can be set to any location in any data SRAM in the CNN. Bit 16 allows the write pointer to not point to any SRAM. This R/W register is accessible through the CNN APB interface. | |

Table 26-31: CNNx16_n_Ly Write Pointer Timeslot Offset Register

| CNNx16_n_Ly Write Pointer Timeslot Offset | | | CNNx16_n_Ly_WPTR_TOFF | | [0x0390]-[0x040C] |
|---|-------|--------|-----------------------|----------------------------------|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |

| CNNx16_n_Ly Write Pointer Timeslot Offset | | | CNNx16_n_Ly_WPTR_TOFF | | [0x0390]-[0x040C] |
|---|-----------|--------|-----------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 16:0 | wptr_toff | R/W | 0 | Write Pointer Timeslot Offset When the CNN is enabled with the CNNx16_n_Ly_POST.ts_en bit set and a non-zero time slot count value loaded into CNNx16_n_Ly_ONED.tscnt_max field, this per layer register sets the CNN convolution result SRAM write pointer timeslot offset value. During a convolution result SRAM data write, this timeslot offset value is multiplied by the timeslot counter and added to the SRAM write address pointer. This offset can be used to determine SRAM write addresses based on timeslot. The offset can be set to any location in any data SRAM in the CNN. This R/W register is accessible through the CNN APB interface. | |

Table 26-32: CNNx16_n_Ly Write Pointer Mask Offset Register

| CNNx16_n_Ly Write Pointer Mask Offset | | | CNNx16_n_Ly_WPTR_MOFF | | [0x0410]-[0x048C] |
|---------------------------------------|-----------|--------|-----------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |
| 16:0 | wptr_moff | R/W | 0 | Write Pointer Mask Offset When the CNN is enabled with the CNNx16_n_Ly_EN.mask_en bit set and a mask count value loaded into the CNNx16_n_Ly_MCNT.mcnt_max register that is greater than the CNNx16_n_Ly_MCNT.mcnt_sad value, this per layer register sets the CNN convolution result SRAM write pointer mask count offset value. During a convolution result SRAM data write, this mask count offset value is multiplied by the mask counter and added to the SRAM write address pointer. This offset can be used to determine SRAM write addresses based on a mask count. The offset can be set to any location in any data SRAM in the CNN. This R/W register is accessible through the CNN APB interface. | |

Table 26-33: CNNx16_n_Ly Write Pointer Multi-Pass Channel Offset Register

| CNNx16_n_Ly Write Pointer Multi-Pass Channel Offset | | | CNNx16_n_Ly_WPTR_CHOFF | | [0x0490]-[0x050C] |
|---|------------|--------|------------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |
| 16:0 | wptr_choff | R/W | 0 | Write Pointer Multi-Pass Offset When the CNN is enabled with the CNNx16_n_Ly_EN.mask_en bit set and a programmed maximum mask counter value (CNNx16_n_Ly_MCNT.mcnt_max - CNNx16_n_Ly_MCNT.mcnt_sad) that is greater than the maximum number of available processors programmed into the CNNx16_n_Ly_LCTRL1.xpch_max register (output channel multi-pass is enabled), the rounded mask counter value is divided by the CNNx16_n_Ly_LCTRL1.xpch_max value and multiplied by the CNNx16_n_Ly_WPTR_CHOFF.wptr_choff to create a multi-pass channel offset value. During a convolution result SRAM data write, this multi-pass channel offset value is added to the SRAM write address pointer. This offset can be used to determine SRAM write addresses based on a multi-pass count value. The offset can be set to any location in any data SRAM in the CNN. This R/W register is accessible through the CNN APB interface. | |

Table 26-34: CNNx16_n_Ly Read Pointer Base Address Register

| CNNx16_n_Ly Read Pointer Base Address | | | CNNx16_n_Ly_RPTR_BASE | | [0x0510]-[0x058C] |
|---------------------------------------|-----------|--------|-----------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |
| 16:0 | rptr_base | R/W | 0 | Read Pointer Base Address When the CNN is enabled, this per layer register sets the CNN convolution result SRAM read pointer base address. The base address can be set to any location in the SRAM dedicated to a specific CNN input channel processor. <i>Note: This field is limited to the 8192 Bytes of SRAM in the MAX78000. Do not write values greater than the memory available.</i> This R/W register is accessible through the CNN APB interface. | |

Table 26-35: CNNx16_n_Ly Layer Control 0 Register

| CNNx16_n_Ly Layer Control 0 | | | CNNx16_n_Ly_LCTRL0 | | [0x0590]-[0x060C] |
|-----------------------------|-----------|--------|--------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |
| 16 | bigdwrt | R/W | 0 | Big Data Write Enables writing out the current output channel in 32-bit accumulator form. This bit allows full resolution of the output layer to be written to assist with software defined SoftMax operation. | |
| 15:12 | cnnsi_en | R/W | 0 | CNN 26-Bit Non-Scaled Non-Quantized Sum of Products Feed When set, this field enables the associated CNNx16_n 26-bit non-scaled, non-quantized sum of products data into the output accumulator, allowing sums of 16, 32, 48, or 64 products. Each bit is associated with one of the remaining 3 CNNx16_n processors. In processor 0, bit 1 is associated with CNNx16_n processor 1, bit 2 with CNNx16_n processor 2, and bit 3 with CNNx16_n processor 3. In processor 1, bit 1 is associated with CNNx16_n processor 2, bit 2 with CNNx16_n processor 3, and bit 3 with CNNx16_n processor 0, and so on. When these bits are set to zero, the internal state of the 26-bit data bus is forced to zero. 0b0000: 26-bit data bus set to 0 0b0001-0b1111: See description | |
| 11 | sramlsrc | R/W | 0 | SRAM CNNx16_n SRAM Global Write Source Data When set, SRAM data is sourced from the global data busses. The device supports 4 global data busses, one from each of the CNNx16_n processors. When all four CNNx16_n processors are used together to form a single sum-of-products value, all four CNNx16_n outputs an identical address, and based on the natural priority of the decode, processor 0 sources the SRAM write data. | |
| 10 | cpad_only | R/W | 0 | Input Frame Column Pad When set, padding is applied only to the input frame columns. In this case, row padding is ignored. This bit is intended to be used for parallel processing. <i>Note: When this field is set, row padding is ignored.</i> | |

| CNNx16_n_Ly Layer Control 0 | | | | CNNx16_n_Ly_LCTRL0 | [0x0590]-[0x060C] |
|-----------------------------|----------|--------|-------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 9 | act_en | R/W | 0 | ReLU Activation Eanble When set, ReLU activation is enabled for each output channel. Activation is applied to the scaled and quantized data. 0: ReLU not enabled 1: ReLU activation enabled for each output channel and is applied to the scaled and quantized data. | |
| 8 | maxpl_en | R/W | 0 | Max Pooling Enable When set to 1, Max Pooling is selected as the pooling mode. In Max Pooling mode, the maximum value in the pool is selected for the field and written into the TRAM. When this field is set to 0, average pooling mode is selected. In Average Pooling Mode, the average of all pooled values is calculated and selected for use in the field. 0: Average Pooling Mode selected. 1: Max Pooling Mode selected <i>Note: This field is only used if the CNNx16_n_Ly_LCTRL0.pool_en field is set to 1.</i> | |
| 7 | pool_en | R/W | 0 | Enable Pooling Setting this bit enables pooling for the associated layer. Pool dimensions are determined by the pool row and column count maximums (<i>CNNx16_n_Ly_PRCNT.prcnt_max</i> and <i>CNNx16_n_Ly_PCCNT.pccnt_max</i>). 0: Pooling disabled 1: Pooling enabled <i>Note: The type of pooling performed is determined by the CNNx16_n_Ly_LCTRL0.maxpl_en field as either average pooling or max pooling.</i> | |
| 6 | parallel | R/W | 0 | Parallel Mode Enable Setting this bit to one enables a single input channel's data to use 4 bytes of memory instead of one. In parallel mode, data is read in byte order from byte 0 to byte 3, and then by memory depth. The purpose of the mode is to allow additional memory to be used for the input layer data to support larger images. When set, the receptive field for the data will be generated in the first processor in each group of 4 processors, provided the processor is enabled. | |
| 5 | master | R/W | 0 | CNNx16_n Processor Group Master Select Enables a CNNx16_n group of processors to independently calculate a sum-of-products result for all adjacent ascending CNNx16_n processor groups not configured for master operation. | |
| 4 | mslave | R/W | 0 | CNNx16_n Processor 0 Mask Leader When this bit is set, all 16 processors within the CNNx16_n share the receptive field with processor 0. This allows the generation of additional output channels using the mask values distributed across the 16 processors. Each processor applies a 3x3 mask of the selected width to be applied to the processor 0 field. <i>Note: Use of timeslots is required to write the parallel generated output channels to memory.</i> | |

| CNNx16_n_Ly Layer Control 0 | | | CNNx16_n_Ly_LCTRL0 | | [0x0590]-[0x060C] |
|-----------------------------|--------|--------|--------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 3:0 | sslave | R/W | 0 | CNNx16_n Processor Group Slave Mode Within a CNNx16_n, this field enables each of the 4 groups of four processors to share the receptive field with the first processor of the x4 group (channels 0,4,8,12). When set, the receptive field of the first processor in the associated x4 group to be passed to the remaining 3 processors in the x4 group. Masks associated with the slaved group of three processors can be applied to the field of the first processor and additional output channels generated. Use of the timeslot counter is required to write the additional output channels to memory. | |

Table 26-36: CNNx16_n_Ly Layer Mask Count Register

| CNNx16_n_Ly Mask Count | | | CNNx16_n_Ly_MCNT | | [0x0610]-[0x068C] |
|------------------------|----------|--------|------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | mcnt_sad | R/W | 0 | Layer Mask SRAM start Address Mask SRAM address counter increments sequentially from this word and bit address during layer processing. Counter restarts each stride and increments once per output channel: <ul style="list-style-type: none"> • <i>mcnt_sad[15:4]</i>→SRAM word (72bits) address • <i>mcnt_sad[2:0]</i>→bit address | |
| 15:0 | mcnt_max | R/W | 0 | Mask SRAM Layer Maximum Address Mask SRAM address counter increments sequentially by word and bit address during layer processing to this address. Counter restarts each stride and increments once per output channel: <ul style="list-style-type: none"> • <i>mcnt_max[15:4]</i>→SRAM word (72bits) address • <i>mcnt_max[2:0]</i>→bit address. | |

Table 26-37: CNNx16_n_Ly TRAM Pointer Register

| CNNx16_n_Ly TRAM Pointer | | | CNNx16_n_Ly_TPTR | | [0x0690]-[0x070C] |
|--------------------------|----------|--------|------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:27 | - | R/W | 0 | Reserved Do not modify | |
| 26:16 | tptr_sad | R/W | 0 | TRAM Start Address This field determines the per layer TRAM pointer start address for initial processing and rollover events. | |
| 15:11 | - | R/W | 0 | Reserved Do not modify | |
| 10:0 | tptr_max | R/W | 0 | TRAM Max Address This field determines the rollover point of the TRAM address pointer. This field, <i>tptr_max</i> , is used together with the TRAM address pointer start address value (CNNx16_n_Ly_TPTR.tptr_sad) to reflect the usable input image row size including pad. | |

Table 26-38: CNNx16_n_Ly Enable Register

| CNNx16_n_Ly Enable | | | CNNx16_n_Ly_EN | | [0x0710]-[0x078C] |
|--------------------|---------|--------|----------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:16 | mask_en | R/W | 0 | CNNx16_n Processor Mask Enable Each bit in this per layer field enables the state of one processor's kernel logic in the CNNx16_n. Bit 0 controls processor 0's mask application (the master processor) and bit 15 controls processor 15's mask application. | |
| 15:0 | pro_en | R/W | 0 | CNNx16_n Processor Enable Each bit in this per layer field controls the enable state of one processor in the CNNx16_n. Bit 0 controls processor 0's (the master processor) enable and bit 15 controls processor 15's enable. | |

Table 26-39: CNNx16_n_Ly Post Processing Register

| CNNx16_n_Ly Post Processing | | | CNNx16_n_Ly_POST | | [0x0790]-[0x080F] |
|-----------------------------|------------|--------|------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31: 29 | - | R/W | 0 | Reserved Do not modify | |
| 28 | deconv_en | R/W | 0 | Deconvolution Enable Virtually expands the input image size by adding a 0 byte after each actual input data byte is shifted into the TRAM, and a row of 0s following each row of column expanded input data is shifted into the TRAM. The receptive field remains at 3x3 scanned across the expanded data. 0: Deconvolution disabled 1: Deconvolution enabled | |
| 27 | flatten_en | R/W | 0 | Flatten Enable Enables flattening all of the input channel data supporting a series of 1x1 convolutions emulating a fully connected network. Setting this bit forces the use of an extended multi-pass count allowing for up to 256 neurons. This bit is used in conjunction with the CNNx16_n_Ly_LCTRL1.inpcchexp , CNNx16_n_Ly_POST.xpmp_cnt , and CNNx16_n_Ly_POST.onexone_en fields to enable Multi-Layer Processing. | |
| 26 | out_abs | R/W | 0 | Absolute Value Output Enable Convert the scaled, quantized convolution output to an absolute value. This bit, along with the activation enable bit, CNNx16_n_Ly_LCTRL0.act_en) determine the final processing operation prior to writing the out channel to memory. 0: Output is not converted to an absolute value 1: Output is converted to an absolute value <i>Note: The CNNx16_n_Ly_POST.out_abs has priority over the activation enable bit, CNNx16_n_Ly_LCTRL0.act_en.</i> | |
| 25 | onexone_en | R/W | 0 | Pass-Thru/1x1 Convolution Mode Enable This bit forces all sixteen processors in the CNNx16_n to either directly pass-thru the result of the pooling logic or compute a 1 data byte by 1 byte (1,2,4,8 bit) weight product. Control of a pass-thru or 1x1 convolution is made for each of the 16 processors using the CNNx16_n_Ly_EN.mask_en control field. 0: Pass-Thru Enabled 1: 1x1 Enabled | |

| CNNx16_n_Ly Post Processing | | | CNNx16_n_Ly_POST | | [0x0790]-[0x080F] |
|-----------------------------|------------|--------|------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 24 | ts_en | R/W | 0 | Timeslot Mode Enable This bit is used to enable the timeslot counter. When enabled, the number of timeslots programmed into the timeslot counter, <i>CNNx16_n_Ly_ONED.ts cnt_max</i> , are added to each output channel slot. The timeslot counter allows pass-thru, 1x1, and elementwise values calculated in parallel to be written sequentially to memory. 0: Timeslot Mode Disabled 1: Timeslot Mode Enabled | |
| 23:22 | mask_size | R/W | 0 | Mask Size Selection This field determines the mask size multiplied with each 8-bit data value in the convolution. 0b00: 8-bits 0b01: 1-bit 0b10: 2-bits 0b11: 4-bits | |
| 21:18 | xpmp_cnt | R/W | 0 | Expanded Multi-Pass (MP) Count Adds 4 additional bits to the MP counter for flattening (MLP) operations. Only used when the <i>CNNx16_n_Ly_POST.flatten_en</i> bit is set to 1. This field is appended to the <i>CNNx16_n_Ly_LCTRL1.inpcchexp</i> bits and make up the 4 most significant bits of the count. | |
| 17 | scale_shft | R/W | 0 | Scale Shift Control This bit selects the shift direction of the pre-activation sum-of-products result of the convolution. 0: Left Shift 1: Right Shift | |
| 16:13 | scale_reg | R/W | 0 | Scale Shift Number This field sets the number of bits to shift the pre-activation sum-of-products result of the convolution. Valid values are 0 to 16-bits and the direction of the shift is controlled by <i>CNNx16_n_Ly_POST.scale_shift</i> . | |
| 12 | bptr_en | R/W | 0 | Bias Enable Enables addition of a scaled bias, stored in each Bias location, to the result of the convolution. Bias values are automatically scaled by a shift left of seven bits in hardware. | |
| 11:0 | bptr_sad | R/W | 0 | Bias Pointer Start Address Bias register file address byte counter increments once each mask count increment. Note that the x16 Bias values can be enabled and used independently across the 4 output processors. | |

Table 26-40: CNNx16_n_Ly Layer Control 1 Register

| CNNx16_n_Ly Layer Control 1 | | | CNNx16_n_Ly_LCTRL1 | | [0x0A10]-[0x0A8C] |
|-----------------------------|-------|--------|--------------------|----------------------------------|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |

| CNNx16_n_Ly Layer Control 1 | | | CNNx16_n_Ly_LCTRL1 | | [0x0A10]-[0x0A8C] |
|-----------------------------|----------|--------|--------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 16:8 | xpch_max | R/W | 0 | Expansion Mode Maximum Processors Selects the maximum channel processor number used in channel expansion mode. This allows for fewer than 64 processors to be used in a multi-pass or channel expansion configuration. <i>Note: Processor management was shown to be important for mask management when input channel processing draws on a relatively small number of channels for a potentially large number of masks.</i> 0-64: Selects the number of processors to use for channel expansion mode <i>Note: This field contains 9 bits, the upper 3 bits are reserved for future use. Only values up to 64 are supported.</i> | |
| 7:4 | wptr_inc | R/W | 0 | Write Pointer Increment This field's value determines the increment for the write pointer counter after all output channels within a given stride are written to memory. Non-zero values in this field are used for multi-pass operations. <i>Note: The Write Pointer is always incremented by 1 or by 4 in parallel mode. The value in this field is added to the internal write pointer increment.</i> | |
| 3:0 | inpchexp | R/W | 0 | Multi-Pass Input Channel Expansion Count This field determines the number of sequential memory locations associated with a single convolution. For example, if a value of 15 (0xf) is programmed into this field, 16 channels are assumed to be sequentially stored in memory (channel then byte order) for each of the 64 processors. This allows for up to 1024 input channels to be processed in a single convolution. Similarly, if this field is set to 1, 2 channels are assumed to be sequentially stored in memory (channel then byte order) as channels for the convolution, totaling 128 channels if all 64 processors are used. A value of zero disables the multi-pass feature allowing for a single channel for each processor. | |

Table 26-41: CNNx16_n Muchselator Data Register

| CNNx16_n Muchselator Data | | | CNNx16_n_MLAT | | [0x1000] |
|---------------------------|---------|--------|---------------|---|----------|
| Bits | Field | Access | Reset | Description | |
| 31:0 | mlatdat | RO | 0 | Packed Channel Data This register accumulates four bytes of output channel data to be read by the host MCU. Channel data is usually stored in the memory depth of a single byte of the SRAM data word, and four channels make up the memory data word. The Muchselator automatically fetches four bytes in the memory depth to generate a packed 4-byte word for efficient reading by the MCU. The target channel is selected using the CNNx16_n_CTRL.mlatch_sel bits and the target SRAM address is determined by the CNNx16_n_Ly_WPTR_BASE register. Setting the CNNx16_n_CTRL.mlat_id bit to 1 loads the CNNx16_n_Ly_WPTR_BASE.wptr_base value into the address counter and initiates the read of the first 4 bytes of channel data. When the current 4 bytes are accumulated in the CNNx16_n_MLAT.mlatdat is read via the APB interface, the next 4 bytes are read in sequence. Reading continues until halted by the MCU. Four clock cycles are required after the completion of the last read (or setting of the CNNx16_n_CTRL.mlat_id bit to 1 to accumulate the 4 bytes of data. It is up to the MCU firmware to ensure there is adequate time between reads to accumulate the 4 bytes of data. | |

26.5.2.2 CNNx16_n Processor Stream Registers

Each of the four CNNx16 Processor arrays support up to 8 simultaneous streams for the first 8 layers. Each stream is controlled using the CNNx16_n's Stream registers. Each stream includes one instance of each Stream register. Each Stream register is 32 bits wide (4 bytes). Register names for a given stream are defined by replacing "z" with the stream number ranging from 0 to 7 (8 streams maximum). The offset address of each stream's specific register is determined by the stream's base offset address and adding 4 times the stream number in hex. For example, for the CNNx16_n_Sz Stream Control 0 Register, the base offset address is [0x0810]. For Stream 5, register CNNx16_n_S5_STRM0, the address offset is [0x0810] + (4 × 0x05) = [0x0824].

Table 26-42: CNNx16_n_Sz Stream Control 0 Register

| CNNx16_n_Sz Stream Control 0 | | | CNNx16_n_Sz_STRM0 | | [0x0810]-[0x082C] |
|------------------------------|------------|--------|-------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:14 | - | R/W | 0 | Reserved Do not modify | |
| 13:0 | strm_isval | R/W | 0 | Per Stream Start Count The per stream start count is based on the previous layer's .tptr_inc count. When streaming is enabled (CNNx16_n_CTRL.stream_en = 1), each time a byte in the prior or input layer is written into the TRAM, the internal stream start counter is incremented. When the counter reaches this field's value, CNNx16_n_Sz_STRM0.strm_isval , processing of the current layer is enabled. This mechanism allows adequate receptive field data to be accumulated before convolution processing in a stream layer begins. Once the counter value reaches this field's value, CNNx16_n_Sz_STRM0.strm_isval , counting is halted and the terminal count value remains in the counter. The R/W register is accessible through the APB interface. | |

Table 26-43: CNNx16_n_Sz Stream Control 1 Register

| CNNx16_n_Sz Stream Control 1 | | | CNNx16_n_Sz_STRM1 | | [0x0890]-[0x08AC] |
|------------------------------|-------------|--------|-------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:28 | - | R/W | 0 | Reserved Do not modify | |
| 27:16 | strm_dsval2 | R/W | 0 | Per Stream Multi-Row Delta Count This field is based on the previous layer's CNNx16_n_Ly_TPTR count. When streaming is enabled (CNNx16_n_CTRL.stream_en = 1), this field determines the number of bytes written into the TRAM of the prior layer between active processing of the current layer. This APB accessible R/W register is used to set the processing cadence across an image row boundary. When the internal delta count counter reaches the value stored in this field, CNNx16_n_Sz_STRM1.strm_dsval2 , processing of the current layer is enabled for one stride (input pooling plus output channel generation) and the delta counter is reset to zero. | |
| 15:9 | - | R/W | 0 | Reserved Do not modify | |

| CNNx16_n_Sz Stream Control 1 | | | CNNx16_n_Sz_STRM1 | | [0x0890]-[0x08AC] |
|------------------------------|-------------|--------|-------------------|---|-------------------|
| Bits | Field | Access | Reset | Description | |
| 8:4 | strm_dsval1 | R/W | 0 | Per Stream In-Row Delta Count This field is based on the previous layer's tptr_inc count. When streaming is enabled (CNNx16_n_CTRL.stream_en = 1), this field determines the number of bytes written into the TRAM of the prior layer between active processing of the current layer. This APB accessible R/W register is used to set the processing cadence of each column across an image row. When the internal delta count counter reaches the value stored in this field, CNNx16_n_Sz_STRM1.strm_dsval1 , processing of the current layer is enabled for one stride (input pooling plus output channel generation) and the delta counter is reset to zero. | |
| 3:0 | strm_invol | R/W | 0 | Per Stream Input Volume Offset This field is based on the stream count. When streaming is enabled (CNNx16_n_CTRL.stream_en = 1), this field determines the input volume offset applied to each stream. The value programmed into this field is multiplied by the stream count to calculate the stream count input volume offset. The CNN supports up to 16 independent input volumes. The input volumes are split between multi-pass and streaming. The streaming input volume offset allows the streaming input volume selection to "skip over" the input volumes used for multi-pass processing. | |

Table 26-44: CNNx16_n_Sz Stream Frame Buffer Size Register

| CNNx16_n_Sz Stream Frame Buffer Size | | | CNNx16_n_Sz_FBUF | | [0x0910]-[0x092C] |
|--------------------------------------|----------|--------|------------------|--|-------------------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify this field. | |
| 16:0 | fbuf_max | R/W | 0 | Per Stream Circular Buffer Max Count When streaming is enabled (CNNx16_n_CTRL.stream_en = 1), the per-layer SRAM read pointer base register value (CNNx16_n_Ly_RPTR_BASE.rptr_base) is subtracted from the internally generated read pointer counter and compared to the value stored in this field. When the adjusted read pointer is equal to this field, the rollover point is detected, and the CNN_FIFO_STAT.rptr_base value is loaded into the read pointer counter. The R/W register is accessible through the APB interface. | |

Table 26-45: CNNx16_n Input FIFO Frame Size Register

| CNNx16_n Input FIFO Frame Size | | | CNNx16_n_IFRM | | [0x0990] |
|--------------------------------|-------|--------|---------------|----------------------------------|----------|
| Bits | Field | Access | Reset | Description | |
| 31:17 | - | R/W | 0 | Reserved Do not modify | |

| CNNx16_n Input FIFO Frame Size | | | CNNx16_n_IFRM | | [0x0990] |
|--------------------------------|----------|--------|---------------|--|----------|
| Bits | Field | Access | Reset | Description | |
| 16:0 | ifrm_reg | R/W | 0 | Streaming Input Frame Size Byte Count When streaming is enabled (<i>CNNx16_n_CTRL.stream_en</i> = 1), this field determines the number of bytes read from the data FIFOs. An internal counter counts the number of input frame bytes read from the input FIFOs and compares the count to the value in this register. Once the value in this field is reached, the terminal count value is retained, incrementing of this counter is inhibited, and processing of the input data is terminated. This R/W register is accessible through the APB interface. | |

27. Revision History

Table 27-1: Revision History

| REVISION NUMBER | REVISION DATE | DESCRIPTION | PAGES CHANGED |
|-----------------|---------------|--|---------------|
| 0.1 | 5/20/20 | Initial Release | - |
| 0.2 | 5/21/20 | Added Semaphore, AES, TRNG, DAP and CRC | - |
| 0.2a | 5/22/20 | Added back WDT | - |
| 0.3 | 06/23/20 | Added Outline CNN Chapter with all Registers and Fields. Added CNN Memory Map to Memory Chapter | - |
| 0.3a | 06/29/20 | Updated operating mode diagrams. Corrected links to instance tables in register description sections. Updated memory map diagrams for code and data memory (CM4 only at present.) RV32 memory map diagrams pending. | - |
| 0.4 | 08/20/20 | Updated ADC, System, Memory, SPI and several other chapters to a lesser extent. | - |
| 0.5 | 08/31/20 | Fixed reference names to registers in system chapter that were showing incomplete register names. Updated SPI chapter to have matching register and field names to msdk spi register names. Added Flash registers for Read Lock and Write Lock Removed AES Key Generation from TRNG chapter | - |